IAR Embedded Workbench™ Version 3+ for MSP430™

User's Guide

Literature Number: SLAU138AC
June 2004–Revised September 2013

Contents

List of Figures

List of Tables

Read This First

About This Manual

This manual describes the use of IAR Embedded Workbench™ (EW430) with the MSP430™ ultra-low-power microcontrollers.

How to Use This Manual

Read and follow the instructions in the Get Started Now! chapter. This chapter provides instructions on installing the software, and describes how to run the demonstration programs. After you see how quick and easy it is to use the development tools, TI recommends that you read all of this manual.

This manual describes only the setup and basic operation of the software development environment, but it does not fully describe the MSP430 microcontrollers or the complete development software and hardware systems. For details of these items, see the appropriate TI and IAR™ documents listed in Related Documentation From Texas Instruments, *Important MSP430 Documents on the CD-ROM and Web*.

This manual applies to the use with Texas Instruments' MSP-FET430UIF, MSP-FET430PIF, and eZ430 development tools series.

These tools contain the most up-to-date materials available at the time of packaging. For the latest materials (including data sheets, user's guides, software, and application information), visit the TI MSP430 web site at www.ti.com/msp430 or contact your local TI sales office.

Information About Cautions and Warnings

This book may contain cautions and warnings.

CAUTION

This is an example of a caution statement.

A caution statement describes a situation that could potentially damage your software or equipment.

WARNING

This is an example of a warning statement.

A warning statement describes a situation that could potentially cause harm to you.

The information in a caution or a warning is provided for your protection. Read each caution and warning carefully.

Related Documentation From Texas Instruments

MSP430 development tools documentation

MSP430 Hardware Tools User's Guide, literature number SLAU278
eZ430-F2013 Development Tool User's Guide, literature number SLAU176
eZ430-RF2480 User's Guide, literature number SWRA176
eZ430-RF2500 Development Tool User's Guide, literature number SLAU227
eZ430-RF2500-SEH Development Tool User's Guide, literature number SLAU273
eZ430-Chronos Development Tool User's Guide, literature number SLAU292

MSP430 device data sheets

MSP430x1xx Family User's Guide, literature number SLAU049

MSP430x2xx Family User's Guide, literature number SLAU144

MSP430x3xx Family User's Guide, literature number SLAU012

MSP430x4xx Family User's Guide, literature number SLAU056

MSP430x5xx and MSP430x6xx Family User's Guide, literature number SLAU208

CC430 device data sheets

CC430 Family User's Guide, literature number SLAU259

If You Need Assistance

Support for the MSP430 devices and the FET development tools is provided by the Texas Instruments Product Information Center (PIC). Contact information for the PIC can be found on the TI web site at www.ti.com/support. The Texas Instruments E2E Community support forums for the MSP430 is available to provide open interaction with peer engineers, TI engineers, and other experts. Additional device-specific information can be found on the MSP430 web site.

NOTE: KickStart™ is supported by Texas Instruments.

Although KickStart is a product of IAR, Texas Instruments provides the support for it. Therefore, please do not request support for KickStart from IAR. Consult the extensive documentation provided with KickStart before requesting assistance.

Get Started Now!

This chapter provides instruction on installing the software, and shows how to run the demonstration programs.

Topic **Page**

1.1 Software Installation

Follow the instructions on the supplied READ ME FIRST document to install the IAR Embedded Workbench™ KickStart. Read the file <Installation Root>\Embedded Workbench x.x\430\doc\readme.htm from IAR for the latest information about the Workbench. The term KickStart refers to the function-limited version of Embedded Workbench (including C-SPY™ debugger). KickStart is supplied on the CD-ROM included with each FET, and the latest version is available from the MSP430 web site.

The documents mentioned in the previous paragraph (and this document) can be accessed using: Start → Programs → IAR Systems → IAR Embedded Workbench KickStart for MSP430 V3.

KickStart is compatible with Windows 2000 (SP4), Windows XP (32 bit and 64 bit), Windows Vista (32 bit and 64 bit), and Windows 7 (32 bit and 64 bit). However, the USB FET interface works with only Windows XP (32 bit and 64 bit), Windows Vista (32 bit and 64 bit), and Windows 7 (32 bit and 64 bit).

1.2 Flashing the LED

This section demonstrates on the FET the equivalent of the C-language "Hello World!" introductory program. An application that flashes the LED is developed and downloaded to the FET, and then run.

1. Start the Workbench (Start → Programs → IAR Systems → IAR Embedded Workbench KickStart for MSP430 V3 → IAR Embedded Workbench).
2. Click File → Open Workspace to open the file at: <Installation Root>\Embedded Workbench x.x\430\FET_examples\Flashing the LED.eww. The workspace window opens.
3. Click on the tab at the bottom of the workspace window that corresponds to the MSP430 device (MSP430xxxx) and desired language (assembler or C) to set a project active (see Figure 1-1).

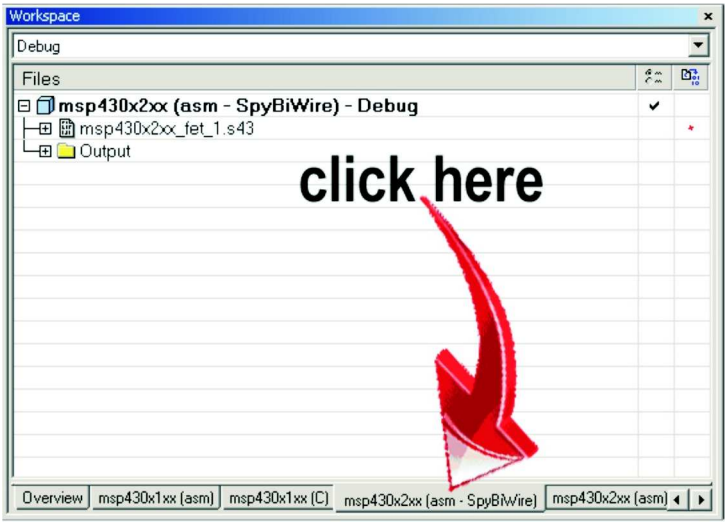

Figure 1-1. Activate Project

Alternatively, right click to activate a project in the Workspace Overview tab (see Figure 1-2).

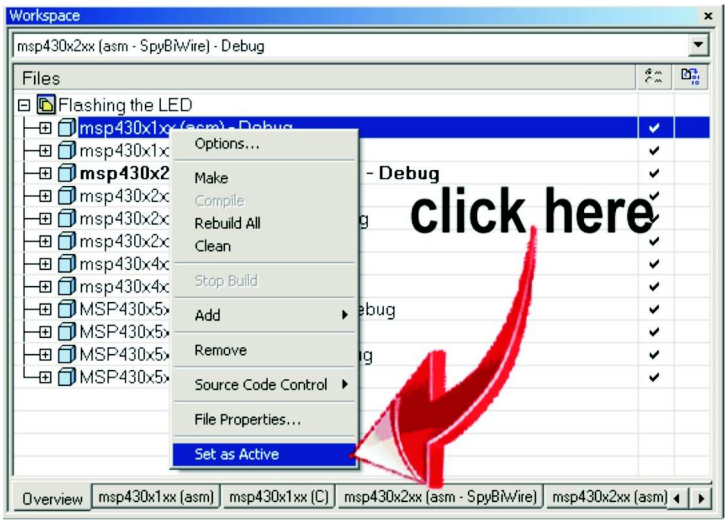

Figure 1-2. Activate Project in Workspace Overview

4. Click Project → Options → FET Debugger → Setup → Connection to select the appropriate port: Texas Instruments LPT-IF for the parallel FET Interface (MSP-FET430PIF) or Texas Instruments USB-IF for the USB Interface (MSP-FET430UIF) or for the eZ430.
5. Click Project → Rebuild All to build and link the source code. You can view the source code by double-clicking on the project, and then double-clicking on the displayed source file.
6. Click Project → Debug to start the C-SPY debugger. C-SPY erases the device flash and then downloads the application object file to the device flash.

   See FAQ Debugging #1 if C-SPY is unable to communicate with the device.
7. Click Debug → Go to start the application. The LED should flash.
8. Click Debug → Stop Debugging to stop debugging, to exit C-SPY, and to return to the Workbench.
9. Click File → Exit to exit the Workbench.

Congratulations, you have just built and tested an MSP430 application!

1.3 Important MSP430 Documents on the CD-ROM and Web

The primary sources of MSP430 information are the device-specific data sheet and user's guide. The most up-to-date versions of these documents that are available at the time of production are provided on the CD-ROM included with this tool. The MSP430 web site (www.ti.com/msp430) contains the most recent version of these documents.

PDF documents describing the IAR tools (Workbench and C-SPY, the assembler, the C compiler, the linker, and the librarian) are in the common\doc and 430\doc folders. Supplements to the documents (that is, the latest information) are available in HTML format in the same directories. 430\doc\readme_start.htm provides a convenient starting point for navigating the IAR documentation.

Development Flow

This chapter describes how to use KickStart to develop application software and how to use C-SPY to debug it.

Topic **Page**

2.1 Overview

Applications are developed in assembler or C using the Workbench, and they are debugged using C-SPY. C-SPY is seamlessly integrated into the Workbench. However, it is more convenient to make the distinction between the code development environment (Workbench) and the debugger (C-SPY). C-SPY can be configured to operate with the FET (that is, an actual MSP430 device) or with a software simulator of the device. KickStart refers to the Workbench and C-SPY collectively. The KickStart software tools are a product of IAR.

