MSP-FET430 Flash Emulation Tool (FET) (For Use With IAR Embedded Workbench Version 3.x)

User's Guide

Literature Number: SLAU138F
June 2004–Revised March 2007

TEXAS INSTRUMENTS

Contents

List of Figures

List of Tables

Read This First

About This Manual

This manual documents the Texas Instruments MSP-FET430 Flash Emulation Tool (FET). The FET is the development tool for the MSP430 ultralow-power microcontroller. Both available interfaces, the parallel port interface and the USB interface, are described here.

How to Use This Manual

Read and follow the instructions in Chapter 1, *Get Started Now!*. This chapter lists the expected contents of the FET, provides instructions on installing the hardware and software, and shows how to run the demonstration programs. After you see how quick and easy it is to use the FET, TI recommends that you read all of this manual.

This manual describes the setup and operation of the FET, but it does not fully describe the MSP430 or the development software systems. For details of these items, refer to the appropriate TI and IAR™ documents listed in Section 1.12, *Important MSP430 Documents on the CD-ROM and Web*.

This manual applies to the following tools (and devices):

- MSP-FET430PIF (debug interface with parallel port connection, for all MSP430 flash-based devices)
- MSP-FET430UIF (debug interface with USB connection, for all MSP430 flash-based devices)

The following tools contain the parallel port debug interface (MSP-FET430PIF) and the respective target socket module:

- MSP-FET430X110 (for the MSP430F11xIDW, MSP430F11x1AIDW, and MSP430F11x2IDW devices)
- MSP-FET430P120 (for the MSP430F12xIDW and MSP430F12x2IDW devices)
- MSP-FET430P140 (for the MSP430F13xIPM, MSP430F14xIPM, MSP430F15xIPM, MSP430F16xIPM, and MSP430F161xIPM devices)
- MSP-FET430P410 (for the MSP430F41xIPM devices)
- MSP-FET430P430 (for the MSP430F43xIPN devices)
- MSP-FET430P440 (for the MSP430F43xIPZ and MSP430F44xIPZ devices)

The following tools contain the USB debug interface (MSP-FET430UIF) and the respective target-socket module:

- MSP-FET430U14 (for MSP430 devices in 14-pin PW packages)
- MSP-FET430U28 (for MSP430 devices in 20- and 28-in DW packages)
- MSP-FET430U38 (for MSP430 devices in 38-pin DA packages)
- MSP-FET430U23x0 (for MSP430F2330/F2350/F2370 devices in 40-pin RHA packages only)
- MSP-FET430U48 (for MSP430 devices in 48-pin DL package)
- MSP-FET430U64 (for MSP430 devices in 64-pin PM package)
- MSP-FET430U80 (for MSP430 devices in 80-pin PN package)
- MSP-FET430U100 (for MSP430 devices in 100-pin PZ package)

These tools contains the most up-to-date materials available at the time of packaging. For the latest materials (data sheets, user's guides, software, application information, etc.), 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

MSP430xxxx device data sheets

MSP430x1xx Family User's Guide, SLAU049

MSP430x2xx Family User's Guide, SLAU144

MSP430x3xx Family User's Guide, SLAU012

MSP430x4xx Family User's Guide, SLAU056

If You Need Assistance

Support for the MSP430 device and the FET 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. Additional device-specific information can be found on the MSP430 web site at www.ti.com/msp430.

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. Please consult the extensive documentation provided with KickStart before requesting assistance.

FCC Warning

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio-frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio-frequency interference. Operation of this equipment in other environments 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.

Get Started Now!

This chapter enables you to inventory your FET, and then it instructs you how to install the software and hardware, and run the demonstration programs.

Topic **Page**

1.1 Kit Contents, MSP-FET430X110

- One READ ME FIRST document
- One MSP430 CD-ROM
- One MSP-FET430X110 Flash Emulation Tool. This is the PCB on which is mounted a 20-pin ZIF socket for the MSP430F11xIDW, MSP430F11x1AIDW, or MSP430F11x2IDW device. A 25-conductor cable originates from the FET for connecting to the PC parallel port.
- One small box containing two MSP430F1121AIDW device samples

1.2 Kit Contents, MSP-FET430PIF

- One READ ME FIRST document
- One MSP430 CD-ROM
- One MSP-FET430PIF interface module
- One 25-conductor cable
- One 14-conductor cable

1.3 Kit Contents, MSP-FET430Pxx0 ('P120, 'P140, 'P410, 'P430, 'P440)

- One READ ME FIRST document
- One MSP430 CD-ROM
- One MSP-FET430PIF FET Interface module. This is the unit that has a 25-pin male D-Sub connector on one end of the case, and a 2x7 pin male connector on the other end of the case.
- One target socket module

  MSP-FET430P120: One MSP-TS430DW28 target socket module. This is the PCB on which is mounted a 28-pin ZIF socket for the MSP430F12xIDW or MSP43012x2IDW device. A 2×7-pin male connector is also present on the PCB.

  MSP-FET430P140: One MSP-TS430PM64 target socket module. This is the PCB on which is mounted a 64-pin clam-shell-style socket for the MSP430F13xIPM, MSP430F14xIPM, MSP430F15xIPM, MSP430F16xIPM, or MSP430F161xIPM device. A 2×7-pin male connector is also present on the PCB.

  MSP-FET430P410: One MSP-TS430PM64 target socket module. This is the PCB on which is mounted a 64-pin clam-shell-style socket for the MSP430F41xIPM device. A 2×7-pin male connector is also present on the PCB.

  MSP-FET430P430: One MSP-TS430PN80 target socket module. This is the PCB on which is mounted an 80-pin ZIF socket for the MSP430F43xIPN device. A 2×7-pin male connector is also present on the PCB.

  MSP-FET430P440: One MSP-TS430PZ100 target socket module. This is the PCB on which is mounted a 100-pin ZIF socket for the MSP430F43xIPZ or MSP430F44xIPZ device. A 2×7-pin male connector is also present on the PCB.
- One 25-conductor cable
- One 14-conductor cable
- Four or eight headers

  MSP-FET430P120: Four PCB 1×14-pin headers (two male and two female)

  MSP-FET430P140: Eight PCB 1×16-pin headers (four male and four female)

  MSP-FET430P410: Eight PCB 1×16-pin headers (four male and four female)

  MSP-FET430P430: Eight PCB 1×20-pin headers (four male and four female)

  MSP-FET430P440: Eight PCB 1×25-pin headers (four male and four female)

- One small box containing two or four MSP430 device samples
  MSP-FET430P120: MSP430F123IDW and/or MSP430F1232IDW
  MSP-FET430P140: MSP430F149IPM and/or MSP430F169IPM
  MSP-FET430P410: MSP430F413IPM
  MSP-FET430P430: MSP430F437IPN and/or MSP430FG439
  MSP-FET430P440: MSP430F449IPZ
  Consult the device data sheets for device specifications. Device errata can be found in the respective device product folder on the web provided as a PDF document. Depending on the device, errata may also be found in the device bug database at www.ti.com/sc/cgi-bin/buglist.cgi.

1.4 Kit Contents, MSP-FET430UIF

- One READ ME FIRST document
- One MSP430 CD-ROM
- One MSP-FET430UIF interface module
- One USB cable
- One 14-conductor cable

1.5 Kit Contents, MSP-FET430Uxx ('U14, 'U28, 'U38, 'U23x0, 'U48, 'U64, 'U80, 'U100)

- One READ ME FIRST document
- One MSP430 CD-ROM
- One MSP-FETP430UIF USB interface module. This is the unit that has a USB B-connector on one end of the case, and a 2×7-pin male connector on the other end of the case.
- One target socket module
  MSP-FET430U14: One MSP-TS430PW14 target socket module. This is the PCB on which is mounted a 14-pin ZIF socket. It fits all MSP430 devices in 14-pin PW packages. A 2×7-pin male connector is also present on the PCB.
  MSP-FET430U28: One MSP-TS430DW28 target socket module. This is the PCB on which is mounted a 28-pin ZIF socket. It fits all MSP430 devices in 20- and 28-pin DW packages. A 2×7-pin male connector is also present on the PCB.
  MSP-FET430U38: One MSP-TS430DA38 target socket module. This is the PCB on which is mounted a 38-pin ZIF socket. It fits all MSP430 devices in 38-pin DA packages. A 2×7-pin male connector is also present on the PCB.
  MSP-FET430U23x0: One MSP-TS430QFN23x0 (former name MSP-TS430QFN40) target socket module. This is the PCB on which is mounted a 40-pin ZIF socket. It fits only MSP430F2330/F2350/F2370 devices in 40-pin RHA package. A 2×7-pin male connector is also present on the PCB.
  MSP-FET430U48: One MSP-TS430DL48 target socket module. This is the PCB on which is mounted a 48-pin ZIF socket. It fits all MSP430 devices in 48-pin DL package. A 2×7-pin male connector is also present on the PCB.
  MSP-FET430U64: One MSP-TS430PM64 target socket module. This is the PCB on which is mounted a 64-pin ZIF socket. It fits all MSP430 devices in 64-pin PM package. A 2×7-pin male connector is also present on the PCB.
  MSP-FET430U80: One MSP-TS430PN80 target socket module. This is the PCB on which is mounted a 80-pin ZIF socket. It fits all MSP430 devices in 80-pin PN package. A 2×7-pin male connector is also present on the PCB.
  MSP-FET430U100: One MSP-TS430PZ100 target socket module. This is the PCB on which is mounted a 100-pin ZIF socket. It fits all MSP430 devices in 100-pin PZ package. A 2×7-pin male connector is also present on the PCB.
- One USB cable
- One 14-conductor cable

- Four or eight headers
  MSP-FET430U14: Four PCB 1×7-pin headers (two male and two female)
  MSP-FET430U28: Four PCB 1×14-pin headers (two male and two female)
  MSP-FET430U38: Four PCB 1×19-pin headers (two male and two female)
  MSP-FET430U23x0: Eight PCB 1×10-pin headers (four male and four female)
  MSP-FET430U48: Four PCB 2×24-pin headers (two male and two female)
  MSP-FET430U64: Eight PCB 1x16-pin headers (four male and four female)
  MSP-FET430U80: Eight PCB 1x20-pin headers (four male and four female)
  MSP-FET430U100: Eight PCB 1x25-pin headers (four male and four female)
- One small box containing two or four MSP430 device samples
  MSP-FET430U14: MSP430F2013IPW
  MSP-FET430U28: MSP430F123IDW and/or MSP430F1232IDW
  MSP-FET430U38: MSP430F2274IDA
  MSP-FET430U23x0: MSP430F2370IRHA
  MSP-FET430U48: MSP430F4270IDL
  MSP-FET430U64: MSP430F417IPM and MSP430F169IPM
  MSP-FET430U80: MSP430FG439IPN
  MSP-FET430U100: MSP430F449IPZ and MSP430FG4619IPZ
  Consult the device data sheets for device specifications. Device errata can be found in the respective device product folder on the web provided as a PDF document. Depending on the device, errata may also be found in the device bug database at www.ti.com/sc/cgi-bin/buglist.cgi.

1.6 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® 98, Windows 2000, Windows ME, Windows NT 4.0, and Windows XP. However, the USB FET interface works only with Windows 2000 and Windows XP.

1.7 Hardware Installation, MSP-FET430X110

1. Connect the 25-conductor cable originating from the FET to the parallel port of your PC. The driver for accessing the PC parallel port is installed during IAR Embedded Workbench installation. Note that a restart is required after the IAR Embedded Workbench installation for the driver to become active.
2. Ensure that the MSP430F1121AIDW is securely seated in the socket and that its pin 1 (indicated with a circular indentation on the top surface) aligns with the "1" mark on the PCB.
3. Ensure that jumpers J1 (near the non-socketed IC on the FET) and J5 (near the LED) are in place. Schematics of the FET and its parts are presented in Appendix B.

1.8 Hardware Installation, MSP-FET430PIF

1. Use the 25-conductor cable to connect the FET interface module to the parallel port of your PC. The necessary driver for accessing the PC parallel port will be installed automatically during IAR Embedded Workbench installation. Note that a restart is required after the IAR Embedded Workbench installation for the driver to become active.
2. Use the 14-conductor cable to connect the parallel port debug interface module to a target board, such as an MSP-TS430xxx target socket module.

1.9 Hardware Installation, MSP-FET430UIF

1. Use the USB cable to connect the USB FET interface module to a USB port of your PC. The USB FET should be recognized instantly, as the USB device driver should have been installed already with the KickStart software. **If for any reason the Install Wizard starts, respond to the prompts and, when prompted, browse to the driver files that are located in <Installation Root>\Embedded Workbench x.x\430\bin\WinXP. Detailed driver installation instructions can be found in Appendix E**.
2. After connecting to a PC, the USB FET performs a selftest during which the red LED flashes for about 2 seconds. If the selftest passed successfully, the green LED lights permanently.
3. Use the 14-conductor cable to connect the USB FET interface module to a target board, such as an MSP-TS430xxx target socket module.
4. Ensure that the MSP430 device is securely seated in the socket and that its pin 1 (indicated with a circular indentation on the top surface) aligns with the "1" mark on the PCB.
5. Compared to the parallel port debug interface, the USB FET has additional features like: JTAG security fuse blow and adjustable target V_{CC} (1.8 V to 3.6 V); target can be supplied with up to 100 mA.

1.10 Hardware Installation, MSP-FET430Uxx ('U14, 'U28, 'U38, 'U23x0, 'U48, 'U64, 'U80, 'U100), MSP-FET430Pxx0 ('P120, 'P140, 'P410, 'P430, 'P440)

1. Connect the MSP-FET430PIF or MSP-FET430UIF debug interface to the appropriate port of your PC. Use the 14-conductor cable to connect the FET Interface module to the supplied target socket module.
2. Ensure that the MSP430 device is securely seated in the socket, and that its pin 1 (indicated with a circular indentation on the top surface) aligns with the "1" mark on the PCB.
3. Ensure that the two jumpers (LED and VCC) near the 2×7-pin male connector are in place. Schematics of the target socket module and its parts are presented in Appendix B.

Note: Regarding 'U38, see FAQ Hardware #2.

1.11 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 will open.
3. Click on the tab at the bottom of the workspace window that corresponds to your MSP430 device (MSP430xxxx) and desired language (assembler or C).
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 will erase the device Flash and then download 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've just built and tested your first MSP430 application!

1.12 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) contain the most recent version of these documents.

Documents describing the IAR tools (Workbench/C-SPY, the assembler, the C compiler, the linker, and the librarian) are located in the common\doc and 430\doc folders. The documents are in PDF format. Supplements to the documents (i.e., 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 and/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 (i.e., an actual MSP430 device), or with a software simulator of the device. KickStart is used to refer 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/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 development environment is function limited. The following restrictions are in place:

- The C compiler does not generate an assembly code list file.
- The linker links a maximum of 4K bytes of code originating from C source (but an unlimited amount of code originating from assembler source).
- The simulator inputs a maximum of 4K bytes of code.

A full (i.e., unrestricted) version of the software tools can be purchased from IAR. A mid-featured tool set – called Baseline, with a 12K-byte C-code size limitation and basic floating-point operations – is also available from IAR. Consult 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. Please 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/required:

- Specify the target device (General Options → Target → Device).
- Enable an assembler project or a C/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/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 (i.e., 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/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 pathnames 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 pathnames permits projects to be moved easily, and projects will not require modification when IAR systems are upgraded (e.g., from KickStart or Baseline to Full).

2.2.2 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 for each device that define the device registers and the bit names, and 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/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 (i.e., 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.

2.2.3 Using an Existing IAR V1.x/V2.x Project

It is possible to use an existing project from an IAR V1.x/V2.x system with the new IAR V3.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.

2.2.4 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 (i.e., the stack extends downwards 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, regardless if 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()).

2.2.5 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 can also be selected.

For more information, see FAQ Program Development #6.

2.2.6 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: 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 C for a description of FET-specific menus within C-SPY.

2.3.1 Breakpoint Types

The C-SPY breakpoint mechanism makes use of 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. Number of Device Breakpoints and Other Emulation Features

Device	4-Wire JTAG	2-Wire Spy-Bi-Wire	Breakpoints (N)	Range Breakpoints	Clock Control	State Sequencer	Trace Buffer
MSP430F11x1	X		2				
MSP430F11x2	X		2				
MSP430F12x	X		2				
MSP430F12x2	X		2				
MSP430F13x	X		3	X			
MSP430F14x	X		3	X			
MSP430F15x	X		8	X	X	X	X
MSP430F16x	X		8	X	X	X	X
MSP430F161x	X		8	X	X	X	X
MSP430F20xx	X	X	2		X		
MSP430F21x1	X		2		X		
MSP430F22x4	X	X	2		X		
MSP430F23x0	X		2		X		
MSP430F41x	X		2		X		
MSP430F42x	X		2		X		
MSP430F42x0	X		2		X		
MSP430F43x	X		8	X	X	X	X
MSP430F44x	X		8	X	X	X	X
MSP430FE42x	X		2		X		
MSP430FG43x	X		2		X		
MSP430FG461x	X		8	X	X	X	X
MSP430FW42x	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.

Reseting 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; i.e., 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 GOing (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 only possible to single step when source statements are present. Breakpoints must be used when running code for which there is no source code (i.e., 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.

Since 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 '#' (i.e., #PC, #SR, #SP, #R5, etc.).

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

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

Design Considerations for In-Circuit Programming

This chapter presents signal requirements for in-circuit programming of the MSP430.

Topic **Page**

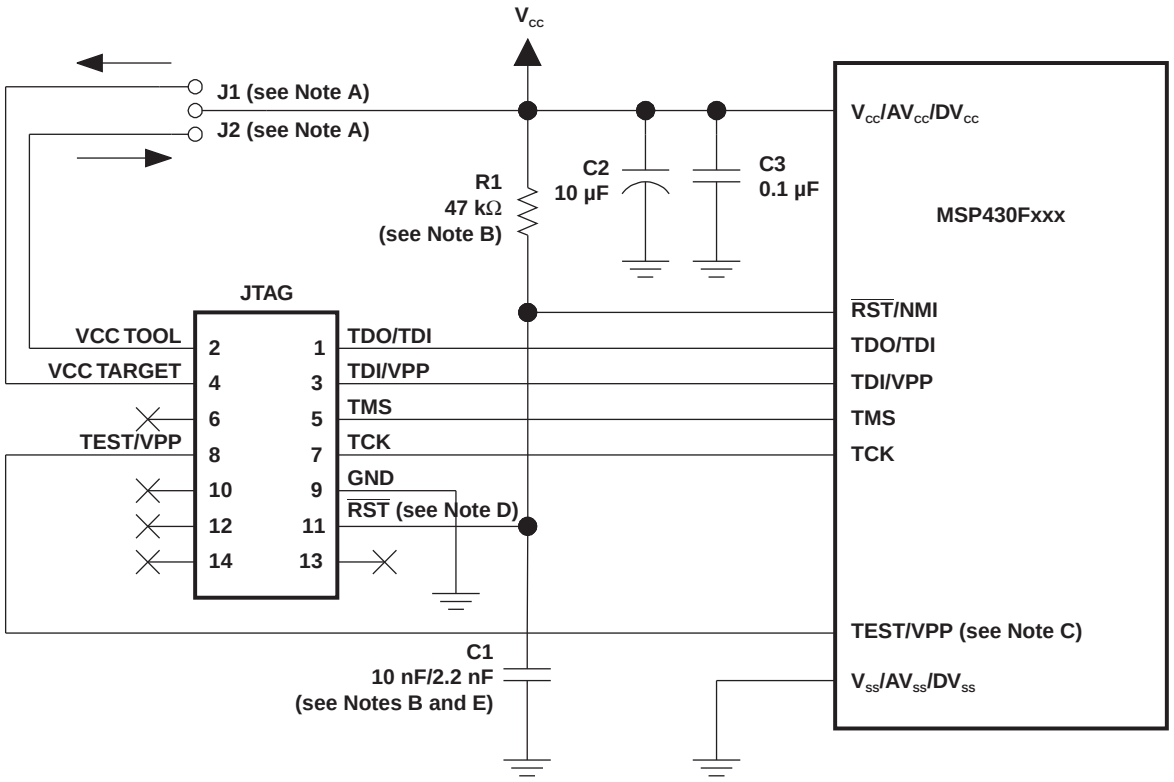
3.1 Signal Connections for In-System Programming and Debugging, MSP-FET430PIF, MSP-FET430UIF, GANG430, PRGS430

With the proper connections, the C-SPY debugger and an FET hardware JTAG interface, such as the MSP-FET430PIF and MSP-FET430UIF, can be used to program and debug code on a target board. In addition, the connections can also support the GANG430 or PRGS430 production programmers, which provide an easy way to program prototype boards, if desired.

Figure 3-1 shows the connections between the 14-pin FET Interface module connector and the target device required to support in-system programming and debugging using C-SPY for 4-wire JTAG communication. Figure 3-2 shows the connections for 2-wire Spy-Bi-Wire communication. While 4-wire JTAG mode is generally supported on all MSP430 devices, 2-wire Spy-Bi-Wire mode is available on selected devices only. See Table 2-1 for information on which interfacing method can be used on which device.

The connections for the FET Interface module and the GANG430 or PRGS430 are identical. Both the FET interface module and GANG430 can supply V_{CC} to your target board (via pin 2). In addition, the FET interface module and GANG430 have a V_{CC}-sense feature that, if used, requires an alternate connection (pin 4 instead of pin 2). The V_{CC}-sense feature senses the local V_{CC} (present on the target board, i.e., a battery or other local power supply) and adjusts the output signals accordingly. If the target board is to be powered by a local V_{CC}, the connection to pin 4 on the JTAG should be made and not the connection to pin 2. This utilizes the V_{CC}-sense feature and prevents any contention that might occur if the local on-board V_{CC} were connected to the V_{CC} supplied from the FET interface module or the GANG430. If the V_{CC}-sense feature is not necessary (i.e., the target board is to be powered from the FET Interface module or the GANG430) the V_{CC} connection is made to pin 2 on the JTAG header and no connection is made to pin 4. Figure 3-1 and Figure 3-2 show a jumper block that supports both scenarios of supplying V_{CC} to the target board. If this flexibility is not required, the desired V_{CC} connections may be hard-wired, eliminating the jumper block. Pins 2 and 4 must not be connected simultaneously.

Note that in 4-wire JTAG communication mode (see Figure 3-1), the connection of the target $\overline{\text{RST}}$ signal to the JTAG connector is optional, and it is not required when using 4-wire JTAG communication mode capable only devices. However, when using 2-wire Spy-Bi-Wire communication mode capable devices in 4-wire JTAG mode, the $\overline{\text{RST}}$ connection must be made. The MSP430 development tools and device programmers perform a target reset by issuing a JTAG command to gain control over the device. However, if this is unsuccessful, the $\overline{\text{RST}}$ signal of the JTAG connector can be used by the development tool or device programmer as an additional way to assert a device reset.



A Make either connection J1 (if a local target power supply is used) or connection J2 (if powering the from the debug/programming adapter).

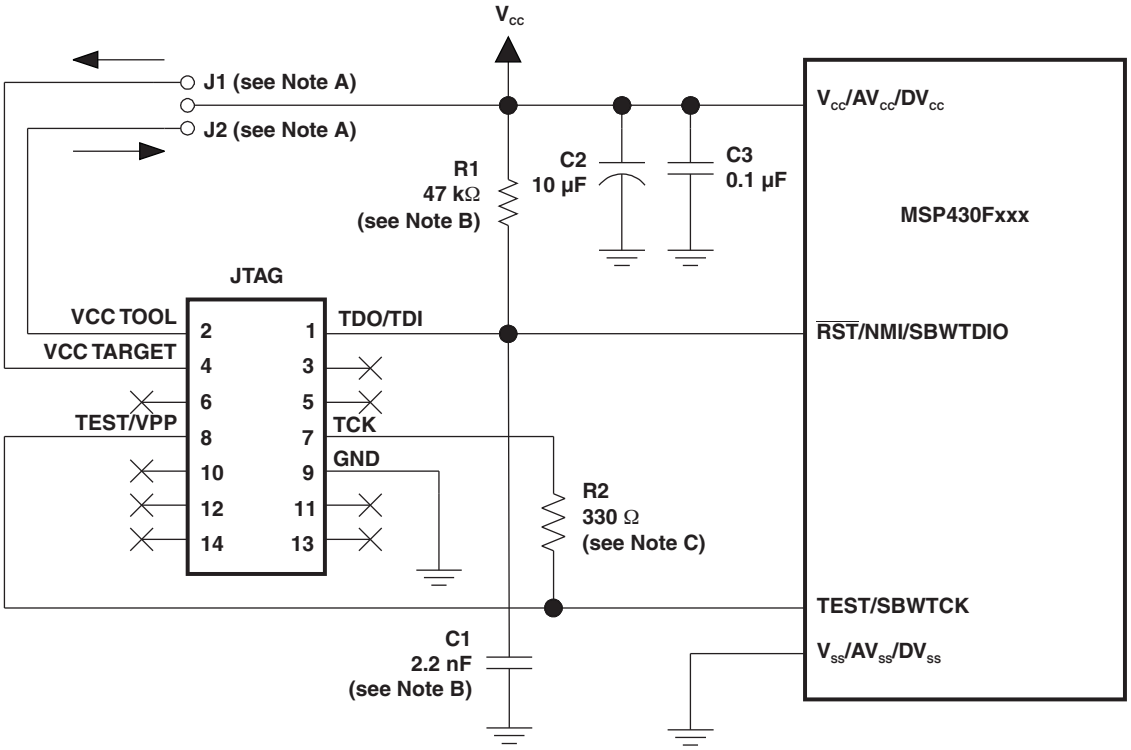
B The $\overline{\text{RST}}$/NMI pin R1/C1 configuration is device-family dependent. See the respective MSP430 family user's guide for the recommended configuration.

C The TEST/VPP pin is only available on MSP430 family members with multiplexed JTAG pins. See the device data sheet to determine if this pin is available.

D The connection to the JTAG connector $\overline{\text{RST}}$ pin is optional when using 4-wire JTAG communication mode capable-only devices and is not required for device programming or debugging. However, this connection is required when using 2-wire Spy-Bi-Wire communication mode capable devices in 4-wire JTAG mode.

E When using 2-wire Spy-Bi-Wire communication capable devices in 4-wire JTAG mode, the upper limit for C1 should not exceed 2.2 nF. This applies to both TI FET interface modules (LPT/USB FET).

Figure 3-1. Signal Connections for 4-Wire JTAG Communication



A Make either connection J1 (if a local target power supply is used) or connection J2 (if powering the from the debug/programming adapter).

B Note that the device RST/NMI/SBWTDIO pin is used in 2-wire Spy-Bi-Wire mode for bi-directional debug communication with the device and that any capacitance attached to this signal may affect the ability to establish a connection with the device. The upper limit for C1 is 2.2 nF when using current TI FET Interface modules (USB FET).

C R2 is used to protect the JTAG debug interface TCK signal against the JTAG security fuse blow voltage that is supplied by the TEST/VPP pin during the fuse blow process. In the case that fuse blow functionality is not needed, R2 is not required (becomes 0 Ω) and the connection TEST/VPP must not be made.

Figure 3-2. Signal Connections for 2-Wire Spy-Bi-Wire Communication

3.2 External Power

The PC parallel port can source only a limited amount of current. Because of the ultralow-power capability of the MSP430, a standalone FET does not exceed the available current. However, if additional circuitry is added to the tool, this current limit could be exceeded. In this case, external power can be supplied to the tool via connections provided on the MSP-FET430X110 and the target socket modules. See the schematics of the MSP-FET430X110 and the target socket modules in Appendix B to locate the external power connectors.