Documentation for the MSP430 family and KickStart is extensive. The CD-ROM supplied with this tool contains a large amount of documentation describing the MSP430. The MSP430 home page (www.ti.com/msp430) is another source of MSP430 information. The components of KickStart (workbench and debugger, assembler, compiler, linker) are fully documented in <Installation Root>\Embedded Workbench x.x\common\doc and <Installation Root>\Embedded Workbench\430\doc. .htm files located throughout the KickStart directory tree contain the most up-to-date information and supplement the PDF files. In addition, KickStart documentation is available online via Help.

Read Me First files from IAR and TI and this document can be accessed using Start → Programs → IAR Systems → IAR Embedded Workbench KickStart for MSP430 V3.

Tool	User's Guide	Most Up-To-Date Information
Workbench, C-SPY	EW430_UsersGuide.pdf	readme.htm, ew430.htm, cs430.htm, cs430f.htm
Assembler	EW430_AssemblerReference.pdf	a430.htm, a430_msg.htm
Compiler	EW430_CompilerReference.pdf	icc430.htm, icc430_msg.htm
C library		CLibrary.htm
Linker and Librarian	xlink.pdf	xlink.htm, xman.htm, xar.htm

2.2 Using KickStart

The KickStart edition is a special starter kit or evaluation version of IAR Embedded Workbench with limitations both in code size and in the service and support that is provided. Limitations:

- The C compiler does not generate an assembly code list file.
- The code size limit of the MSP430 IAR KickStart C/C++ Compiler is set to 4Kbytes for traditional MSP430 devices and 8Kbytes for MSP430X devices (see Table 2-1 for detailed information about which MSP430 device is based on which architecture).
- The IAR Assembler delivered is the full version without any restrictions.
- The IAR XLINK Linker links a maximum of 4Kbytes originating from C source code for traditional MSP430 devices and 8Kbytes for MSP430X devices (see Table 2-1 for detailed information about which MSP430 device is based on which architecture), but an unlimited amount of code originating from assembly code.
- The IAR KickStart C-SPY Simulator reads a maximum of 4Kbytes originating from C code for traditional MSP430 devices and 8Kbytes for MSP430X devices but is unlimited in the amount of assembly code read (see Table 2-1 for detailed information about which MSP430 device is based on which architecture).
- MISRA C is not available.
- The runtime library source code is not included.

A full (that is, unrestricted) version of the software tools can be purchased from IAR. A mid-featured tool set – called Baseline, with a 12Kbyte C-code size limitation and basic floating-point operations – is also available from IAR. See the IAR web site (www.iar.se) for more information.

2.2.1 Project Settings

The settings required to configure the Workbench and C-SPY are numerous and detailed. Read and thoroughly understand the documentation supplied by IAR when dealing with project settings. Review the project settings of the supplied assembler and C examples (the project settings are accessed using Project → Options with the project name selected). Use these project settings as templates when developing your own projects. Note that if the project name is not selected when settings are made, the settings are applied to the selected file (not to the project).

The following project settings are recommended or required:

- Specify the target device (General Options → Target → Device).
- Enable an assembler project or a C or assembler project (General Options → Target → Assembler-only project).
- Enable the generation of an executable output file (General Options → Output → Output file → Executable).
- To most easily debug a C project, disable optimization [C/C++ Compiler → Optimizations → Size → None (Best debug support)].
- Enable the generation of debug information in the compiler output (C/C++ Compiler → Output → Generate debug information).
- Specify the search path for the C preprocessor (C/C++ Compiler → Preprocessor → Include Paths).
- Enable the generation of debug information in the assembler output (Assembler → Output → Generate Debug Info).
- Specify the search path for the assembler preprocessor (Assembler → Preprocessor → Include Paths).
- To debug the project using C-SPY, specify a compatible format [Linker → Output → Format → Debug information for C-SPY (with runtime control modules or with I/O emulation modules)].
- Specify the search path for any used libraries (Linker → Config → Search paths).
- Specify the C-SPY driver. Select Project → Options → Debugger → Setup → Driver → FET Debugger to debug on the FET (that is, MSP430 device). Select Simulator to debug on the simulator. If FET Debugger is selected, use Project → Options → FET Debugger → Setup → Connection to select the appropriate port: Texas Instruments LPT-IF for the parallel FET Interface (MSP-FET430PIF) or Texas Instruments USB-IF for the USB Interface (MSP-FET430UIF) or for the eZ430.
- Enable the Device Description file. This file makes C-SPY "aware" of the specifics of the device it is debugging. This file corresponds to the specified target device (Debugger → Setup → Device description file → Override default).
- Enable the erasure of the Main and Information memories before object code download (FET Debugger → Download → Erase main and Information memory).
- To maximize system performance during debug, disable Virtual Breakpoints (FET Debugger → Breakpoints → Use virtual breakpoints) and disable all System Breakpoints (FET Debugger → Breakpoints → System breakpoints on).

NOTE: Use Factory Settings to quickly configure a project.

Use the Factory Settings button to quickly configure a project to a usable state.

The following steps can be used to quickly configure a project. Note that the General Options tab does not have a Factory Settings button.

1. Specify the target device (General Options → Target → Device).
2. Enable an assembler project or a C or assembler project (General Options → Target → Assembler-only project).
3. Enable the generation of an executable output file (General Options → Output → Output file → Executable).
4. Accept the factory settings for the compiler (C/C++ Compiler → Factory Settings).
5. Accept the factory settings for the assembler (Assembler → Factory Settings).

6. Accept the factory settings for the linker (Linker → Factory Settings).
7. Accept the factory settings for C-SPY (Debugger → Factory Settings).
8. Debug on the hardware (Debugger → Setup → Driver → FET Debugger).
9. Specify the active parallel port used to interface to the FET if not LPT1 (FET Debugger → Setup → Connection → Texas Instruments LPT-IF) or specify the USB port (FET Debugger → Setup → Connection → Texas Instruments USB-IF).

NOTE: Avoid the use of absolute path names when referencing files.

Instead, use the relative pathname keywords $TOOLKIT_DIR$ and $PROJ_DIR$. See the IAR documentation for a description of these keywords. The use of relative path names permits projects to be moved easily, and projects do not require modification when IAR systems are upgraded (for example, from KickStart or Baseline to Full).

2.2.2 Using Math Library for MSP430 (MSPMathlib) in IAR EW430 5.60.1 and Newer

TI's MSPMathlib is part of EW430 5.60.1 and newer releases. This optimized library provides up to 26x better performance in applications that use floating point scalar math. For details, see the MSPMathlib web page (http://www.ti.com/tool/mspmathlib).

MSPMathlib may be enabled for new and existing projects on all supported devices. Enable or disable MSPMathlib in the project options (General Options → Library Configuration → MathLib).

2.2.3 Additional Project Settings for MSP430L092 and MSP430C092

The MSP430L092 can operate in two different modes: L092 mode and C092 emulation mode. The purpose of the C092 emulation mode is to behave like a C092 with up to 1920Bytes of code at its final destination for mask generation.

The operation mode is determined by EW430 before starting the debugger. Two radio buttons are available for the mode selection. By default the L092 mode is selected (see Figure 2-1 and Figure 2-2).

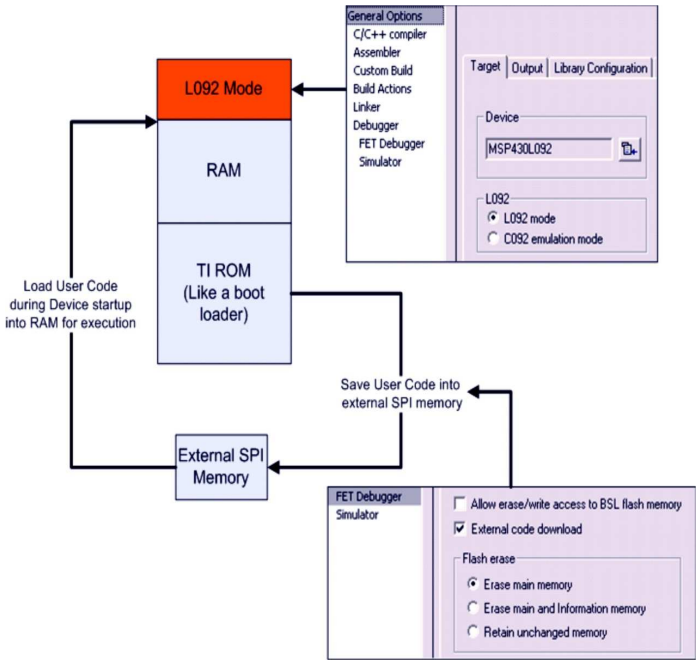

Figure 2-1. L092 Mode

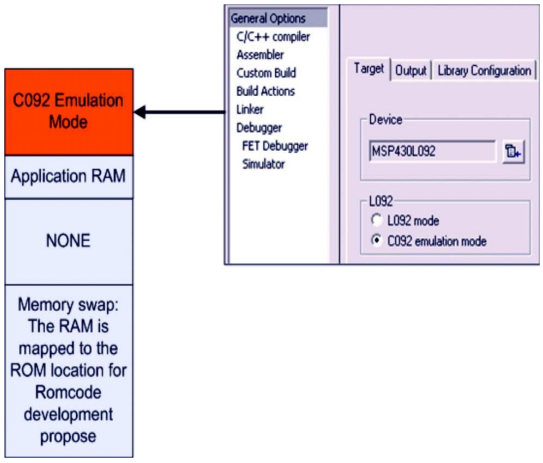

Figure 2-2. C092 Emulation Mode

2.2.3.1 MSP430L092 Loader Code

The Loader Code in the MSP430L092 is a ROM-code from TI that provides a series of services. It enables customers to build autonomous applications without needing to develop a ROM mask. Such an application consists of an MSP430 device containing the loader (for example, MSP430L092) and an SPI memory device (for example, '95512 or '25AA40); these and similar devices are available from various manufacturers.

The majority of use cases for an application with a loader device and external SPI memory for native 0.9-V supply voltage are late development, prototyping, and small series production.

Figure 2-1 shows the selection for loading the application into the external SPI memory.

2.2.3.2 Password Protection of MSP430C092

The MSP430C092 is a customer-specific ROM device that is protected by a password. To start a debug session, the password must be provided to EW430. Figure 2-3 shows how to provide a HEX password in EW430.

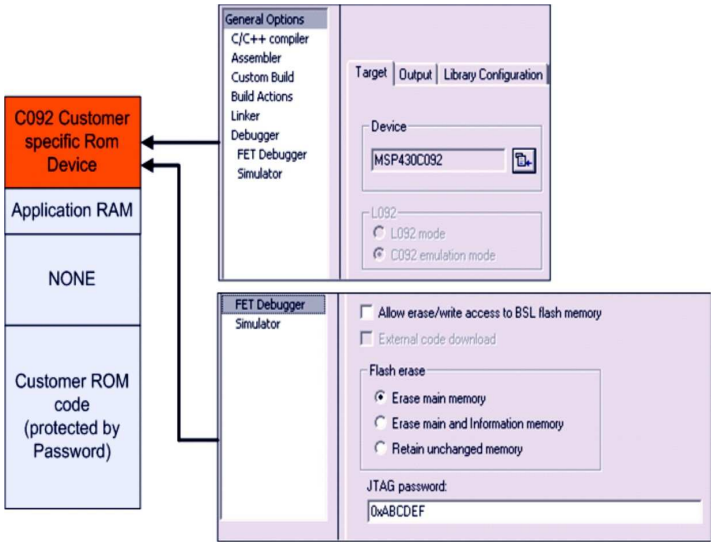

Figure 2-3. C092 Password

2.2.4 Creating a Project From Scratch

This section presents step-by-step instructions to create an assembler or C project from scratch, and to download and run the application on the MSP430 (see also Section 2.2.1, Project Settings). The *MSP430 IAR Embedded Workbench IDE User's Guide* presents a more comprehensive overview of the process.

1. Start the Workbench (Start → Programs → IAR Systems → IAR Embedded Workbench KickStart for MSP430 V3 → IAR Embedded Workbench).
2. Create a new text file (File → New → File).
3. Enter the program text into the file.

NOTE: Use .h files to simplify your code development.

KickStart is supplied with files that define the device registers and the bit names for each device. These files can greatly simplify the task of developing your program. The files are located in <Installation Root>\Embedded Workbench x.x\430\inc. Include the .h file corresponding to your target device in your text file (#include "msp430xyyy.h"). Additionally, files io430xxxx.h are provided and are optimized to be included by C source files.

4. Save the program text file (File → Save).

   It is recommended that assembler text files be saved with a file-type suffix of ".s43" and that C text files be saved with a file-type suffix of ".c".
5. Create a new workspace (File → New → Workspace).
6. Create a new project (Project → Create New Project). Select Tool chain: MSP430, Project Templates: Empty project and click OK. Specify a project name and click Save.
7. Add the program text file to the project (Project → Add Files). Select the program text file and click Open. Alternatively, double-click on the file to add it to the project.

NOTE: How to add assembler source files to your project

The default file type presented in the Add Files window is "C/C++ Files". To view assembler files (.s43), select "Assembler Files" in the "Files of type" drop-down menu.

8. Save the workspace (File → Save Workspace). Specify a workspace name and click Save.
9. Configure the project options (Project → Options). For each of the subcategories (General Options, C/C++ Compiler, Assembler, Linker, Debugger), accept the default Factory Settings with the following exceptions:
   - Specify the target device (General Options → Target → Device).
   - Enable an assembler project or a C or assembler project (General Options → Target → Assembler-only project).
   - Enable the generation of an executable output file (General Options → Output → Output file → Executable).
   - To debug on the FET (that is, the MSP430), click Debugger → Setup → Driver → FET Debugger.
   - Specify the active port used to interface to the FET (FET Debugger → Setup → Connection).
10. Build the project (Project → Rebuild All).
11. Debug the application using C-SPY (Project → Debug). This starts C-SPY, and C-SPY takes control of the target, erases the target memory, programs the target memory with the application, and resets the target.

    See FAQ Debugging #1 if C-SPY is unable to communicate with the device.
12. Click Debug → Go to start the application.
13. Click Debug → Stop Debugging to stop the application, to exit C-SPY, and to return to the Workbench.
14. Click File → Exit to exit the Workbench.

2.2.5 Additional Project Settings for LPMx.5 Debugging

2.2.5.1 What is LPMx.5

LPMx.5 is a low-power mode in which the entry and exit is handled differently than the other low-power modes.

LPMx.5, when used properly, gives the lowest power consumption available on a device. To achieve this, entry to LPMx.5 disables the LDO of the PMM module, which removes the supply voltage from the core and the JTAG module of the device. Because the supply voltage is removed from the core, all register contents and SRAM contents are lost. Exit from LPMx.5 causes a BOR event, which forces a complete reset of the system.

NOTE: The option "RELEASE JTAG ON GO" is currently not supported in the Embedded Workbench when LPMx.5 debugging is active.

See the MSP430 device family user's guide for additional LPMx.5 details.

2.2.5.2 Debug LPMx.5

To enable the LPMx.5 debug feature the “Debug LPMx.5” checkbox must be enabled by clicking FET Debugger → Setup → Debug LPMx.5 (see Figure 2-4).

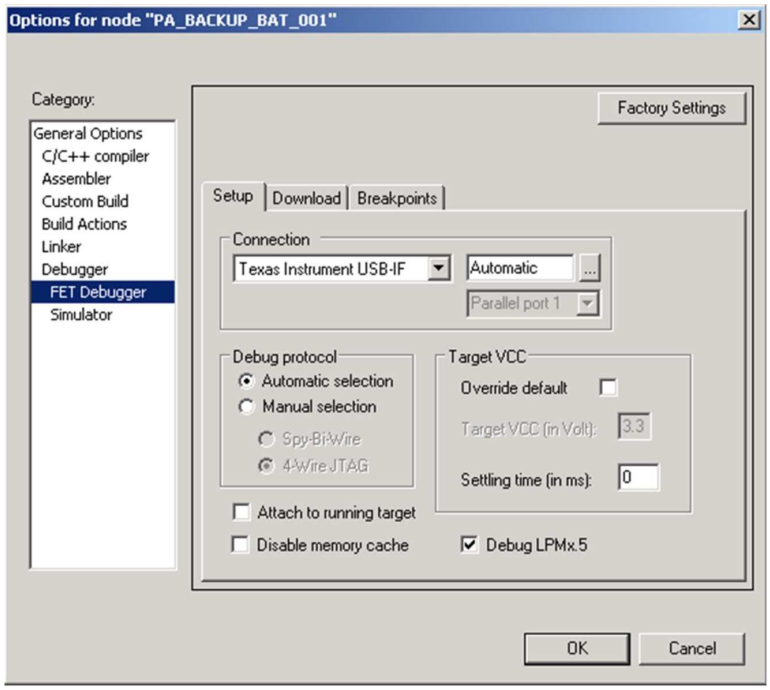

Figure 2-4. Enable LPMx.5

If the LPMx.5 debug mode is enabled a notification is displayed in the Debugger log every time the target device enters and leaves LPMx.5 mode (see Figure 2-5).

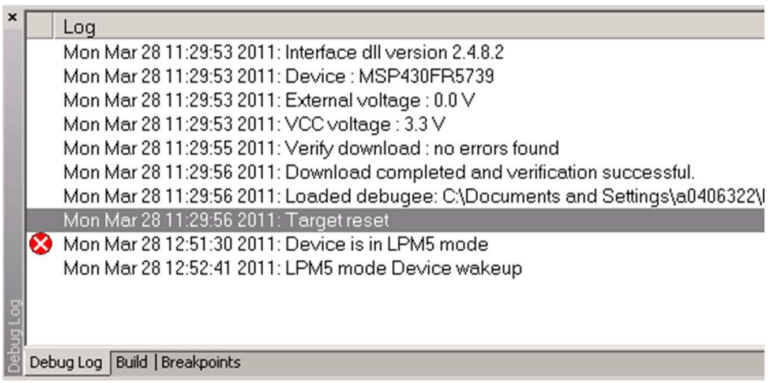

Figure 2-5. LPMx.5 Notifications

Press the Halt or Reset button in EW430 to wake up the target device from LPMx.5 and stop it at code start. All breakpoints that were active before LPMx.5 are restored and reactivated automatically.