The MSP-FET430UIF can supply targets with up to 100 mA through pin 2 of the 14-pin connector. V_{CC} for the target can be selected between 1.8 V and 3.6 V in steps of 0.1 V. Alternatively, the target can be supplied externally. In this case, the external voltage should be connected to pin 4 of the 14-pin connector. The MSP-FET430UIF then automatically adjusts the level of the JTAG signals to external V_{CC}. Only pin 2 (if MSP-FET430UIF supplies target) or pin 4 (if target is externally supplied) must be connected, not both at the same time.

When an MSP-FET430X110 is powered from an external supply, an on-board device regulates the external voltage to the level required by the MSP430.

When a target socket module is powered from an external supply, the external supply powers the device on the target socket module and any user circuitry connected to the target socket module, and the FET interface module continues to be powered from the PC via the parallel port. If the externally supplied voltage differs from that of the FET interface module, the target socket module must be modified so that the externally supplied voltage is routed to the FET interface module (so that it may adjust its output voltage levels accordingly). See the target socket module schematics in Appendix B.

3.3 Bootstrap Loader

The JTAG pins provide access to the Flash memory of the MSP430Fxxx devices. On some devices, these pins are shared with the device port pins, and this sharing of pins can complicate a design (or it may simply not be possible to do so). As an alternative to using the JTAG pins, most MSP430Fxxx devices contain a program (a "Bootstrap Loader") that permits the Flash memory to be erased and programmed simply, using a reduced set of signals. Application reports SLAA089 and SLAA096 fully describe this interface. TI does not produce a BSL tool. However, customers can easily develop their own BSL tools using the information in the application reports, or BSL tools can be purchased from third parties. See the MSP430 web site for the application reports and a list of MSP430 third-party tool developers.

TI suggests that MSP430Fxxx customers design their circuits with the BSL in mind (i.e., TI suggests providing access to these signals via, for example, a header).

See FAQ Hardware #11 for a second alternative to sharing the JTAG and port pins.

The BSL tool requires the following device signals:

- $\overline{\text{RST}}$/NMI
- TEST(1) or TCK
- GND
- VCC
- P1.1
- P2.2 or P1.0(2)

(1) If present on device

(2) '1xx and '2xx devices use pins P1.1 and P2.2 for the BSL. '4xx devices use pins P1.0 and P1.1 for the BSL.

Frequently Asked Questions

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

Topic **Page**

A.1 Hardware

1. **The state of the device** (CPU registers, RAM memory, etc.) **is undefined following a reset**. Exceptions to the above statement are that the PC is loaded with the word at 0xFFFE (i.e., the reset vector), the status register is cleared, and the peripheral registers (SFRs) are initialized as documented in the device-family user's guides. C-SPY resets the device after programming it.
2. **MSP430F22xx Target Socket Module (MSP-TS430DA38) – Important Information**
   Due to the large capacitive coupling introduced by the device socket between the adjacent signals XIN/P2.6 (socket pin 6) and $\overline{\text{RST}}$/SBWTDIO (socket pin 7), in-system debugging can disturb the LFXT1 low-frequency crystal oscillator operation (ACLK). This behavior only applies to the Spy-Bi-Wire (2-wire) JTAG configuration and only to the period while a debug session is active.
   Workarounds:
   - Use the 4-wire JTAG mode debug configuration instead of the Spy-Bi-Wire (2-wire) JTAG configuration. This can be achieved by placing jumpers JP4 through JP9 accordingly.
   - Use the debugger option “Release JTAG On Go” that can be selected from the IDE drop down menu. This will prevent the debugger from accessing the MSP430 while the application is running. Note that in this mode a manual halt is required to see if a breakpoint was hit. Refer to the IDE documentation for more information on this feature.
   - Use an external clock source to drive XIN directly.
3. With current interface hardware and software, there is a weakness when adapting target boards that are powered externally. This leads to an accidental fuse check in the MSP430. This is valid for PIF and UIF but is mainly seen on UIF. A solution is being developed.
   Workarounds:
   - Connect $\overline{\text{RST}}$/NMI pin to JTAG header (pin 11), LPT/USB tools are able to pull the RST line, which alsos reset the device internal fuse logic.
   - Use the debugger option "Release JTAG On Go" that can be selected from the IDE dropdown menu. This prevents the debugger from accessing the MSP430 while the application is running. Note that in this mode, a manual halt is required to see if a breakpoint was hit. See the IDE documentation for more information on this feature.
   - Use an external clock source to drive XIN directly.
4. When the MSP-FET430X110 is used as an interface to an MSP430 on the user's circuit (i.e., there is no MSP430 device in the FET socket), **the XOUT and XIN signals from the FET should not be connected to the corresponding pins of the in-circuit MSP430**. Similarly, when using the interface module, do not connect the XOUT and XIN signals from the interface module to the corresponding pins of the in-circuit MSP430.
5. The 14-conductor **cable** connecting the FET interface module and the target socket module **must not exceed 8 inches (20 centimeters) in length**.
6. The signal assignment on the **14-conductor cable is identical** for the **parallel port interface** and the **USB FET**.
7. **To utilize the on-chip ADC voltage references, C6** (10 µF, 6.3 V, low leakage) **must be installed** on the target socket module.
8. **Crystals/resonators Q1 and Q2** (if applicable) **are not provided** on the target socket module. For MSP430 devices that contain user-selectable loading capacitors, the effective capacitance is the selected capacitance plus 3 pF (pad capacitance) divided by two.
9. **Crystals/resonators have no effect on the operation of the tool and C-SPY** (as any required clocking/timing is derived from the internal DCO/FLL).
10. **On 20-pin and 28-pin devices** with multiplexed port/JTAG pins (P1.4 to P1.7), **it is required that "Release JTAG On Go" be selected to use these pins in their port capacity**. See Section C.1.2 for additional information regarding this mechanism.
11. **As an alternative to sharing the JTAG and port pins** (on 20 and 28 pin devices), **consider using an MSP430 device that is a "superset" of the smaller device**. A very powerful feature of the MSP430 is that the family members are code and architecturally compatible, so code developed on one device (for example, one without shared JTAG and port pins) ports effortlessly to another (assuming an equivalent set of peripherals).

12. **Information memory may not be blank** (erased to 0xFF) when the device is delivered from TI. Customers should erase the information memory before its first usage. Main memory of packaged devices is blank when the device is delivered from TI.
13. **The device current increases by approximately 10 μA when a device in low-power mode is stopped** (using ESC), **and then the low-power mode is restored** (using Go). This behavior appears to happen on all devices except the MSP430F12x.
14. The following **ZIF sockets** are used in the FET tools and target socket modules:
    - 14-pin device (PW package): ENPLAS OTS-14-065-01
    - 20-pin device (PW package): Yamaichi IC189-0202-64
    - 28-pin device (DW package): Wells-CTI 652 D028
    - 38-pin device (DA package): Yamaichi IC189-0382-037
    - 40-pin device (RHA package): Enplas QFN-40B-0.5-01
    - 48-pin device (DL package): Yamaichi IC51-0482-1163
    - 64-pin device (PM package): Yamaichi IC51-0644-807
    - 80-pin device (PN package): Yamaichi IC201-0804-014
    - 100-pin device (PZ package): Yamaichi IC201-1004-008

    ENPLAS: www.enplas.com

    Wells-CTI: www.wellscti.com/

    Yamaichi: www.yamaichi.us/
15. **Supply current measurement on target socket modules**. On each module a jumper connects V_{CC} with VCC430. If this jumper is removed and an ampere meter is connected to the jumper pins, the supply current of the module can be measured. As the pullup resistor (47 kΩ) on the reset line is connected to V_{CC}, the MSP430 device sees a marginal voltage at pin $\overline{RST}$/NMI if V_{CC} is present and the jumper is open. Therefore, V_{CC} should be applied after the ampere meter has been connected.

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/line feed is ignored** (i.e., 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 (i.e., Demo, Tutor, etc.) are not correct**. The programs work only in the simulator. However, 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 `WDT = 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/or statements that have no effect** and can effect 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/3.x system**; refer to 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/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 (e.g., 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 no other software application has reserved/taken control of the 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/FET driver from accessing the parallel port and, hence, communicating with the device.
   - 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/MSP-FET430UIF interface tools are able to pull the $\overline{\text{RST}}$ line, this will also reset the device internal fuse logic.
   - Do NOT connect both Vcc Tool (pin 2) and Vcc Target (pin 4) of the JTAG header and specify a value for Vcc in the debugger which 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 will display "-" to indicate that a register/memory field is invalid. The user must stop the device to access device registers and memory. Any displayed register/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 will be powered by the signals on the JTAG pins and the measurements will be erroneous (see FAQ Debugging #14 and Hardware #13).
13. **Most C-SPY settings** (breakpoints, etc.) **are preserved between sessions**.
14. **When C-SPY has control of the device, the CPU is ON** (i.e., 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 will be 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 and Hardware #13).
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 only be programmed 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) will be 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 will execute code normally**. This behavior may have side effects. Once C-SPY has regained control of the device, it will perform a reset of the device and retain 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** (i.e., interrupt flags) **are cleared when read while being debugged** (i.e., memory dump, peripheral registers).

    When using certain MSP430 devices (such as MSP430F15x/16x and MSP430F43x/44x devices), bits do not behave this way (i.e., 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 'F12x and 'F41x devices**. A work around 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** (i.e., 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 alwayses process an active and enabled interrupt before processing non-ISR (i.e., 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 appear 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 only active 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. Reseting 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** (i.e., 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's 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()/getchar() breakpoints are set only if these functions are present** (and the mechanism is enabled). Note that putchar()/getchar() could be indirectly referenced by a library function.
39. **The Flash program/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 updated until the entire write operation is complete.

Hardware

This appendix contains information relating to the FET hardware, including schematics and PCB pictorials.