2.2.5.3 LPMx.5 Debug Limitations

If the target device is in LPMx.5 mode, it is not possible to set or remove advanced conditional or software breakpoints. It is possible to set hardware breakpoints. In addition, only hardware breakpoints that were set during LPMx.5 can be removed in LPMx.5 mode. Attach to running target is not possible in combination with LPMx.5 mode debugging, as this results in device reset.

2.2.6 Password Protection for MSP430 Devices

When debugging an MSP430 device that supports protection by a user password, the hexadecimal JTAG password must be provided to start a debug session.

Set JTAG password by clicking FET Debugger → Download → JTAG password (see Figure 2-6).

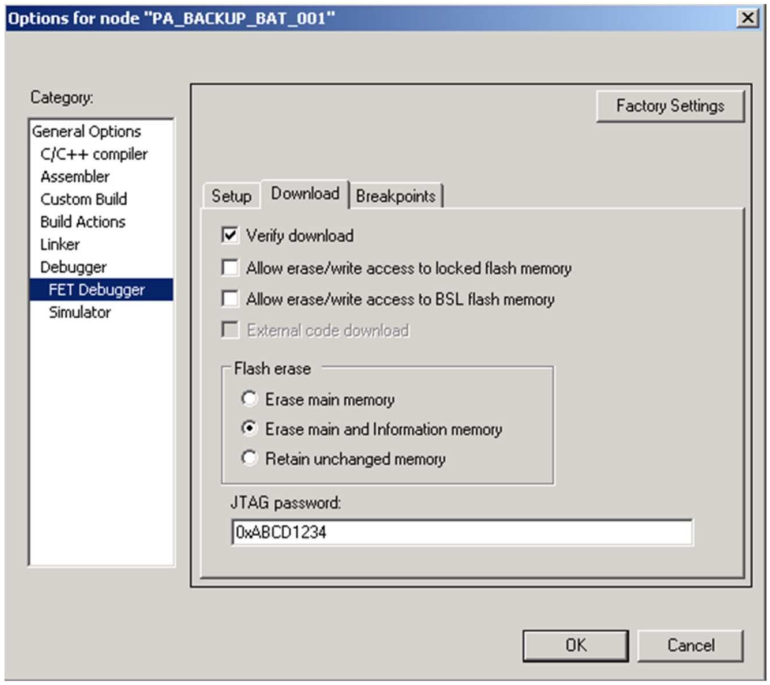

Figure 2-6. JTAG Password

2.2.7 Using an Existing IAR V1.x, V2.x, or V3.x Project

It is possible to use an existing project from an IAR V1.x, V2.x, or V3.x system with the new IAR V4.x system; see the IAR document *Step by Step Migration for EW430 x.xx*. This document is in <Installation Root>\Embedded Workbench x.x\430\doc\migration.htm.

2.2.8 Stack Management and .xcl Files

The reserved stack size can be configured through either the project options dialog (General Options → Stack/Heap) or through direct modification of the .xcl linker control files. These files are input to the linker and contain statements that control the allocation of device memory (RAM, flash). See the IAR XLINK documentation for a complete description of these files. The .xcl files provided with the FET (<Installation Root>\Embedded Workbench x.x\430\config\lnk430xxxx.xcl) define a relocatable segment (RSEG) called CSTACK. CSTACK is used to define the region of RAM that is used for the system stack within C programs. CSTACK can also be used in assembler programs (`MOV.W #SFE(CSTACK), SP`). CSTACK is defined to extend from the last location of RAM for 50 bytes (that is, the stack extends downward through RAM for 50 bytes).

Other statements in the .xcl file define other relocatable regions that are allocated from the first location of RAM to the bottom of the stack. It is critical to note that:

- **The supplied .xcl files reserve 50 bytes of RAM for the stack, whether or not this amount of stack is actually required (or if it is sufficient).**
- **There is no runtime checking of the stack. The stack can overflow the 50 reserved bytes and possibly overwrite the other segments. No error is output.**

The supplied .xcl files can be modified to tune the size of the stack to the needs of the application; edit -D_STACK_SIZE=xx to allocate xx bytes for the stack. Note that the .xcl file also reserves 50 byes for the heap if required (for example, by malloc()).

2.2.9 How to Generate Texas Instruments .TXT (and Other Format) Files

The KickStart linker can be configured to output objects in TI .TXT format for use with the GANG430 and PRGS430 programmers. Click Project → Options → Linker → Output → Format → Other → msp430-txt. Intel™ and Motorola™ formats also can be selected.

For more information, see FAQ Program Development #6 in Appendix A.

2.2.10 Overview of Example Programs

Example programs for MSP430 devices are provided in <Installation Root>\Embedded Workbench x.x\430\FET_examples. Each tool folder contains folders that contain the assembler and C sources.

<Installation Root>\Embedded Workbench\x.x\430\FET_examples\Flashing the LED.eww conveniently organizes the FET_1 demonstration code into a workspace. The workspace contains assembler and C projects of the code for each of the MSP430 device families. Debug and Release versions are provided for each of the projects.

<Installation Root>\Embedded Workbench x.x\430\FET_examples\contents.htm conveniently organizes and documents the examples.

Additional code examples can be found on the MSP430 home page under Code Examples. Note that some example programs require a 32-kHz crystal on LFXT1, and not all FETs are supplied with a 32-kHz crystal.

2.3 Using C-SPY

See Appendix B for a description of FET-specific menus within C-SPY.

2.3.1 Breakpoint Types

The C-SPY breakpoint mechanism uses a limited number of on-chip debugging resources (specifically, N breakpoint registers, see Table 2-1). When N or fewer breakpoints are set, the application runs at full device speed (or realtime). When greater than N breakpoints are set and Use Virtual Breakpoints is enabled (FET Debugger → Breakpoints → Use virtual breakpoints), the application runs under the control of the host PC; the system operates at a much slower speed but offers unlimited software breakpoints (or non-realtime). During non-realtime mode, the PC, in effect, repeatedly single steps the device and interrogates the device after each operation to determine if a breakpoint has been hit.

Both (code) address and data (value) breakpoints are supported. Data breakpoints and range breakpoints each require two MSP430 hardware breakpoints.

Table 2-1. Device Architecture, Breakpoints, and Other Emulation Features

Device	MSP430 Architecture	4-Wire JTAG	2-Wire JTAG[1]	Break-points (N)	Range Break-points	Clock Control	State Sequencer	Trace Buffer	LPMx.5 Debugging Support
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			

(1) The 2-wire JTAG debug interface is also referred to as Spy-Bi-Wire (SBW) interface. Note that this interface is supported only by the USB emulators (eZ430-xxxx and MSP-FET430UIF USB JTAG emulator) and the MSP-GANG430 production programming tool. The MSP-FET430PIF parallel port JTAG emulator does not support communication in 2-wire JTAG mode.

Table 2-1. Device Architecture, Breakpoints, and Other Emulation Features (continued)

Device	MSP430 Architecture	4-Wire JTAG	2-Wire JTAG[1]	Break-points (N)	Range Break-points	Clock Control	State Sequencer	Trace Buffer	LPMx.5 Debugging Support
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			
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 Using Breakpoints

If C-SPY is started with greater than N breakpoints set and virtual breakpoints are disabled, a message is output to inform the user that only N (realtime) breakpoints are enabled (and one or more breakpoints are disabled). Note that the workbench permits any number of breakpoints to be set, regardless of the Use Virtual Breakpoints setting of C-SPY. If virtual breakpoints are disabled, a maximum of N breakpoints can be set within C-SPY.

Resetting a program temporarily requires a breakpoint if Project → Options → Debugger → Setup → Run To is enabled (see FAQ Debugging #32).

The Run To Cursor operation temporarily requires a breakpoint. Consequently, only N – 1 breakpoints can be active when Run To Cursor is used if virtual breakpoints are disabled (see FAQ Debugging #33).

If, while processing a breakpoint, an interrupt becomes active, C-SPY stops at the first instruction of the interrupt service routine (see FAQ Debugging #26).

2.3.3 Using Single Step

When debugging an assembler file, Step Over, Step Out, and Next Statement operate like Step Into; that is, the current instruction is executed at full speed.

When debugging an assembler file, a step operation of a CALL instruction stops at the first instruction of the called function.

When debugging an assembler file, a (true) Step Over a CALL instruction that executes the called function at full device speed can be synthesized by placing a breakpoint after the CALL and using GO (to the breakpoint in realtime mode).

When debugging a C file, a single step (Step) operation executes the next C statement. Thus, it is possible to step over a function reference. If possible, a hardware breakpoint is placed after the function reference, and a GO is implicitly executed. This causes the function to be executed at full speed. If no hardware breakpoints are available, the function is executed in non-realtime mode. Step Into is supported. Step Out is supported.

Within Disassembly mode (View → Disassembly), a step operation of a non-CALL instruction executes the instruction at full device speed.

Within Disassembly mode (View → Disassembly), a step operation of a CALL instruction places, if possible, a hardware breakpoint after the CALL instruction, and then executes Go. The called function executes at full device speed. If no hardware breakpoint is available prior to the Go, the called function is executed in non-realtime mode. In either case, execution stops at the instruction following the CALL.

It is possible to single step only when source statements are present. Breakpoints must be used when running code for which there is no source code (that is, place the breakpoint after the CALL to the function for which there is no source, and then Go to the breakpoint in realtime mode).

If, during a single step operation, an interrupt becomes active, the current instruction is completed and C-SPY stops at the first instruction of the interrupt service routine (see FAQ Debugging #26).

2.3.4 Using Watch Windows

The C-SPY Watch Window mechanism permits C variables to be monitored during the debugging session. Although not originally designed to do so, the Watch Window mechanism can be extended to monitor assembler variables.

Assume that the variables to watch are defined in RAM, for example:

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

In C-SPY:

1. Open the Watch Window (View → Watch).
2. Click Debug → Quick Watch.
3. To watch varword, enter in the Expression box:
   `(__data16 unsigned int *) varword`
4. To watch varchar, enter in the Expression box:

```
(__data16 unsigned char *) varchar
```

5. Click the Add Watch button.
6. Close the Quick Watch window.
7. For the created entry in the Watch Window, click on the + symbol to display the contents (or value) of the watched variable.

To change the format of the displayed variable (default, binary, octal, decimal, hex, char), select the type, click the right mouse button, and then select the desired format. The value of the displayed variable can be changed by selecting it, and then entering the new value.

In C, variables can be watched by selecting them and then dragging and dropping them into the Watch Window.

Because the MSP430 peripherals are memory mapped, it is possible to extend the concept of watching variables to watching peripherals. Be aware that there may be side effects when peripherals are read and written by C-SPY (see FAQ Debugging #24).

CPU core registers can be specified for watching by preceding their name with '#' (that is, #PC, #SR, #SP, #R5, etc.).

Variables watched within the Watch Window are updated only when C-SPY gets control of the device (for example, following a breakpoint hit, a single step, or a stop or escape).

Although registers can be monitored in the Watch Window, View → Register is the preferred method.

Frequently Asked Questions

This appendix presents solutions to frequently asked questions regarding program hardware development and debugging tools.

Topic **Page**

A.1 Hardware

For a complete list of hardware-related FAQs, see the *MSP430 Hardware Tools User's Guide* (SLAU278).

A.2 Program Development (Assembler, C-Compiler, Linker)

1. **The files supplied in the 430\tutor folder work only with the simulator**. Do not use the files with the FET (see FAQ Program Development #11).
2. **A common MSP430 "mistake" is to fail to disable the Watchdog mechanism**; the Watchdog is enabled by default, and it resets the device if not disabled or properly handled by the application (see FAQ Program Development #14).
3. **When adding source files to a project, do not add files that are included by source files that have already been added to the project** (for example, an .h file within a .c or .s43 file). These files are added to the project file hierarchy automatically.
4. **In assembler, enclosing a string in double quotes** ("string") **automatically appends a zero byte** to the string (as an end-of-string marker). Enclosing a string in single-quotes ('string') does not.
5. When using the compiler or the assembler, **if the last character of a source line is backslash (\), the subsequent carriage return or line feed is ignored** (that is, it is as if the current line and the next line are a single line). When used in this way, the backslash character is a "line continuation" character.
6. **The linker output format must be "Debug information for C-SPY" (.d43) for use with C-SPY**. C-SPY does not start otherwise, and an error message is output. C-SPY cannot input a .TXT file.
7. **Position-independent code can be generated** using Project → Options → General Options → Target → Position-Independent Code.
8. **Within the C libraries, GIE (Global Interrupt Enable) is disabled before** (and restored after) **the hardware multiplier is used**. To disable this behavior, contact TI for the source code for these libraries.
9. **It is possible to mix assembler and C programs within the Workbench**. See the Assembler Language Interface chapter of the C/C++ Compiler Reference Guide from IAR.
10. The Workbench can produce an object file in TI .TXT format. **C-SPY cannot input an object file in TI .TXT format**. An error message is output in this case.
11. **The example programs given in the KickStart documentation (that is, Demo, Tutor, etc.) are not correct**. The programs work only in the simulator. The programs do not function correctly on an actual device, because the Watchdog mechanism is active. The programs need to be modified to disable the Watchdog mechanism. Disable the Watchdog mechanism with this C-statement:
    ```
    WDTCTL = WDTPW + WDTHOLD;
    ```
    or with this assembler statement:
    ```
    mov.w # WDTPW+WDTHOLD,&WDTCTL
    ```
12. **Access to MPY using an 8-bit operation is flagged as an error**. Within the .h files, 16-bit registers are defined in such a way that 8-bit operations upon them are flagged as an error. This feature is normally beneficial and can catch register access violations. However, in the case of MPY, it is also valid to access this register using 8-bit operators. If 8-bit operators are used to access MPY, the access violation check mechanism can be defeated by using "MPY_" to reference the register. Similarly, 16-bit operations on 8-bit registers are flagged.
13. **Constant definitions (#define) used within the .h files are effectively reserved** and include, for example, C, Z, N, and V. Do not create program variables with these names.

14. **The CSTARTUP that is implicitly linked with all C applications does not disable the Watchdog timer**. Use `WDTCL = WDTPW + WDTHOLD;` to explicitly disable the Watchdog. This statement is best placed in the __low_level_init() function that gets executed before main().