Topic **Page**

B.1 Schematics and PCBs

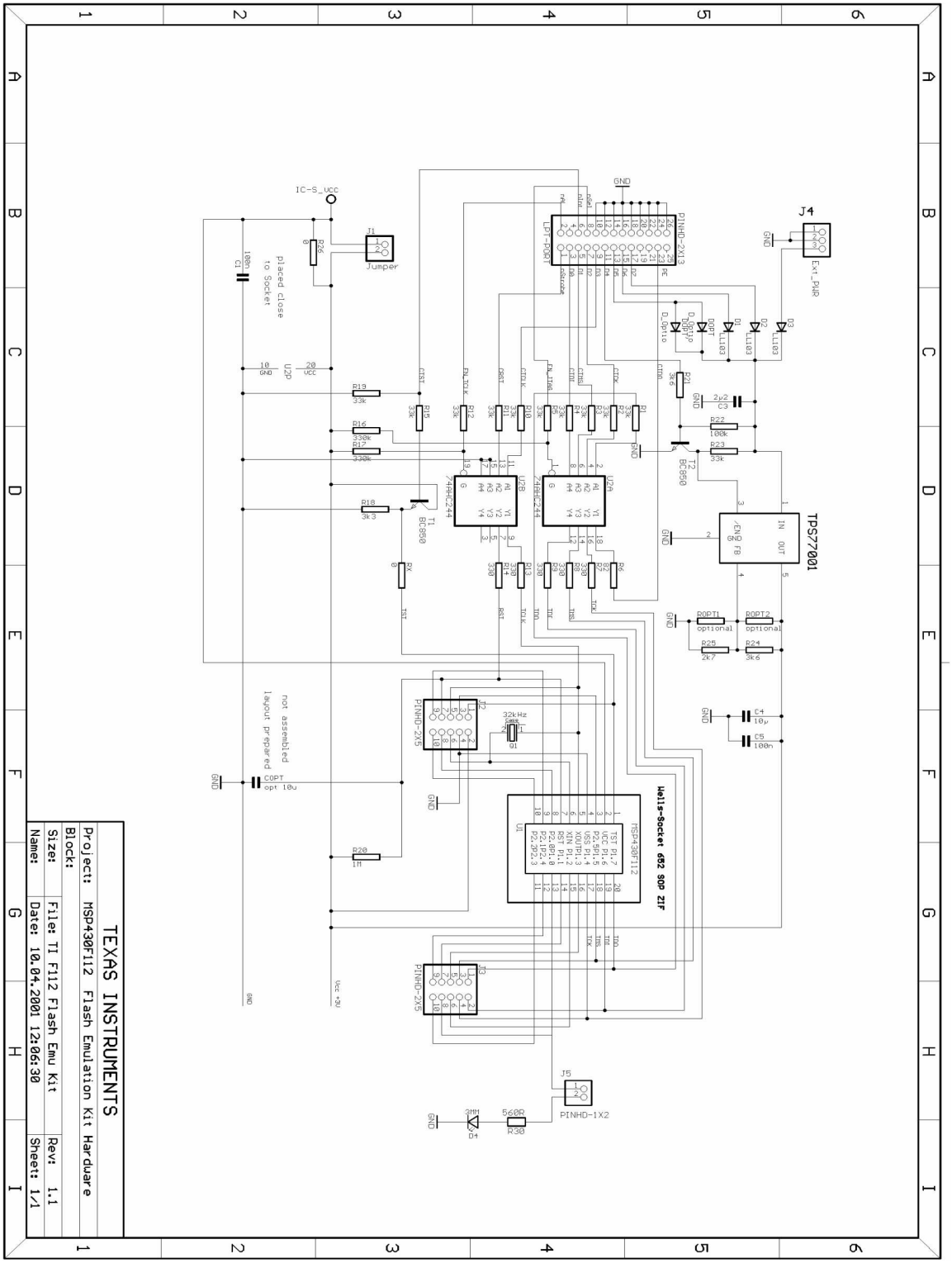
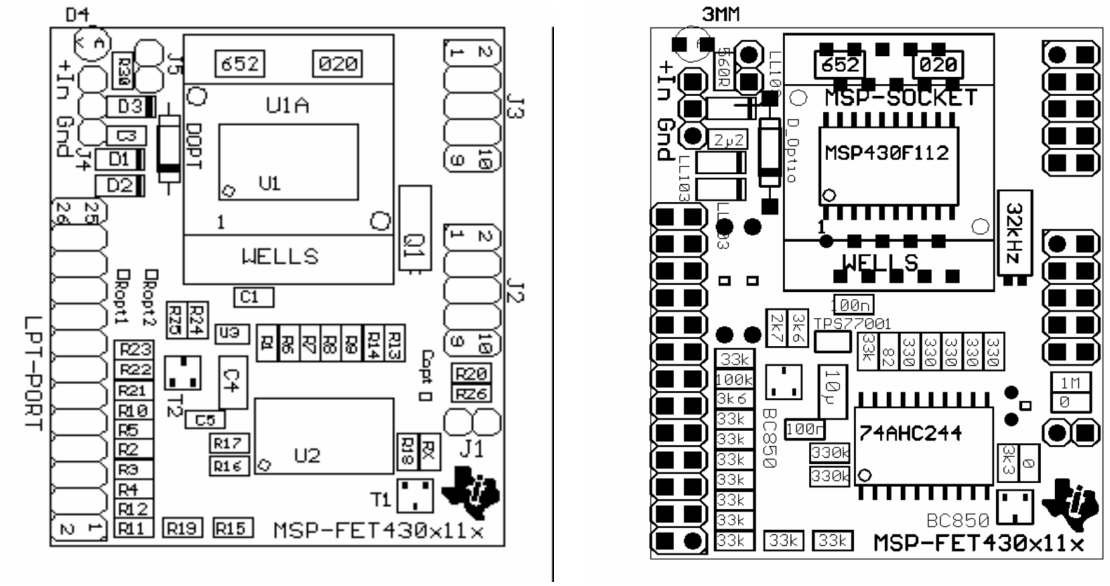
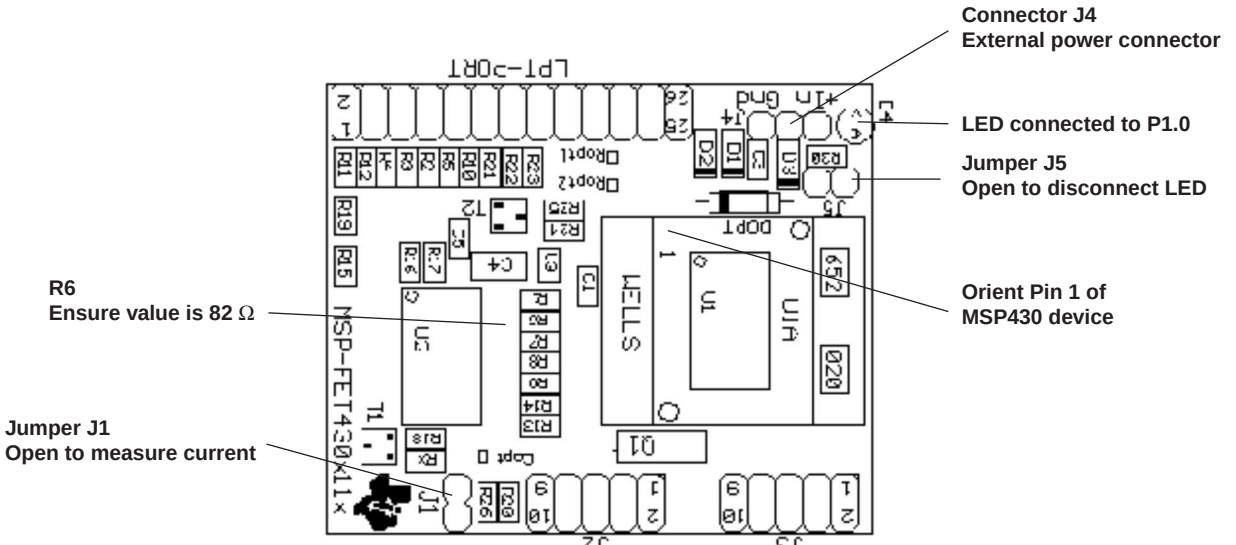


Figure B-1. MSP-FET430X110, Schematic





	J2			
P2.1	RST	XOUT	P2.5	TST
P2.2	P2.0	XIN	Vss	Vcc

J3				
P2.4	P1.1	P1.3	P1.5	P1.7
P2.3	P1.0	P1.2	P1.4	P1.6

Figure B-2. MSP-FET430X110, PCB

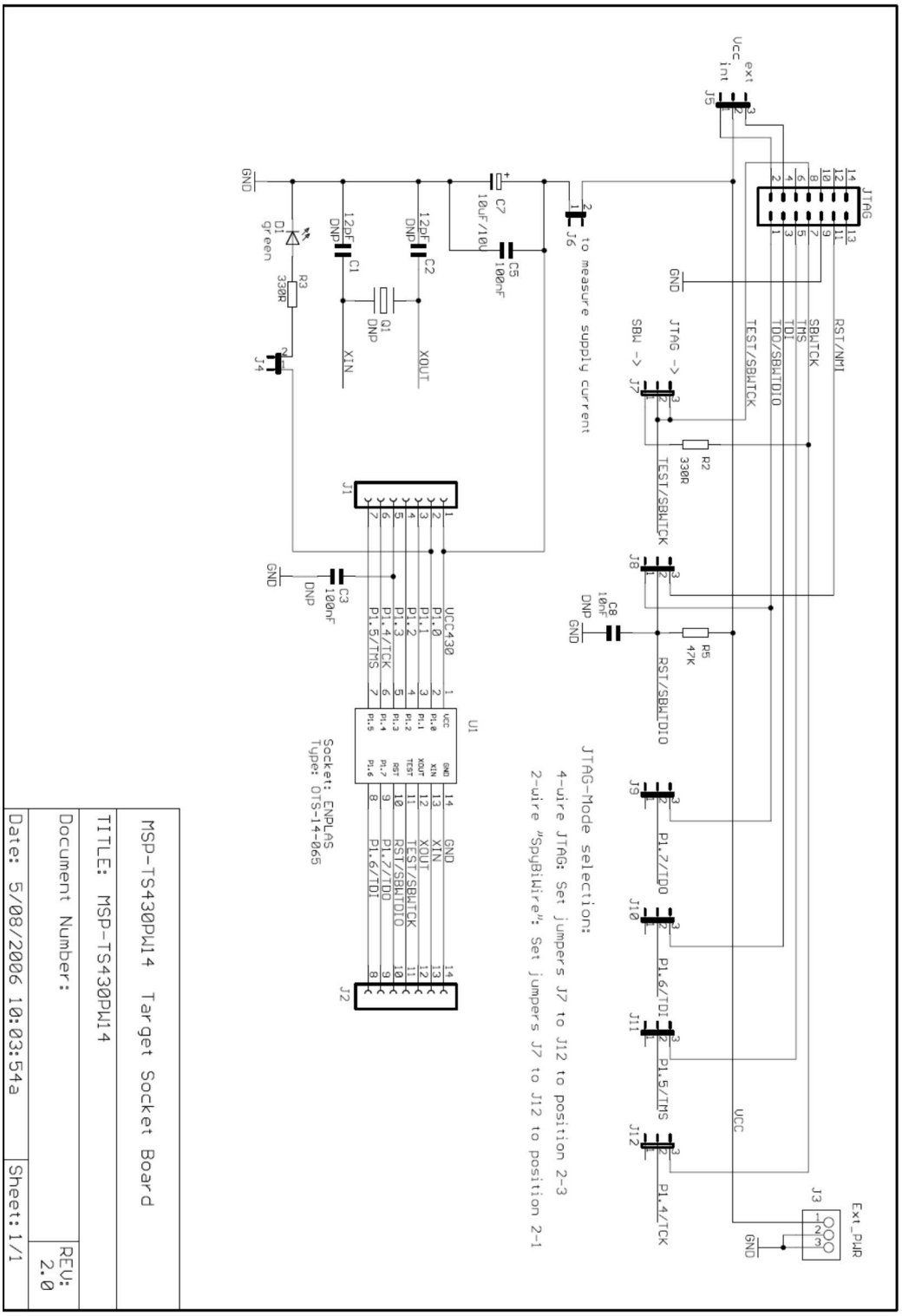


Figure B-3. MSP-TS430PW14 Target Socket Module, Schematic

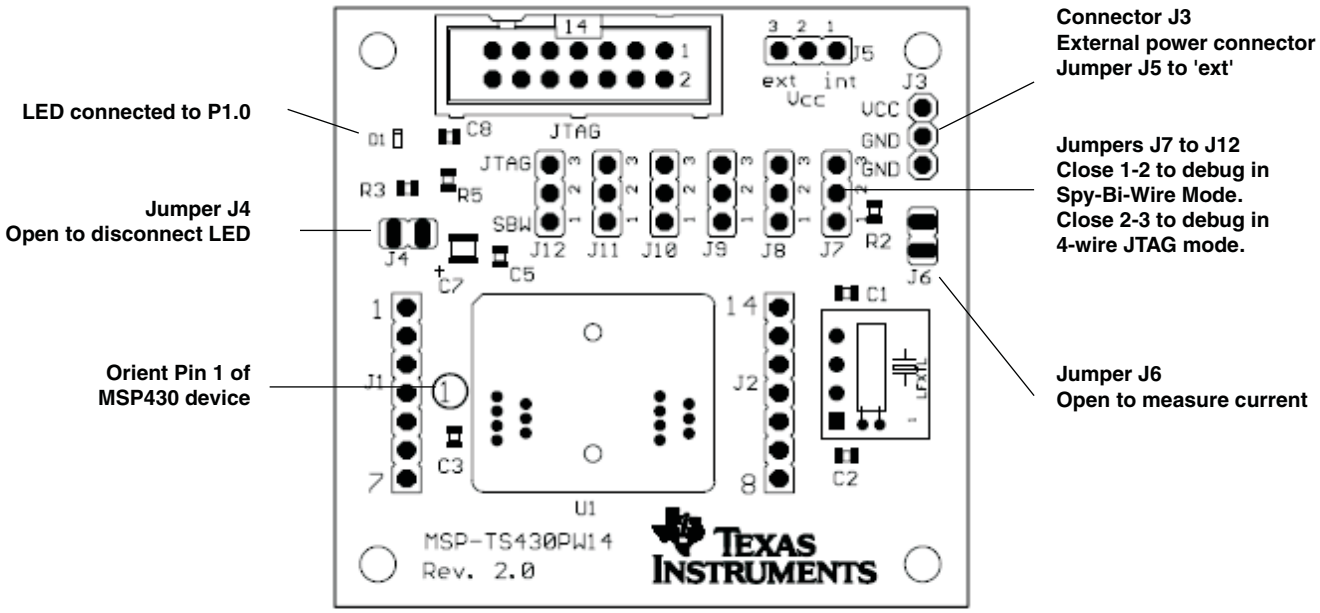
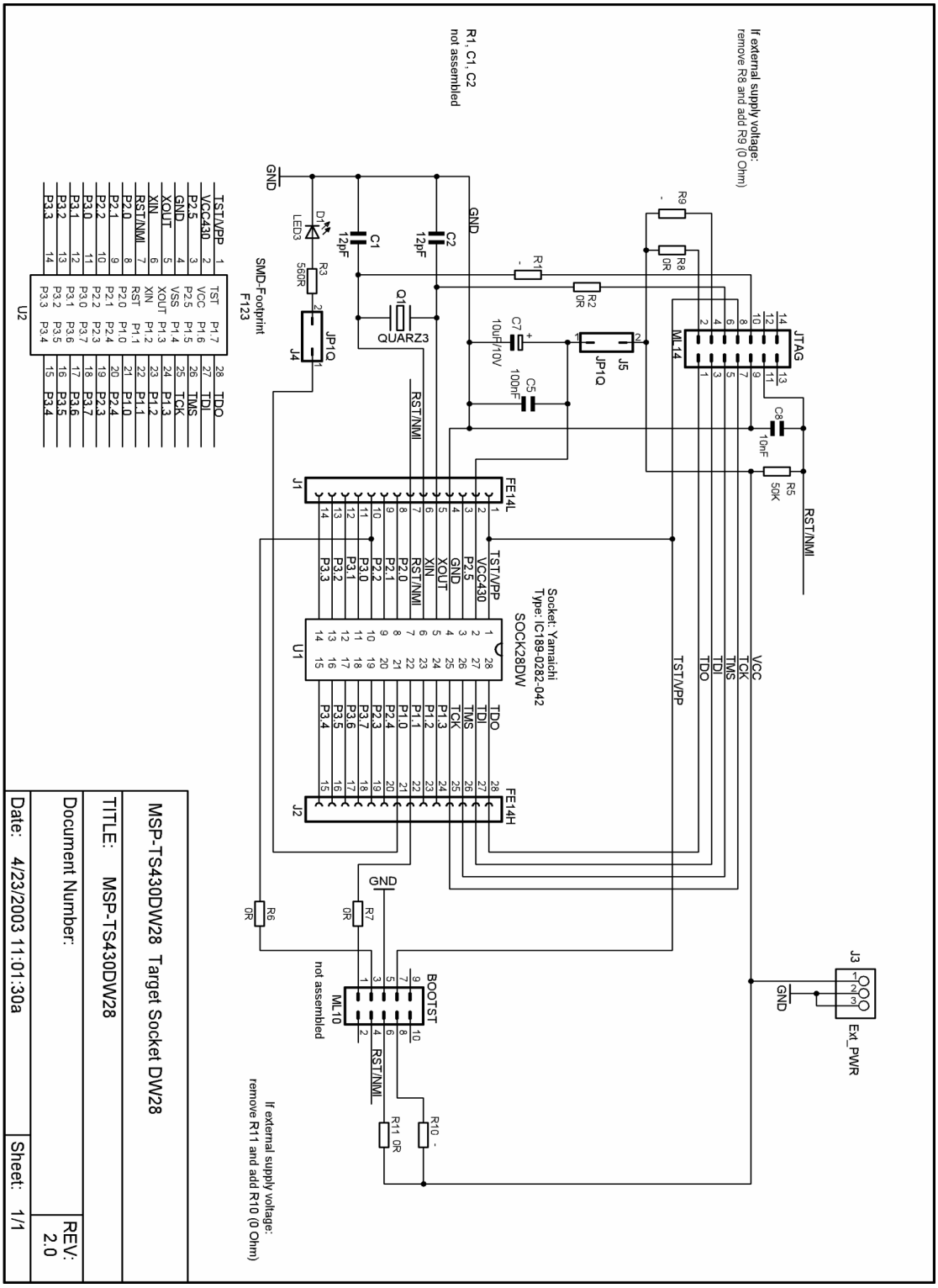


Figure B-4. MSP-TS430PW14 Target Socket Module, PCB



Note: Connections between the JTAG header and pins XOUT and XIN are not required and should not be made.