    If the Watchdog timer is not disabled, and the Watchdog triggers and resets the device during CSTARTUP, **the source screen goes blank**, as C-SPY is not able to locate the source code for CSTARTUP. Be aware that CSTARTUP can take a significant amount of time to execute if a large number of initialized global variables are used.

```
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. **Compiler optimization can remove unused variables and statements that have no effect** and can affect debugging. Optimization: NONE is supported within Project → Options → C/C++ Compiler → Code → Optimizations. Alternatively, variables can be declared volatile.
16. **The IAR tutorial assumes a Full or Baseline version of the Workbench**. Within a KickStart system, it is not possible to configure the C compiler to output assembler mnemonics.
17. Existing **projects from an IAR 1.x system can be used within the new IAR 2.x or 3.x system**; see the IAR document migration guide for EW430 x.x. This document is located in <Installation Root>\Embedded Workbench x.x\430\doc\migration.htm
18. **Assembler projects must reference the code segment** (RSEG CODE) **to use the Linker → Processing → Fill Unused Code Memory** mechanism. No special steps are required to use Linker → Processing → Fill Unused Code Memory with C projects.
19. **Ensure that the proper C runtime library is selected for C-only and mixed C and assembly language projects** (Project → General Options → Library Configuration → Library). For assembly-only projects, the runtime library must not get linked in, otherwise the build fails and a linker error is output (for example, that the RESET vector is allocated twice).
20. **Numerous C and C++ runtime libraries are provided with the Workbench**:

    cl430d: C, 64-bit doubles

    cl430dp: C, 64-bit doubles, position independent

    cl430f: C, 32-bit doubles

    cl430fp: C, 32-bit doubles, position independent

    dl430d: C++, 64-bit doubles

    dl430dp: C++, 64-bit doubles, position independent

    dl430f: C++, 32-bit doubles

    dl430fp: C++, 32-bit doubles, position independent

    See the IAR MSP430 C/C++ compiler reference guide for more information on which library to use.

A.3 Debugging (C-SPY)

1. **Debugging with C-SPY does not seem to affect an externally connected MSP430 device**. Should this be the case, check whether the main debugger menu bar contains a menu item called Simulator. If so, an actual C-SPY MSP430 core simulator session is running, and no actual communication with the target device is established. **Solution: Ensure that the C-SPY driver is set to FET Debugger** (Project → Options → Debugger → Setup → Driver).
2. **C-SPY reports that it cannot communicate with the device**. Possible solutions to this problem include:
   - Ensure that the correct debug interface is selected; use Project → Options → FET Debugger → Connection.
   - Ensure that the correct parallel port (LPT1, 2, or 3) is being specified in the C-SPY configuration in the case a parallel port MSP-FET430PIF interface is used; use Project → Options → FET Debugger → Connection → Parallel Port → LPT1 (default) or LPT2 or LPT3. Check the PC BIOS for the parallel port address (0x378, 0x278, 0x3bc), and the parallel port configuration (ECP, Compatible, Bidirectional, or Normal) (see FAQ Debugging #8). For users of IBM ThinkPad™ computers, try port specifications LPT2 and LPT3, even if the operating system reports the parallel port is located at LPT1.
   - Ensure that the jumper settings are configured correctly on the target hardware.
   - Ensure that no other software application has reserved or taken control of the COM or parallel port (for example, printer drivers, ZIP drive drivers, etc.) if a parallel port MSP-FET430PIF interface is used. Such software can prevent the C-SPY or FET driver from accessing the parallel port and, hence, communicating with the device.
   - Open the Device Manager and determine if the driver for the FET tool has been correctly installed and if the COM or parallel port is successfully recognized by the Windows OS.
   - It may be necessary to reboot the computer to complete the installation of the required port drivers.
   - Ensure that the MSP430 device is securely seated in the socket (so that the "fingers" of the socket completely engage the pins of the device), and that its pin 1 (indicated with a circular indentation on the top surface) aligns with the "1" mark on the PCB.

> **CAUTION**
>
> **Possible Damage to Device**
>
> Always handle MSP430 devices using a vacuum pick-up tool only; do not use your fingers, as they can easily bend the device pins and render the device useless. Also, always observe and follow proper ESD precautions.

3. **C-SPY reports that the device JTAG security fuse is blown**. With current MSP-FET430PIF and MSP430-FET430UIF JTAG interface tools there is a weakness when adapting target boards that are powered externally. This leads to an accidental fuse check in the MSP430 and results in the JTAG security fuse being recognized as blown although it is not. This is valid for MSP-FET430PIF and MSP-FET430UIF but is mainly seen on MSP-FET430UIF.
   Workarounds:
   - Connect the device $\overline{\text{RST}}$/NMI pin to JTAG header (pin 11), MSP-FET430PIF or MSP-FET430UIF interface tools are able to pull the $\overline{\text{RST}}$ line, this also resets the device internal fuse logic.
   - Do NOT connect both V_{CC} Tool (pin 2) and V_{CC} Target (pin 4) of the JTAG header and also specify a value for V_{CC} in the debugger that is equal to the external supply voltage.
4. **C-SPY can download data into RAM, information, and flash main memories**. A warning message is output if an attempt is made to download data outside of the device memory spaces.
5. **C-SPY can debug applications that utilize interrupts and low power modes** (see FAQ Debugging #26).
6. **C-SPY cannot access the device registers and memory while the device is running**. C-SPY displays "-" to indicate that a register or memory field is invalid. The user must stop the device to access device registers and memory. Any displayed register or memory fields are then updated.

7. **When C-SPY is started, the flash memory is erased and the opened file is programmed** in accordance with the download options as set in Project → Options → FET Debugger → Download Control. This initial erase and program operations can be disabled selecting Project → Options → FET Debugger → Download Control → Suppress Download. Programming of the flash can be initiated manually with Emulator → Init New Device.
8. **The parallel port designators (LPTx) have the following physical addresses: LPT1: 378h, LPT2: 278h, LPT3: 3BCh**. The configuration of the parallel port (ECP, Compatible, Bidirectional, Normal) is not significant; ECP seems to work well (see FAQ Debugging #1 for additional hints on solving communication problems between C-SPY and the device).
9. **C-SPY may assert $\overline{\text{RST}}$/NMI to reset the device** when C-SPY is started and when the device is programmed. The device is also reset by the C-SPY RESET button, and when the device is manually reprogrammed (Emulator → Init New Device), and when the JTAG is resynchronized (Emulator → Resynchronize JTAG). When $\overline{\text{RST}}$/NMI is not asserted (low), C-SPY sets the logic driving $\overline{\text{RST}}$/NMI to high-impedance, and $\overline{\text{RST}}$/NMI is pulled high via a resistor on the PCB.

   $\overline{\text{RST}}$/NMI may get asserted and negated after power is applied when C-SPY is started. $\overline{\text{RST}}$/NMI may then get asserted and negated a second time after device initialization is complete.

   Within C-SPY, Emulator → "Power on" Reset cycles the power to the target to generate a power-on reset.
10. **C-SPY can debug a device whose program reconfigures the function of the $\overline{\text{RST}}$/NMI pin to NMI**.
11. **The level of the XOUT/TCLK pin is undefined when C-SPY resets the device**. The logic driving XOUT/TCLK is set to high-impedance at all other times.
12. **When making current measurements of the device, ensure that the JTAG control signals are released** (Emulator → Release JTAG on Go), otherwise the device is powered by the signals on the JTAG pins and the measurements are erroneous (see FAQ Debugging #14).
13. **Most C-SPY settings** (breakpoints, etc.) **are preserved between sessions**.
14. **When C-SPY has control of the device, the CPU is ON** (that is, it is not in low-power mode) regardless of the settings of the low-power mode bits in the status register. Any low-power mode conditions are restored prior to Step or Go. Consequently, do not measure the power consumed by the device while C-SPY has control of the device. Instead, run your application using Go with JTAG released (see FAQ Debugging #12).
15. The View → Memory → Memory Fill dialog of C-SPY requires **hexadecimal values** for Starting Address, Length, and Value to be **preceded with "0x"**. Otherwise the values are interpreted as decimal.
16. The Memory debug view of C-SPY (View → Memory) can be used to view the RAM, the information memory, and the flash main memory. The Memory utility of C-SPY can be used to modify the RAM; **the information memory and flash main memory cannot be modified using the Memory utility**. The information memory and flash main memory can be programmed only when a project is opened and the data is downloaded to the device, or when Emulator → Init New Device is selected.
17. **C-SPY does not permit the individual segments of the information memory and the flash main memory to be manipulated separately**; consider the information memory to be one contiguous memory, and the flash main memory to be a second contiguous memory.
18. The Memory window correctly displays the contents of memory where it is present. However, **the Memory window incorrectly displays the contents of memory where there is none present**. Memory should be used only in the address ranges specified by the device data sheet.
19. C-SPY utilizes the system clock to control the device during debugging. Therefore, **device counters, etc., that are clocked by the Main System Clock (MCLK) are affected when C-SPY has control of the device**. Special precautions are taken to minimize the effect upon the Watchdog Timer. The CPU core registers are preserved. All other clock sources (SMCLK, ACLK) and peripherals continue to operate normally during emulation. In other words, **the Flash Emulation Tool is a partially intrusive tool**.

    Devices that support clock control (Emulator → Advanced → Clock Control) can further minimize these effects by selecting to stop the clock(s) during debugging (see FAQ Debugging #24).

20. **There is a time after C-SPY performs a reset of the device** [when the C-SPY session is first started, when the flash is reprogrammed (via Init New Device), and when JTAG is resynchronized (Resynchronize JTAG)] and before C-SPY has regained control of the device **that the device executes code normally**. This behavior may have side effects. Once C-SPY has regained control of the device, it performs a reset of the device and retains control.
21. When programming the flash, **do not set a breakpoint on the instruction immediately following the write to flash operation**. A simple workaround to this limitation is to follow the write to flash operation with a NOP, and set a breakpoint on the instruction following the NOP (see FAQ Debugging #23).
22. The **Dump Memory length specifier is restricted to four hexadecimal digits (0 to FFFF)**. This limits the number of bytes that can be written from 0 to 65535. Consequently, it is not possible to write memory from 0 to 0xFFFF inclusive, as this would require a length specifier of 65536 (or 10000h).
23. Multiple internal machine cycles are required to clear and program the flash memory. **When single stepping over instructions that manipulate the flash**, control is given back to C-SPY before these operations are complete. Consequently, **C-SPY updates its memory window with erroneous information**. A workaround to this behavior is to follow the flash access instruction with a NOP, and then step past the NOP before reviewing the effects of the flash access instruction (see FAQ Debugging #21).
24. **Peripheral bits that are cleared when read during normal program execution** (that is, interrupt flags) **are cleared when read while being debugged** (that is, memory dump, peripheral registers).