Figure B-5. MSP-TS430DW28 Target Socket Module, Schematic

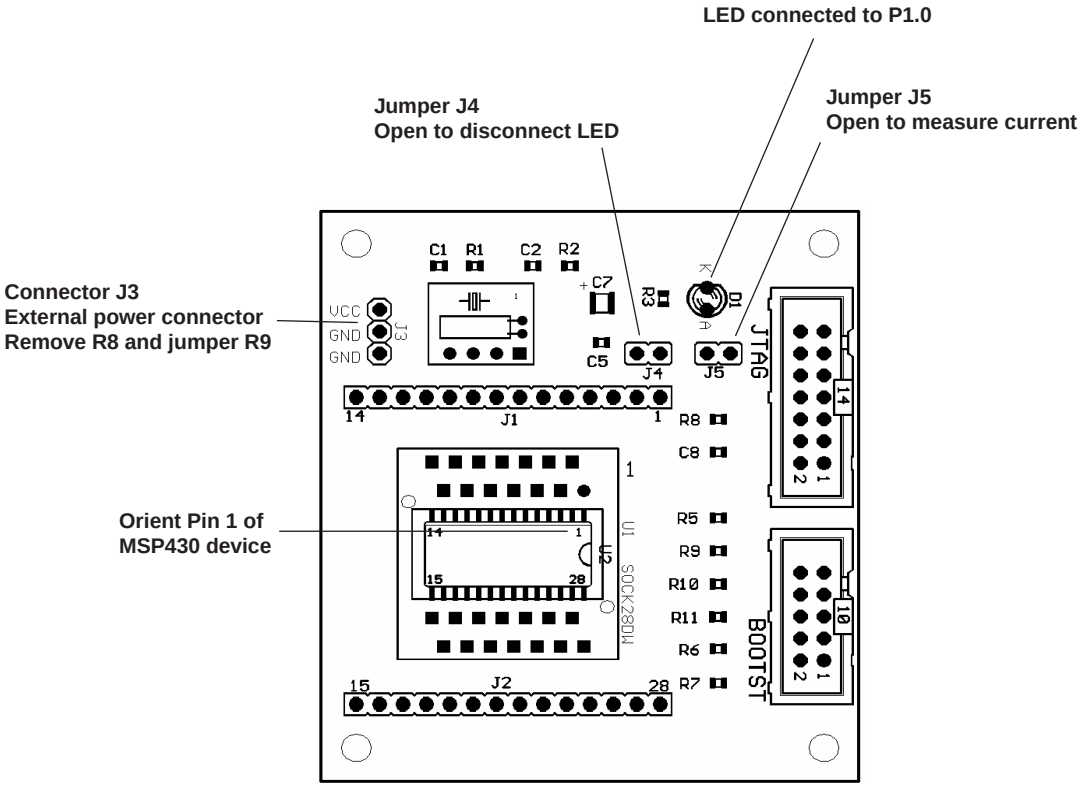


Figure B-6. MSP-TS430DW28 Target Socket Module, PCB

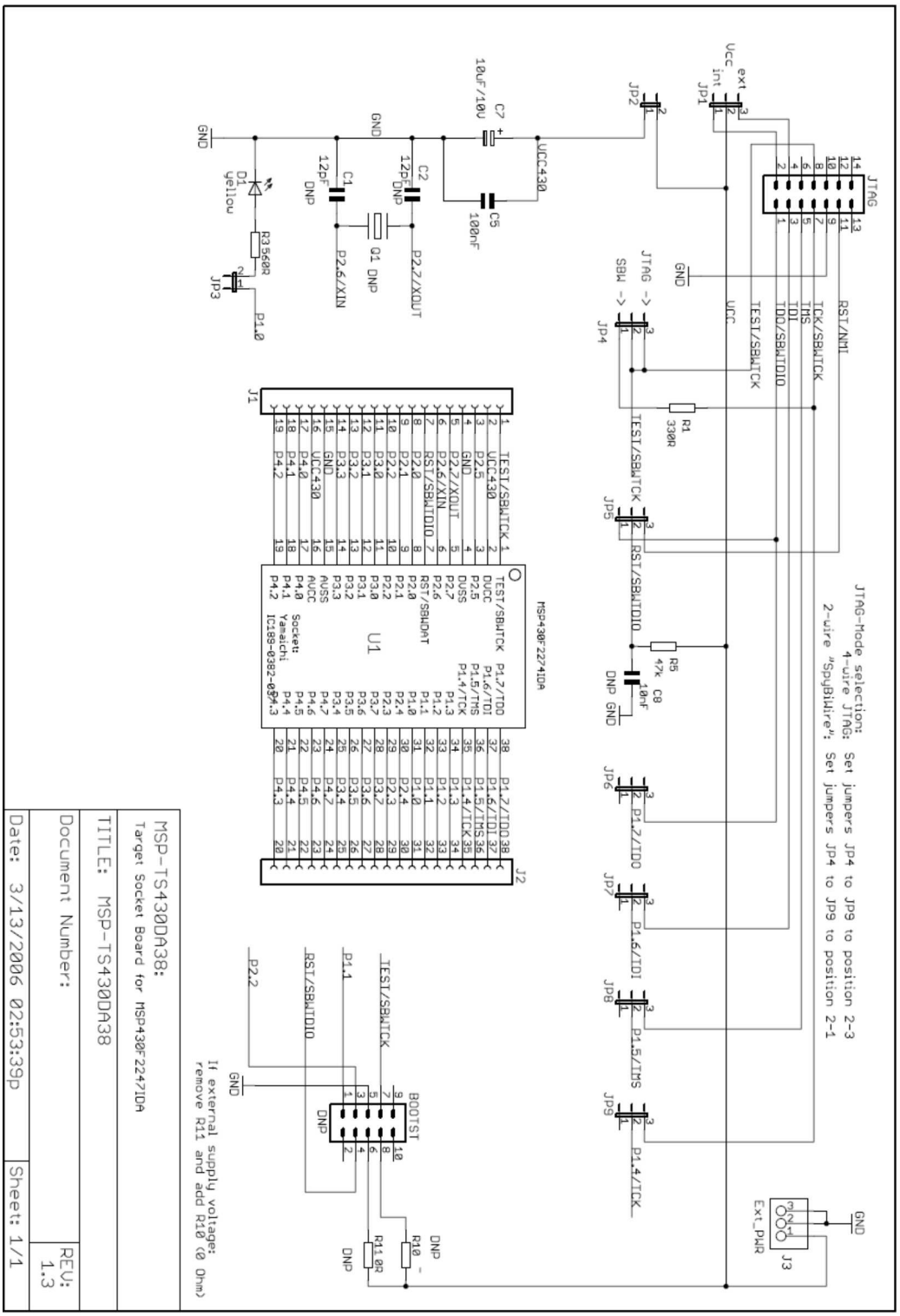


Figure B-7. MSP-TS430DA38 Target Socket Module, Schematic

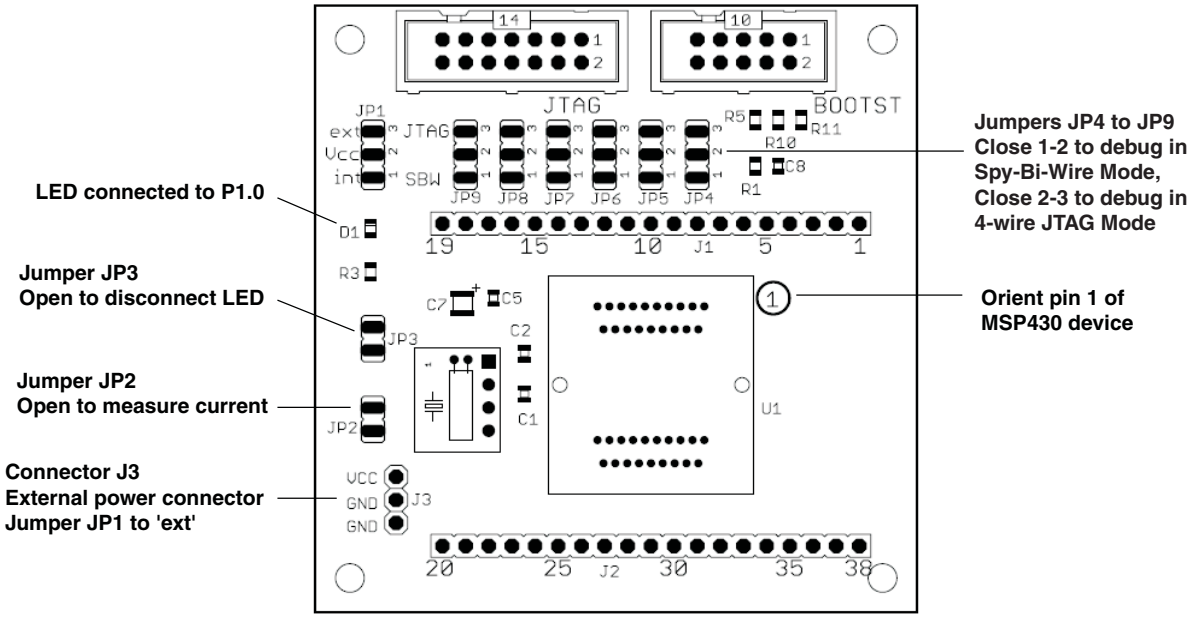


Figure B-8. MSP-TS430DA38 Target Socket Module, PCB

Note: See Section A.1, note 2.