    When using certain MSP430 devices (such as MSP430F15x, MSP430F16x, MSP430F43x, and MSP430F44x devices), bits do not behave this way (that is, the bits are not cleared by C-SPY read operations).
25. **C-SPY cannot be used to debug programs that execute in the RAM of MSP430F12x and MSP430F41x devices**. A workaround to this limitation is to debug programs in flash.
26. **While single stepping with active and enabled interrupts, it can appear that only the interrupt service routine (ISR) is active** (that is, the non-ISR code never appears to execute, and the single step operation always stops on the first line of the ISR). However, this behavior is correct because the device always processes an active and enabled interrupt before processing non-ISR (that is, mainline) code. A workaround for this behavior is, while within the ISR, to disable the GIE bit on the stack so that interrupts are disabled after exiting the ISR. This permits the non-ISR code to be debugged (but without interrupts). Interrupts can later be reenabled by setting GIE in the status register in the Register window.

    On devices with the clock control emulation feature, it may be possible to suspend a clock between single steps and delay an interrupt request (Emulator → Advanced → Clock Control).
27. **The base** (decimal, hexadecimal, etc.) **property of Watch Window variables is not preserved between C-SPY sessions**; the base reverts to Default Format.
28. On devices equipped with a Data Transfer Controller (DTC), **the completion of a data transfer cycle preempts a single step of a low-power mode instruction**. The device advances beyond the low-power mode instruction only after an interrupt is processed. Until an interrupt is processed, it appears that the single step has no effect. A workaround to this situation is to set a breakpoint on the instruction following the low-power mode instruction, and then execute (Go) to this breakpoint.
29. **The transfer of data by the Data Transfer Controller (DTC) may not stop precisely when the DTC is stopped in response to a single step or a breakpoint**. When the DTC is enabled and a single step is performed, one or more bytes of data can be transferred. When the DTC is enabled and configured for two-block transfer mode, the DTC may not stop precisely on a block boundary when stopped in response to a single step or a breakpoint.
30. The C-SPY **Register window supports instruction cycle length counters**. The cycle counter is active only while single stepping. The count is reset when the device is reset, or the device is run (Go). The count can be edited (normally set to zero) at any time.
31. **It is possible to use C-SPY to get control of a running device whose state is unknown**. Simply use C-SPY to program a dummy device, and then start the application with Release JTAG on Go selected. Remove the JTAG connector from the dummy device and connect to the unknown device. Select Debug → Break (or the Stop hand) to stop the unknown device. The state of the device can then be interrogated.

32. Resetting a program temporarily requires a breakpoint if Project → Options → Debugger → Setup → Run To is enabled. If N or more breakpoints are set, Reset sets a virtual breakpoint and runs to the Run To function. Consequently, **it may require a significant amount of time before the program resets** (that is, stops at the Run To function). During this time the C-SPY indicates that the program is running, and C-SPY windows may be blank (or may not be correctly updated).
33. Run To Cursor temporarily requires a breakpoint. If N breakpoints are set and virtual breakpoints are disabled, **Run To Cursor incorrectly uses a virtual breakpoint**. This results in very slow program execution.
34. **The simulator is a CPU core simulator only**; peripherals are not simulated, and interrupts are statistical events.
35. On devices without data breakpoint capabilities, it is possible to associate with an instruction breakpoint an (arbitrarily complex) expression that C-SPY evaluates when the breakpoint is hit. **This mechanism can be used to synthesize a data breakpoint**. See the C-SPY documentation for a description of this complex breakpoint mechanism.
36. **The ROM Monitor** referenced by the C-SPY documentation applies only to older MSP430Exxx (EPROM) based devices; it **can be ignored** when using the FET and the flash-based MSP430F devices.
37. **Special function registers** (SFRs) and the peripheral registers **are displayed in View → Register**.
38. **The putchar() and getchar() breakpoints are set only if these functions are present** (and the mechanism is enabled). Note that putchar() and getchar() could be indirectly referenced by a library function.
39. **The flash program and download progress bar does not update gradually**. This behavior is to be expected. The progress bar updates whenever a "chunk" of memory is written to flash. The development tools attempt to minimize the number of program chunks to maximize programming efficiency. Consequently, it is possible, for example, for a 60K-byte program to be reduced to a single chunk, and the progress bar is not updated until the entire write operation is complete.
40. **After moving a complete EW430 project** (including workspace, project, source and generated object files) **to a different storage location** (for example, a different PC) a rebuild of the object files (rebuild project) is required before starting C-Spy. The Linker stores absolute path names in the object files, which probably do not match the new storage location. C-Spy can show a message that the source files cannot be located or can show strange artifacts during debugging.

FET-Specific Menus

Topic Page

B.1 Menus

B.1.1 Emulator → Device Information

Opens a window with information about the target device being used. Also, this window allows adjusting the target voltage in the case an MSP-FET430UIF interface is used to supply power to the target by performing a right-click inside this window. The supply voltage can be adjusted between 1.8 V and 5 V. This voltage is available on pin 2 of the 14-pin target connector to supply the target from the MSP-FET430UIF. If the target is supplied externally, the external supply voltage should be connected to pin 4 of the target connector, so the MSP-FET430UIF can set the level of the output signals accordingly.

B.1.2 Emulator → Release JTAG on Go

C-SPY uses the device JTAG signals to debug the device. On some MSP430 devices, these JTAG signals are shared with the device port pins. Normally, C-SPY maintains the pins in JTAG mode so that the device can be debugged. During this time the port functionality of the shared pins is not available.

However, when Release JTAG On Go is selected, the JTAG drivers are set to three-state, and the device is released from JTAG control (TEST pin is set to GND) when Go is activated. Any active on-chip breakpoints are retained, and the shared JTAG port pins revert to their port functions.

At this time, C-SPY has no access to the device and cannot determine if an active breakpoint (if any) has been reached. C-SPY must be manually commanded to stop the device, at which time the state of the device is determined (that is, was a breakpoint reached?) (see FAQ Debugging #12).

B.1.3 Emulator → Resynchronize JTAG

Regain control of the device.

It is not possible to Resynchronize JTAG while the device is operating.

B.1.4 Emulator → Init New Device

Initialize the device according to the settings in the Download Options. Basically, the current program file is downloaded to the device memory. The device is then reset. This option can be used to program multiple devices with the same program from within the same C-SPY session.

It is not possible to select Init New Device while the device is operating.

B.1.5 Emulator → Secure - Blow JTAG Fuse

Blows the fuse on the target device. After the fuse is blown, no communication with the device is possible.

B.1.6 Emulator → Breakpoint Usage

List all used hardware and virtual breakpoints, as well as all currently defined EEM breakpoints.

B.1.7 Emulator → Advanced → Clock Control

Disable the specified system clock while C-SPY has control of the device (following a Stop or breakpoint). All system clocks are enabled following a Go or a single step (Step or Step Into) (see FAQ Debugging #19).

B.1.8 Emulator → Advanced → Emulation Mode

Specify the device to be emulated. The device must be reset (or reinitialized through Init New Device) following a change to the emulation mode.

B.1.9 Emulator → Advanced → Memory Dump

Write the specified device memory contents to a specified file. A conventional dialog is displayed that permits the user to specify a file name, a memory starting address, and a length. The addressed memory is then written in a text format to the named file. Options permit the user to select word or byte text format, and address information and register contents also can be appended to the file.

B.1.10 Emulator → Advanced → Breakpoint Combiner

Open the Breakpoint Combiner dialog box. The Breakpoint Combiner dialog box permits one to specify breakpoint dependencies. A breakpoint is triggered when the breakpoints are encountered in the specified order.

B.1.11 Emulator → State Storage Control

Open the State Storage dialog box. The State Storage dialog box permits the user to use the state storage module. The State Storage Module is not present on all MSP430 derivatives. See Table 2-1 for implementation details

See the IAR C-SPY FET Debugger section in the MSP430 IAR Embedded Workbench IDE User Guide.

B.1.12 Emulator → State Storage Window

Open the State Storage window, and display the stored state information as configured by the State Storage dialog.

See the IAR C-SPY FET Debugger section in the MSP430 IAR Embedded Workbench IDE User Guide.

B.1.13 Emulator → Sequencer Control

Open the Sequencer dialog box. The Sequencer dialog box permits the user to configure the sequencer state machine.

See the IAR C-SPY FET Debugger section in the MSP430 IAR Embedded Workbench IDE User Guide.

B.1.14 Emulator → "Power on" Reset

Cycle power to the device to effect a reset.

B.1.15 Emulator → GIE on/off

Enables or disables all interrupts. Needs to be restored manually before Go.

B.1.16 Emulator → Leave Target Running

If C-SPY is closed, the target keeps running the user program.

B.1.17 Emulator → Force Single Stepping

On Go the program is executed by single steps. The cycle counter works correctly only in this mode.

NOTE: Availability of Emulator → Advanced menus

Not all Emulator → Advanced menus are supported by all MSP430 devices. These menus are grayed out.

Revision History

Version	Comments
SLAU138AC	Table 2-1, Added emulation features for MSP430F523x, F524x, F525x (see *MSP430F52xx*), and i2040 (see *MSP430i20xx*). Section 2.2.2, Added MSPMathlib support.
SLAU138AB	Added emulation features for MSP430TCH5E.
SLAU138AA	Added emulation features for MSP430F535x, F565x, and F645x in Table 2-1.
SLAU138Z	Added emulation features for MSP430SL5438A. Updated LPMx.5 debugging support for MSP430F665x and MSP430F67xx in Table 2-1.
SLAU138Y	Added emulation features for MSP430FR59xx and MSP430F665x. Updated LPMx.5 debugging support in Table 2-1.
SLAU138X	Related Documentation From Texas Instruments, Added CC430 documentation. Table 2-1, Added CC430F512x, CC430F514x, CC430F614x, MSP430G22xx.
SLAU138W	Updated Table 2-1.
SLAU138V	Added Section 2.2.5 and Section 2.2.6. Updated Table 2-1.
SLAU138U	Updated Table 2-1
SLAU138T	Updated Table 2-1
SLAU138S	Added Section 2.2.3. Updated Table 2-1 with MSP430x092 and MSP430F438 and MSP430F439.
SLAU138R	Added emulation features for MSP430G2xxx, MSP430F51x1, MSP430F51x2, MSP430F550x, MSP430F5510, MSP430F551x, MSP430F552x, MSP430F663x.
SLAU138Q	Updated supported operating systems in Section 1.1.
SLAU138P	Updated features for MSP430F461x and MSP430F461x 1 in Table 2-1.
SLAU138O	Removed Appendix C: 80-Pin MSP430F44x and MSP430F43x Device Emulation
SLAU138N	Added information on MSP430F5xx and CC430 devices
SLAU138M	• Removed information on hardware. It was moved into the *MSP430 Hardware Tools User's Guide* (SLAU278). • Added emulation features for MSP430F55xx and MSP430F54xxA in Table 2-1. • Updated and extended Table 2-1 with architecture information.
SLAU138L	• Added MSP-FET430U100A kit in Section 1.7 and MSP-TS430PZ100A target socket module schematic (Figure B-19) and PCB (Figure B-20). • Added emulation features for CC430F513x, CC430F612x, CC430F613x, MSP430F41x2, MSP430F47x, MSP430FG479, and MSP430F471xx in Table 2-1. • Updated MSP-TS430PN80 target socket module schematic (Figure B-15) with information on MSP430F47x and MSP430FG47x. • Removed information throughout on MSP-FET430Pxx0 and MSP-FET430X110 kits.