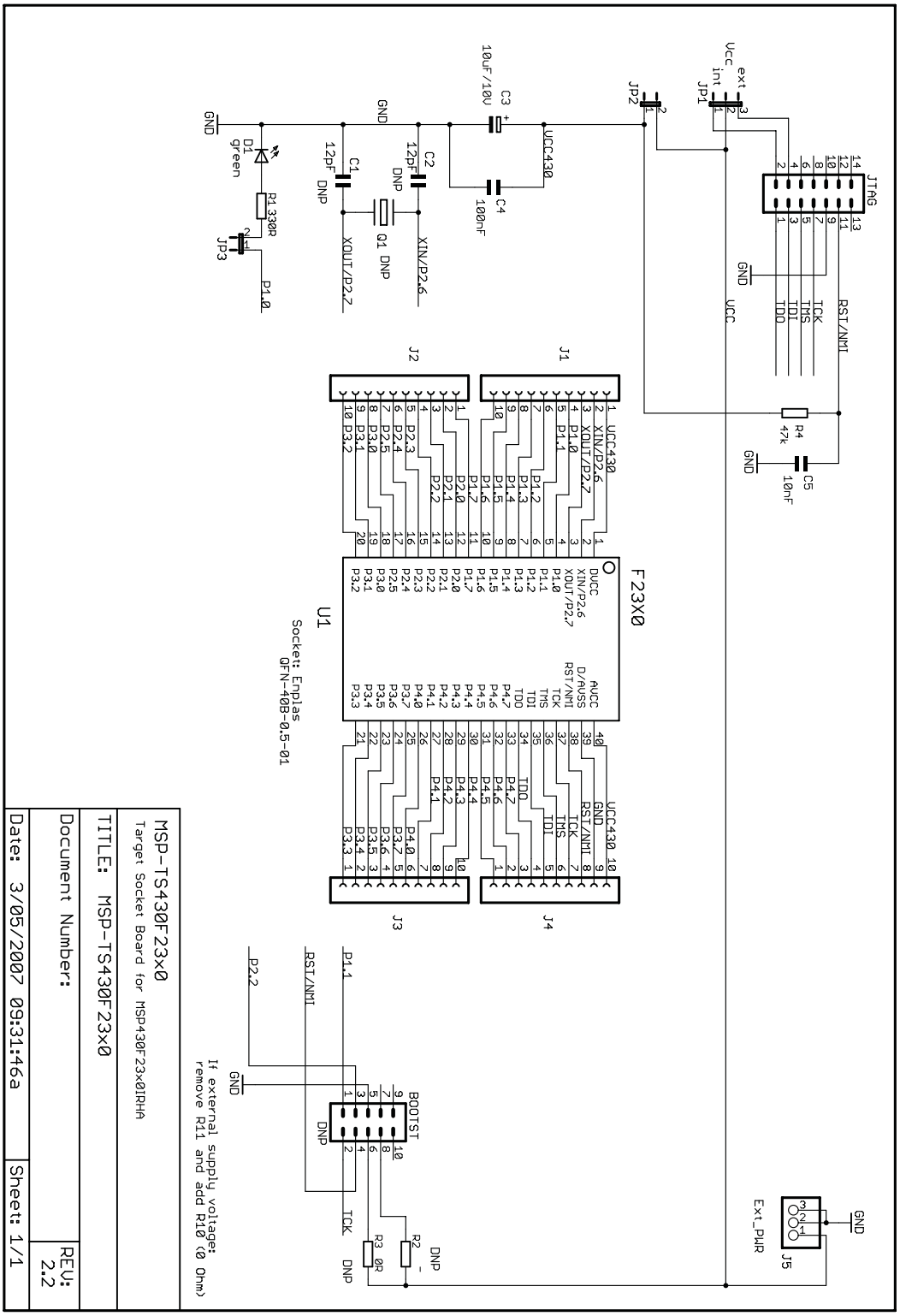


Figure B-9. MSP-TS430QFN23x0 Target Socket Module, Schematic

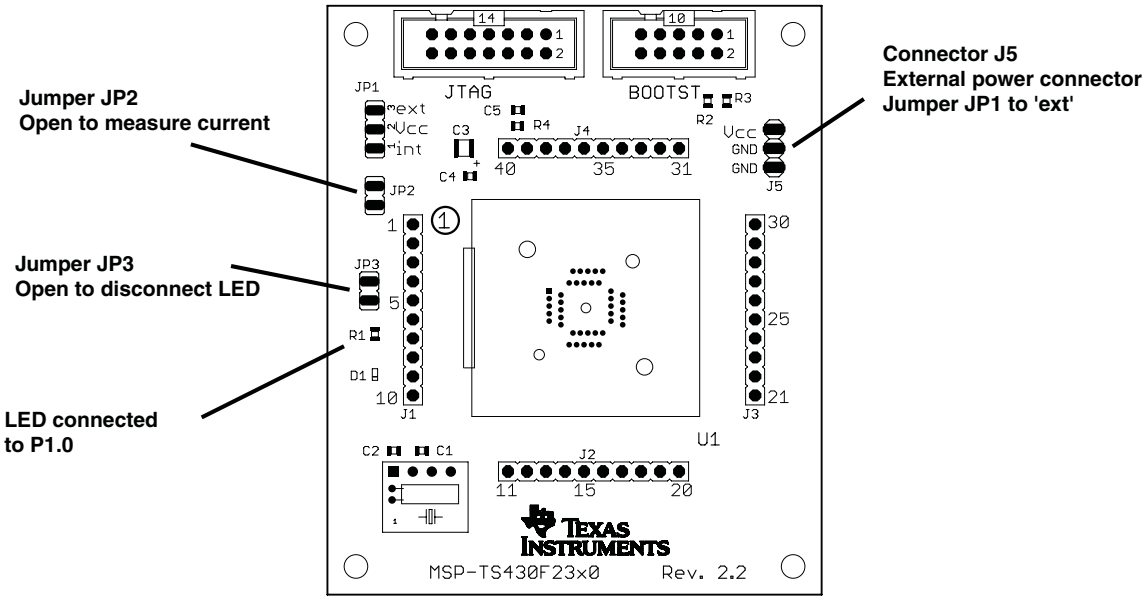


Figure B-10. MSP-TS430QFN23x0 Target Socket Module, PCB

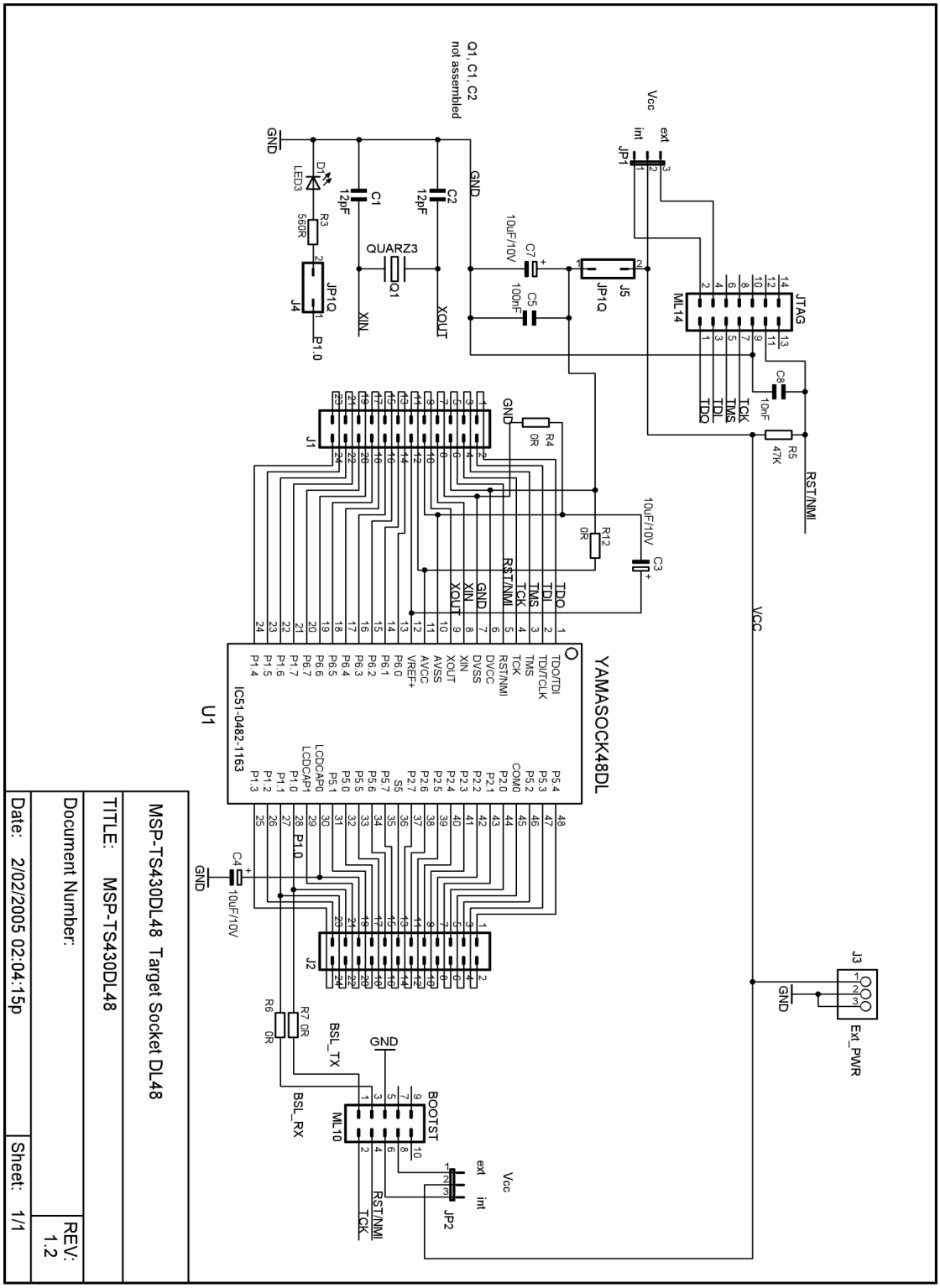


Figure B-11. MSP-TS430DL48 Target Socket Module, Schematic

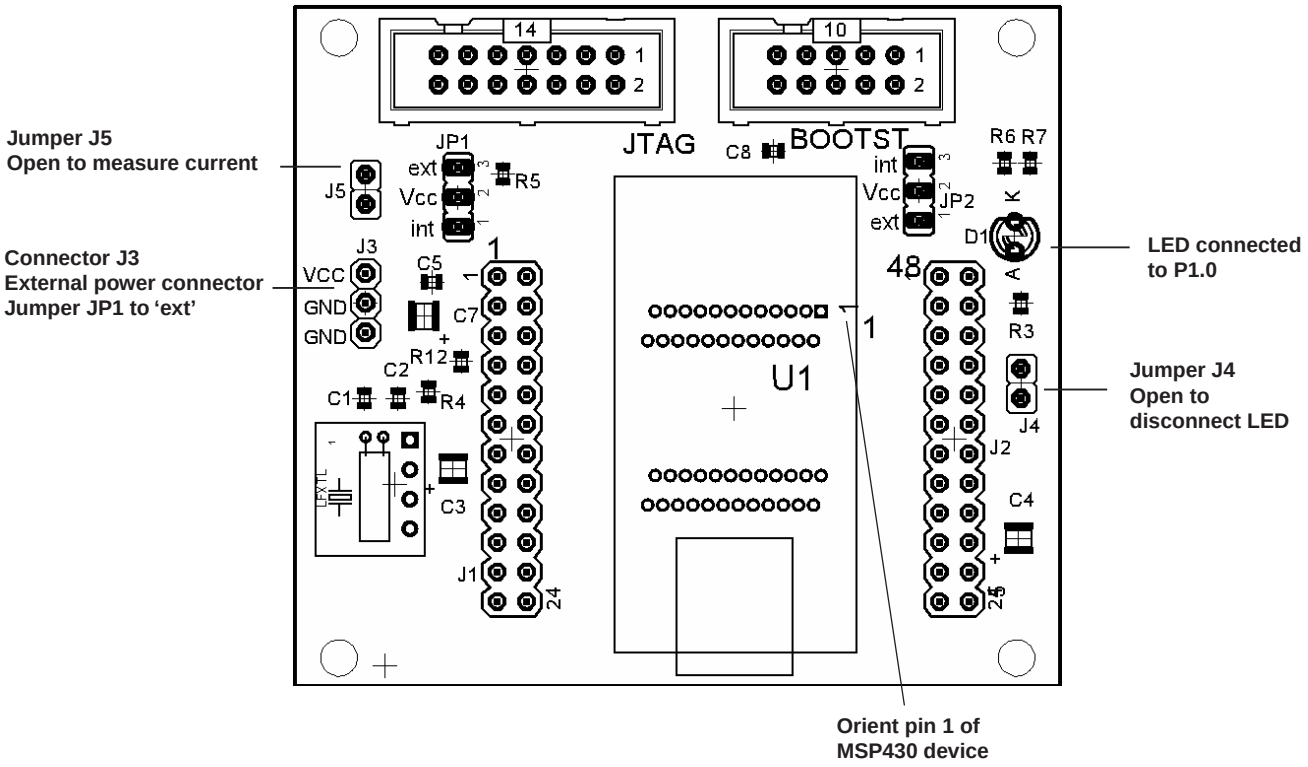
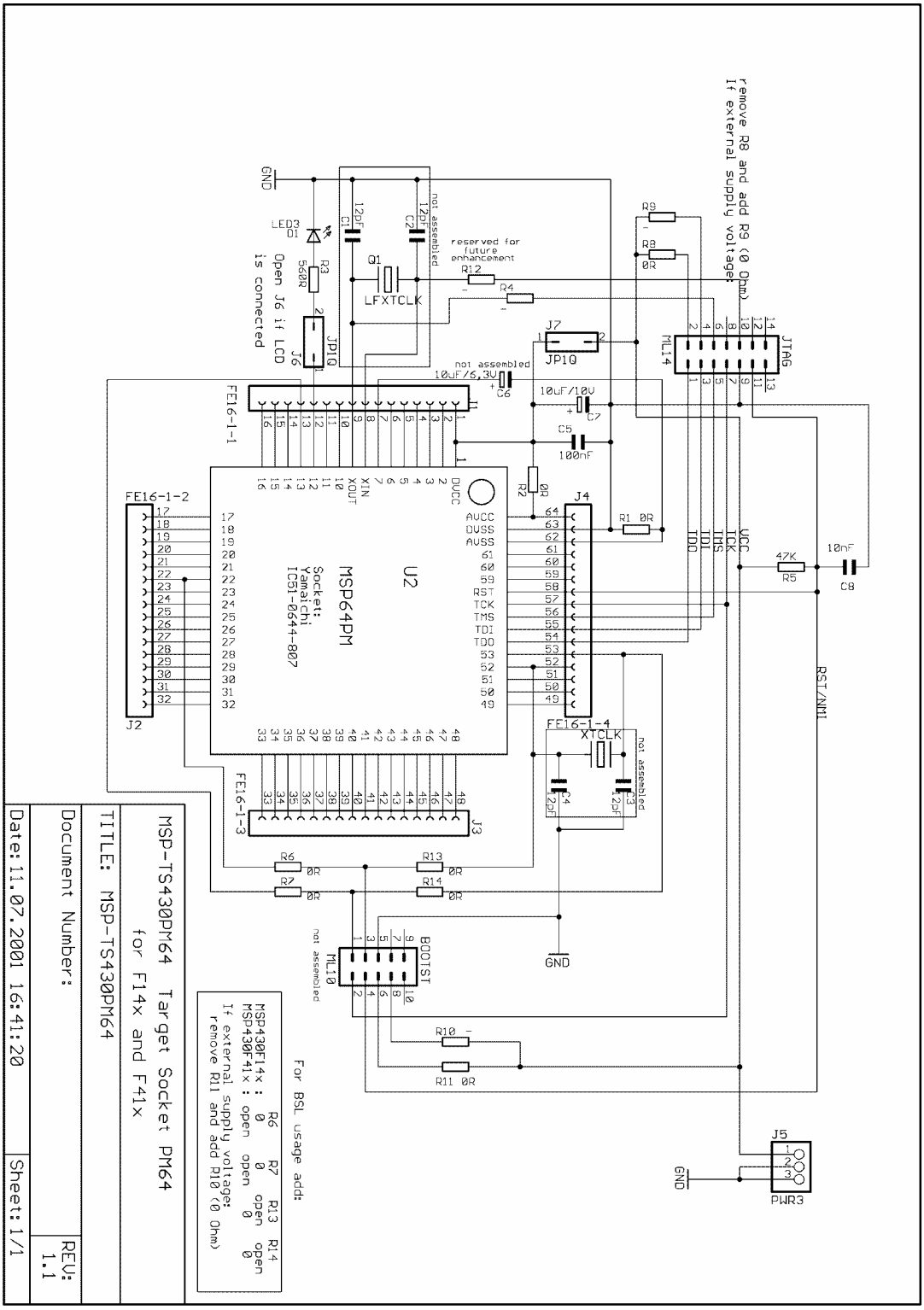


Figure B-12. MSP-TS430DL48 Target Socket Module, PCB



Note: Connections between the JTAG header and pins XOUT and XIN are not required and should not be made.