SLAU138K	• Added MSP-FET430U5x100 kit and MSP-TS430PZ5x100 target socket module schematic. • Added overview of debug interfaces as Table 1-1. • Added eZ430-F2013, T2012, and eZ430-RF2500.
SLAU138J	• Updated Section 1-7.
SLAU138I	• Added MSP-TS430PW28 Target Socket Module, Schematic (Figure B-5) and PCB (Figure B-6). • Updated MSP-FET430U28 kit content information (DW or PW package support) (Section 1-7). • Added Emulation features for MSP430F21x2 to Table 2-1. • Updated MSP-TS430PW14 Target Socket Module, Schematic (Figure B-1). • Updated MSP-TS430DA38 Target Socket Module, Schematic (Figure B-7).
SLAU138H	• Updated Appendix E
SLAU138G	• Added Emulation features for MSP430F22x2, MSP430F241x, MSP430F261x, MSP430FG42x0 and MSP430F43x1 to Table 2-1. • Updated Figure B-15 and Figure B-16 in Appendix B.

Version	Comments
SLAU138F	• Renamed MSP-FET430U40 to MSP-FET430U23x0. • Replaced MSP-FET430U40 schematic and PCB figures with renamed MSP-FET430U23x0 images. • Added FAQ Hardware #2 in Section A.1. • Added FAQ Debugging #3 in Section A.3.

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

EVALUATION BOARD/KIT/MODULE (EVM) ADDITIONAL TERMS

Texas Instruments (TI) provides the enclosed Evaluation Board/Kit/Module (EVM) under the following conditions:

The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies TI from all claims arising from the handling or use of the goods.

Should this evaluation board/kit not meet the specifications indicated in the User's Guide, the board/kit may be returned within 30 days from the date of delivery for a full refund. THE FOREGOING LIMITED WARRANTY IS THE EXCLUSIVE WARRANTY MADE BY SELLER TO BUYER AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. EXCEPT TO THE EXTENT OF THE INDEMNITY SET FORTH ABOVE, NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

Please read the User's Guide and, specifically, the Warnings and Restrictions notice in the User's Guide prior to handling the product. This notice contains important safety information about temperatures and voltages. For additional information on TI's environmental and/or safety programs, please visit www.ti.com/esh or contact TI.

No license is granted under any patent right or other intellectual property right of TI covering or relating to any machine, process, or combination in which such TI products or services might be or are used. TI currently deals with a variety of customers for products, and therefore our arrangement with the user is not exclusive. TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein.

REGULATORY COMPLIANCE INFORMATION

As noted in the EVM User's Guide and/or EVM itself, this EVM and/or accompanying hardware may or may not be subject to the Federal Communications Commission (FCC) and Industry Canada (IC) rules.

For EVMs **not** subject to the above rules, this evaluation board/kit/module is intended for use for ENGINEERING DEVELOPMENT, DEMONSTRATION OR EVALUATION PURPOSES ONLY and is not considered by TI to be a finished end product fit for general consumer use. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to part 15 of FCC or ICES-003 rules, which are designed to provide reasonable protection against radio frequency interference. Operation of the equipment may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

General Statement for EVMs including a radio

User Power/Frequency Use Obligations: This radio is intended for development/professional use only in legally allocated frequency and power limits. Any use of radio frequencies and/or power availability of this EVM and its development application(s) must comply with local laws governing radio spectrum allocation and power limits for this evaluation module. It is the user's sole responsibility to only operate this radio in legally acceptable frequency space and within legally mandated power limitations. Any exceptions to this are strictly prohibited and unauthorized by Texas Instruments unless user has obtained appropriate experimental/development licenses from local regulatory authorities, which is responsibility of user including its acceptable authorization.

For EVMs annotated as FCC – FEDERAL COMMUNICATIONS COMMISSION Part 15 Compliant

Caution

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

FCC Interference Statement for Class A EVM devices

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

FCC Interference Statement for Class B EVM devices

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

For EVMs annotated as IC – INDUSTRY CANADA Compliant

This Class A or B digital apparatus complies with Canadian ICES-003.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Concerning EVMs including radio transmitters

This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Concerning EVMs including detachable antennas

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (e.i.r.p.) is not more than that necessary for successful communication.

This radio transmitter has been approved by Industry Canada to operate with the antenna types listed in the user guide with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Cet appareil numérique de la classe A ou B est conforme à la norme NMB-003 du Canada.

Les changements ou les modifications pas expressément approuvés par la partie responsable de la conformité ont pu vider l'autorité de l'utilisateur pour actionner l'équipement.

Concernant les EVMs avec appareils radio

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Concernant les EVMs avec antennes détachables

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire à l'établissement d'une communication satisfaisante.

Le présent émetteur radio a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés dans le manuel d'usage et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur.

【Important Notice for Users of EVMs for RF Products in Japan】

This development kit is NOT certified as Confirming to Technical Regulations of Radio Law of Japan

If you use this product in Japan, you are required by Radio Law of Japan to follow the instructions below with respect to this product:

1. Use this product in a shielded room or any other test facility as defined in the notification #173 issued by Ministry of Internal Affairs and Communications on March 28, 2006, based on Sub-section 1.1 of Article 6 of the Ministry's Rule for Enforcement of Radio Law of Japan,
2. Use this product only after you obtained the license of Test Radio Station as provided in Radio Law of Japan with respect to this product, or
3. Use of this product only after you obtained the Technical Regulations Conformity Certification as provided in Radio Law of Japan with respect to this product. Also, please do not transfer this product, unless you give the same notice above to the transferee. Please note that if you could not follow the instructions above, you will be subject to penalties of Radio Law of Japan.

Texas Instruments Japan Limited
(address) 24-1, Nishi-Shinjuku 6 chome, Shinjuku-ku, Tokyo, Japan

http://www.tij.co.jp

【無線電波を送信する製品の開発キットをお使いになる際の注意事項】

本開発キットは技術基準適合証明を受けておりません。

本製品のご使用に際しては、電波法遵守のため、以下のいずれかの措置を取っていただく必要がありますのでご注意ください。

1. 電波法施行規則第6条第1項第1号に基づく平成18年3月28日総務省告示第173号で定められた電波暗室等の試験設備でご使用いただく。
2. 実験局の免許を取得後ご使用いただく。
3. 技術基準適合証明を取得後ご使用いただく。

なお、本製品は、上記の「ご使用にあたっての注意」を譲渡先、移転先に通知しない限り、譲渡、移転できないものとします。

上記を遵守頂けない場合は、電波法の罰則が適用される可能性があることをご留意ください。

日本テキサス・インスツルメンツ株式会社
東京都新宿区西新宿６丁目２４番１号
西新宿三井ビル
http://www.tij.co.jp

EVALUATION BOARD/KIT/MODULE (EVM)
WARNINGS, RESTRICTIONS AND DISCLAIMERS

For Feasibility Evaluation Only, in Laboratory/Development Environments. Unless otherwise indicated, this EVM is not a finished electrical equipment and not intended for consumer use. It is intended solely for use for preliminary feasibility evaluation in laboratory/development environments by technically qualified electronics experts who are familiar with the dangers and application risks associated with handling electrical mechanical components, systems and subsystems. It should not be used as all or part of a finished end product.

Your Sole Responsibility and Risk. You acknowledge, represent and agree that:

1. You have unique knowledge concerning Federal, State and local regulatory requirements (including but not limited to Food and Drug Administration regulations, if applicable) which relate to your products and which relate to your use (and/or that of your employees, affiliates, contractors or designees) of the EVM for evaluation, testing and other purposes.
2. You have full and exclusive responsibility to assure the safety and compliance of your products with all such laws and other applicable regulatory requirements, and also to assure the safety of any activities to be conducted by you and/or your employees, affiliates, contractors or designees, using the EVM. Further, you are responsible to assure that any interfaces (electronic and/or mechanical) between the EVM and any human body are designed with suitable isolation and means to safely limit accessible leakage currents to minimize the risk of electrical shock hazard.
3. Since the EVM is not a completed product, it may not meet all applicable regulatory and safety compliance standards (such as UL, CSA, VDE, CE, RoHS and WEEE) which may normally be associated with similar items. You assume full responsibility to determine and/or assure compliance with any such standards and related certifications as may be applicable. You will employ reasonable safeguards to ensure that your use of the EVM will not result in any property damage, injury or death, even if the EVM should fail to perform as described or expected.
4. You will take care of proper disposal and recycling of the EVM’s electronic components and packing materials.

Certain Instructions. It is important to operate this EVM within TI’s recommended specifications and environmental considerations per the user guidelines. Exceeding the specified EVM ratings (including but not limited to input and output voltage, current, power, and environmental ranges) may cause property damage, personal injury or death. If there are questions concerning these ratings please contact a TI field representative prior to connecting interface electronics including input power and intended loads. Any loads applied outside of the specified output range may result in unintended and/or inaccurate operation and/or possible permanent damage to the EVM and/or interface electronics. Please consult the EVM User's Guide prior to connecting any load to the EVM output. If there is uncertainty as to the load specification, please contact a TI field representative. During normal operation, some circuit components may have case temperatures greater than 60°C as long as the input and output are maintained at a normal ambient operating temperature. These components include but are not limited to linear regulators, switching transistors, pass transistors, and current sense resistors which can be identified using the EVM schematic located in the EVM User's Guide. When placing measurement probes near these devices during normal operation, please be aware that these devices may be very warm to the touch. As with all electronic evaluation tools, only qualified personnel knowledgeable in electronic measurement and diagnostics normally found in development environments should use these EVMs.

Agreement to Defend, Indemnify and Hold Harmless. You agree to defend, indemnify and hold TI, its licensors and their representatives harmless from and against any and all claims, damages, losses, expenses, costs and liabilities (collectively, "Claims") arising out of or in connection with any use of the EVM that is not in accordance with the terms of the agreement. This obligation shall apply whether Claims arise under law of tort or contract or any other legal theory, and even if the EVM fails to perform as described or expected.

Safety-Critical or Life-Critical Applications. If you intend to evaluate the components for possible use in safety critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, such as devices which are classified as FDA Class III or similar classification, then you must specifically notify TI of such intent and enter into a separate Assurance and Indemnity Agreement.

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

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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