Figure B-13. MSP-TS430PM64 Target Socket Module, Schematic

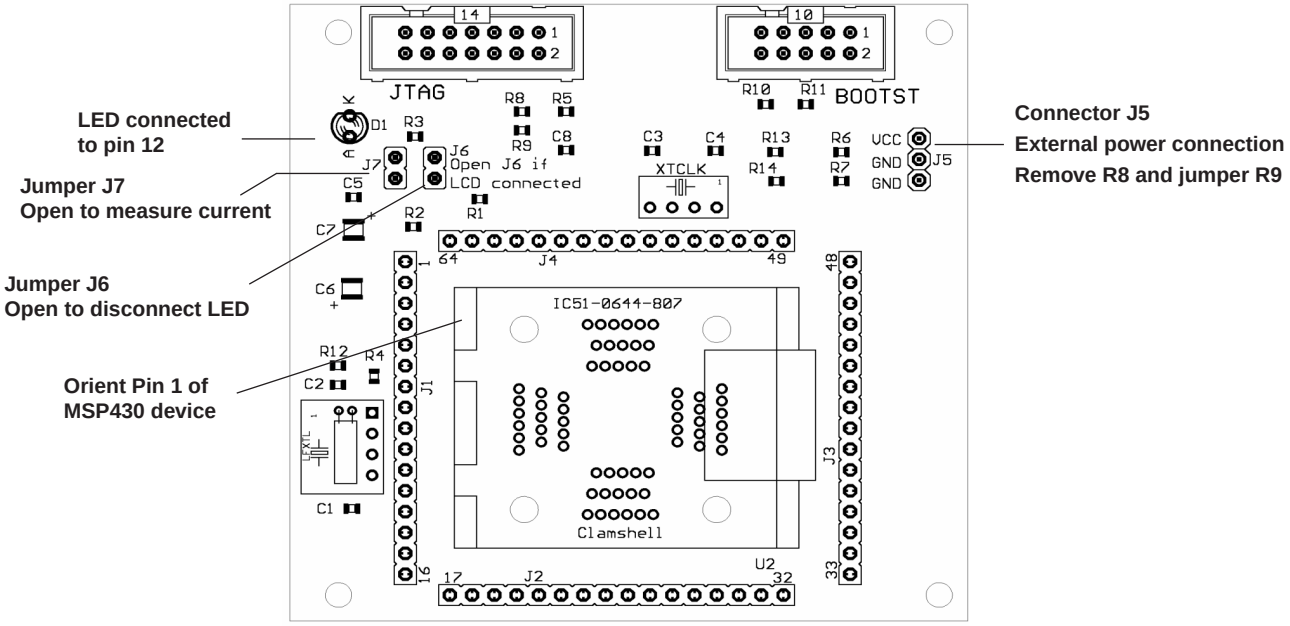


Figure B-14. MSP-TS430PM64 Target Socket Module, PCB

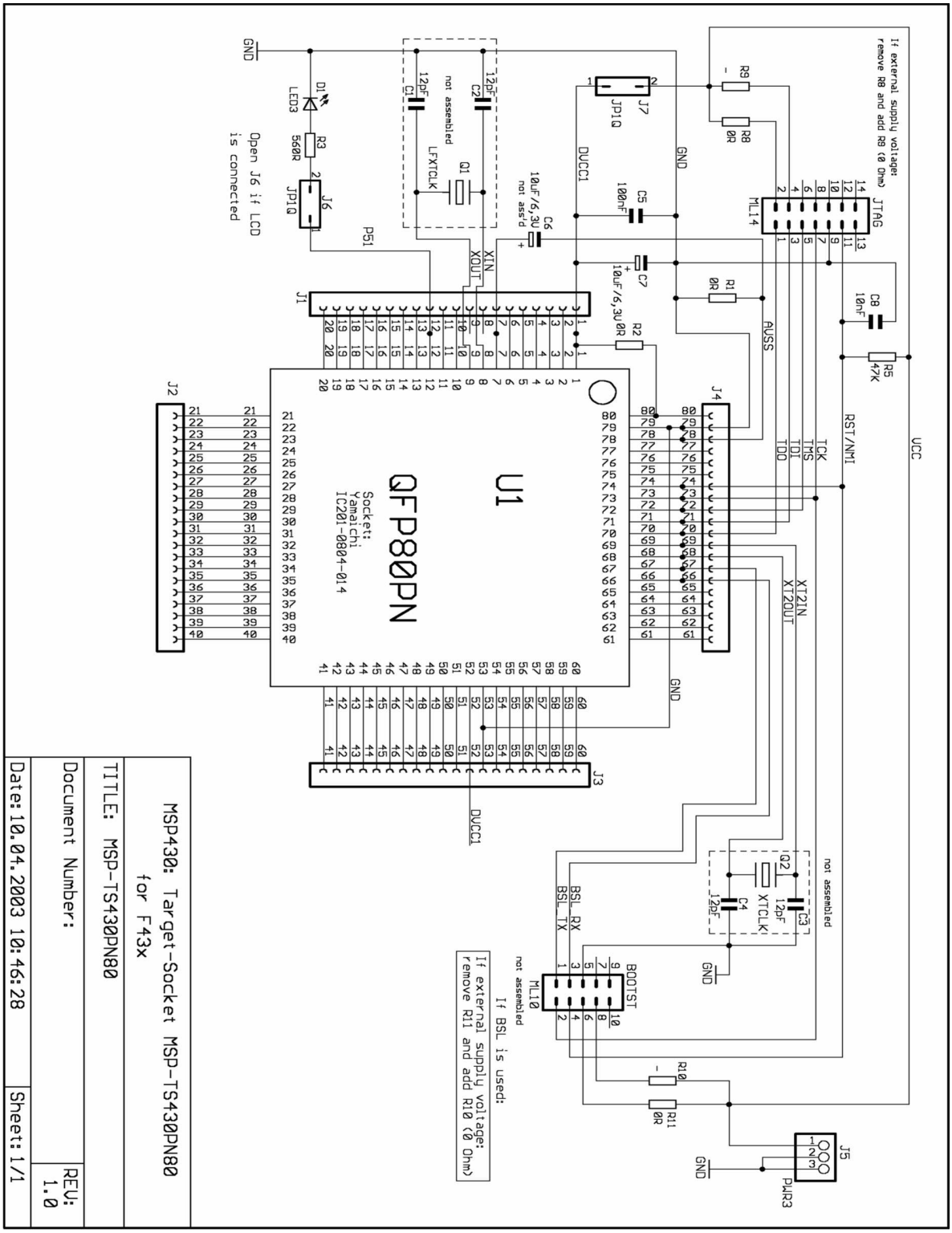


Figure B-15. MSP-TS430PN80 Target Socket Module, Schematic

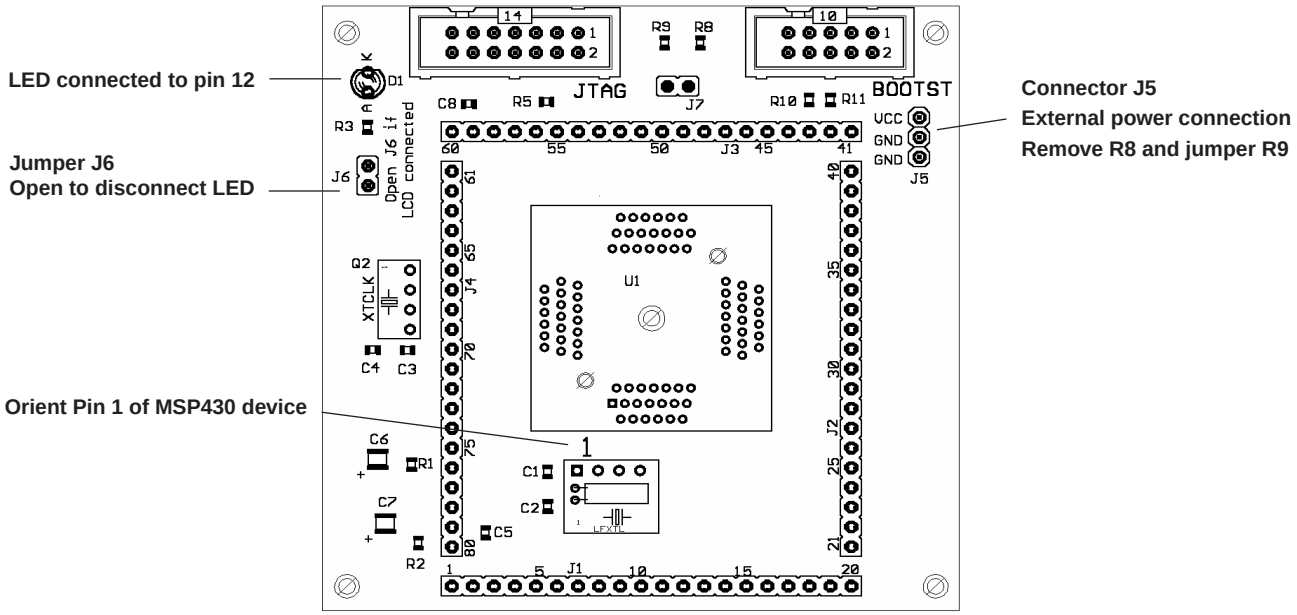
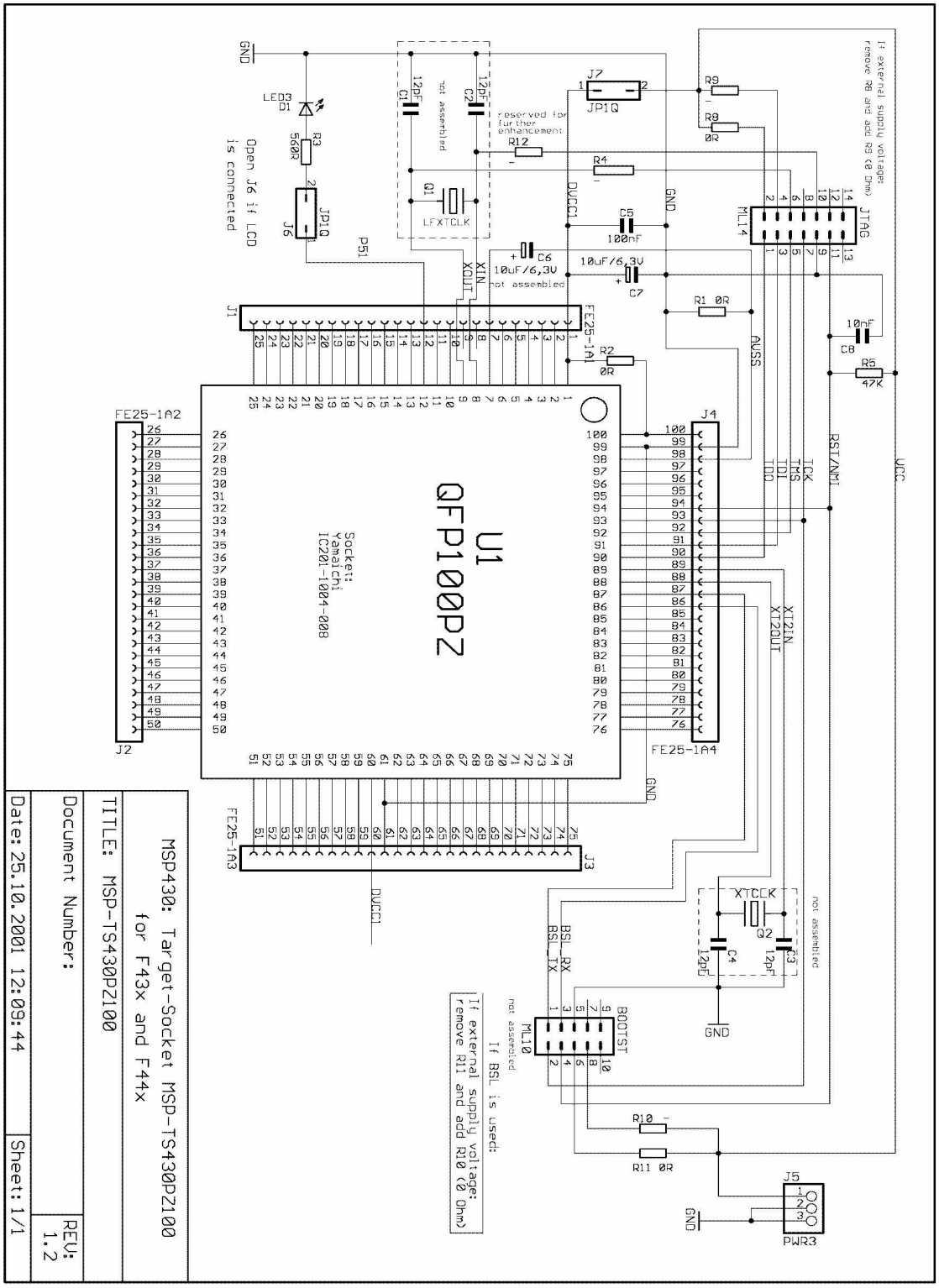


Figure B-16. MSP-TS430PN80 Target Socket Module, PCB



Note: Connections between the JTAG header and pins XOUT and XIN are not required and should not be made.

Figure B-17. MSP-TS430PZ100 Target Socket Module, Schematic

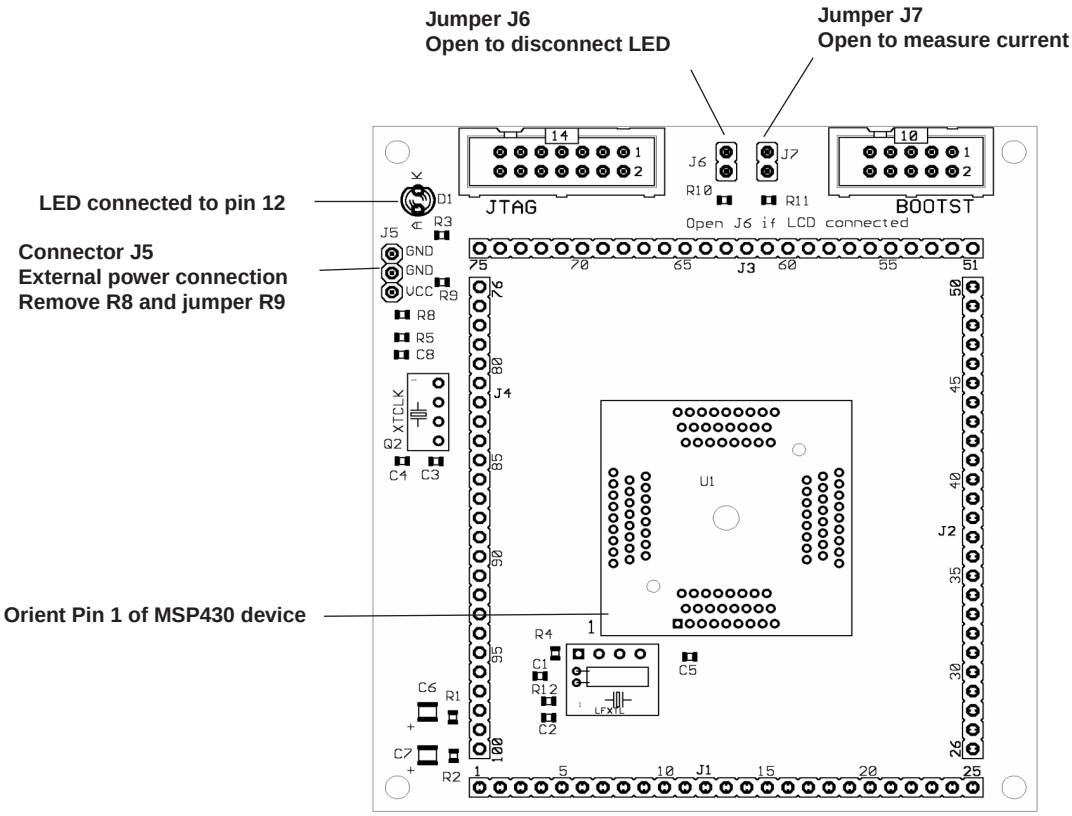


Figure B-18. MSP-TS430PZ100 Target Socket Module, PCB

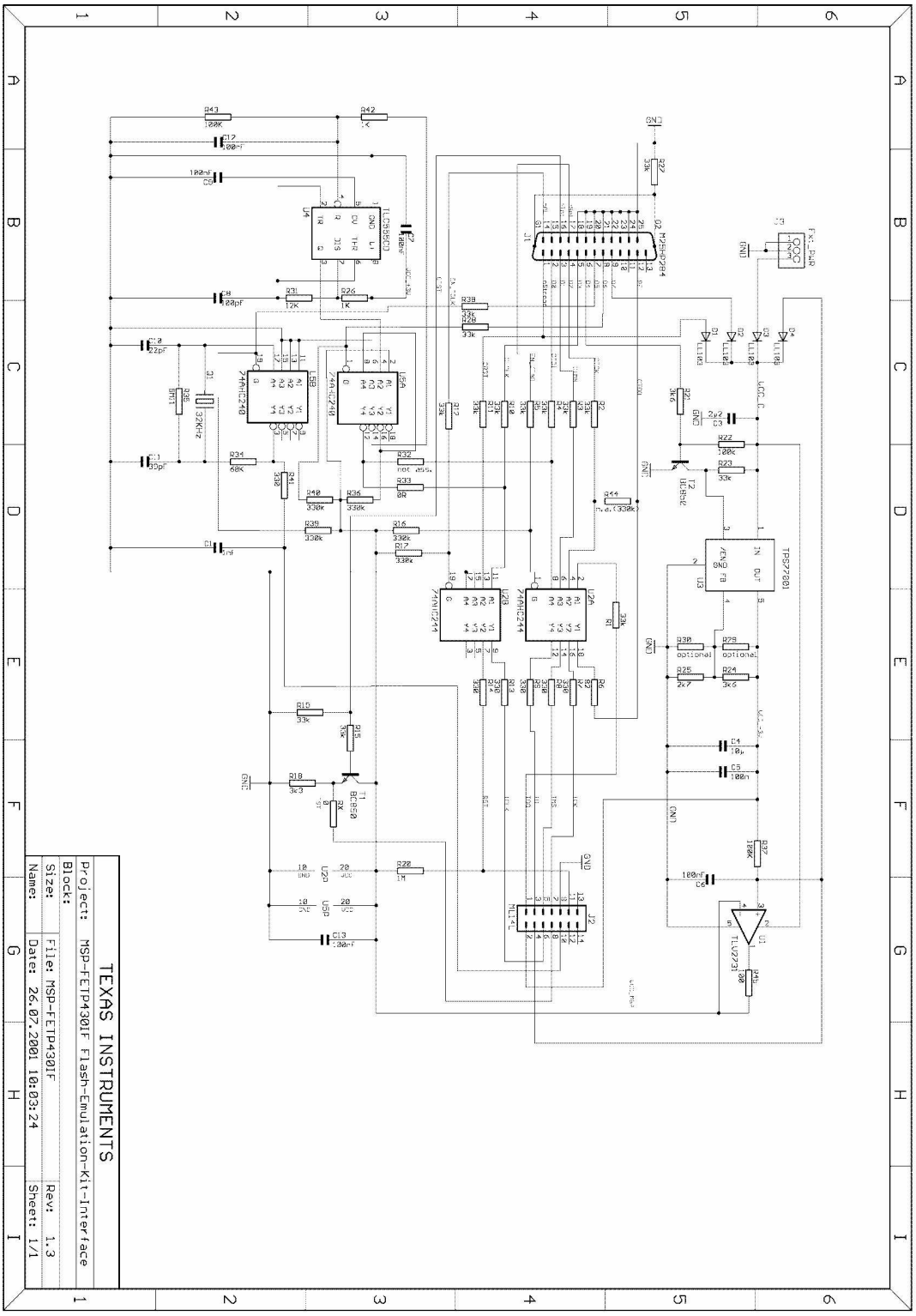


Figure B-19. MSP-FET430PIF FET Interface Module, Schematic

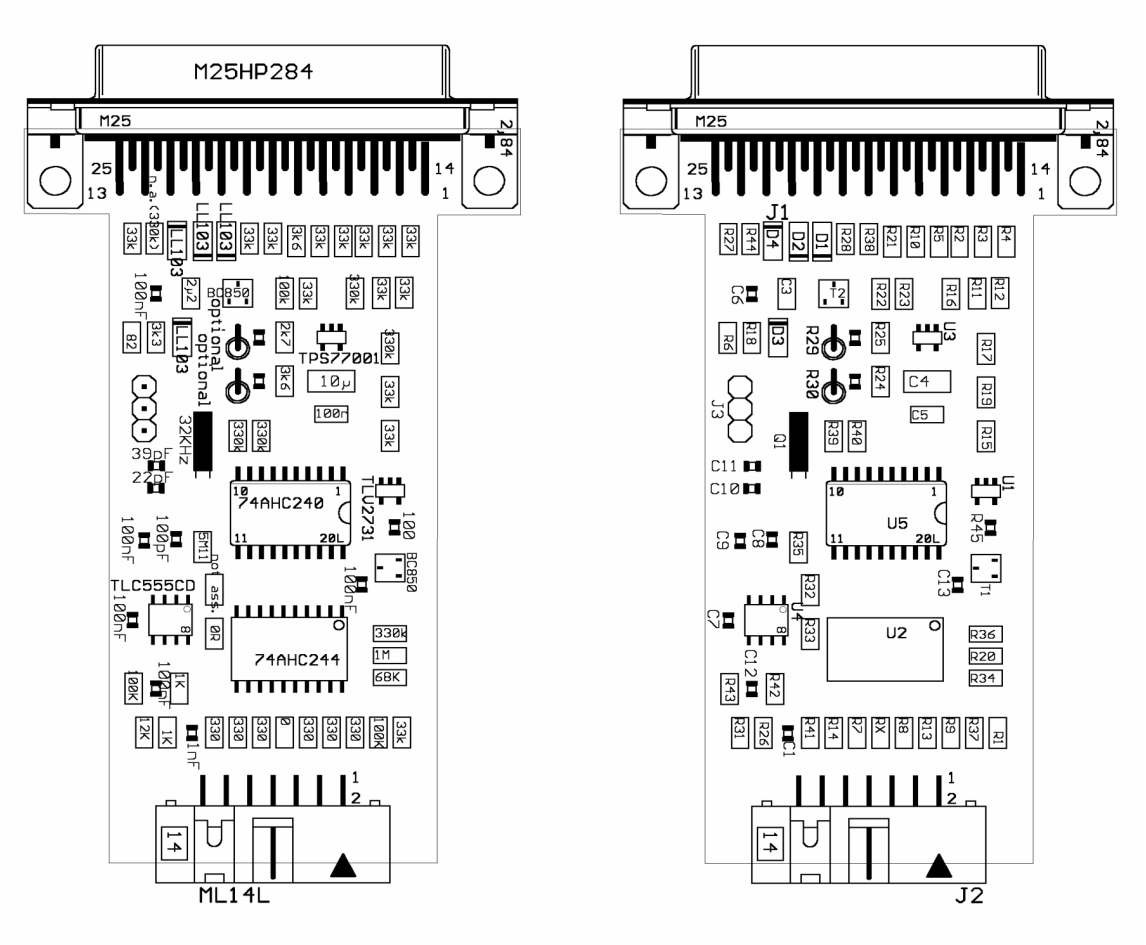


Figure B-20. MSP-FET430PIF FET Interface Module, PCB

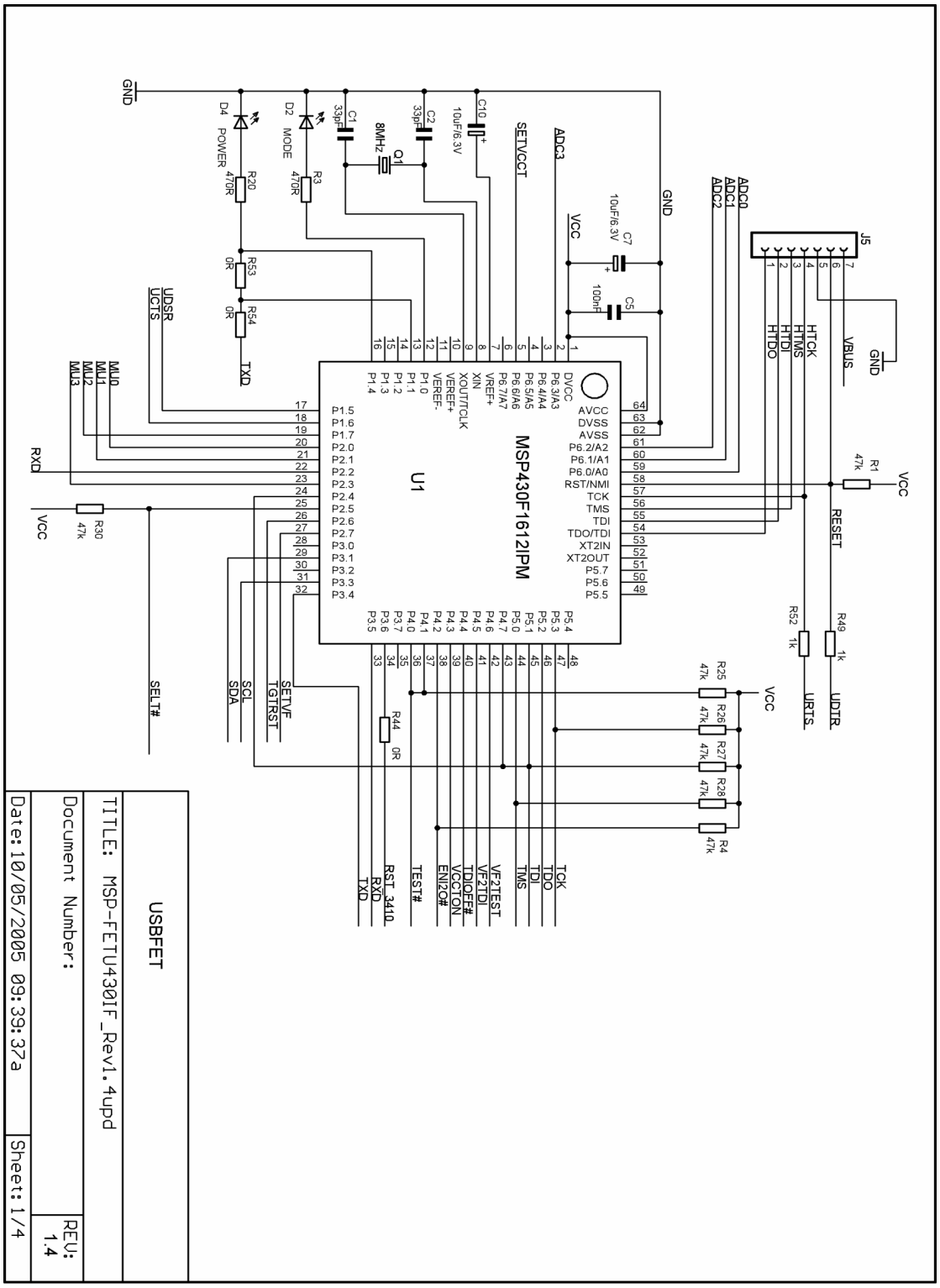


Figure B-21. MSP-FET430UIF USB Interface, Schematic (1 of 4)

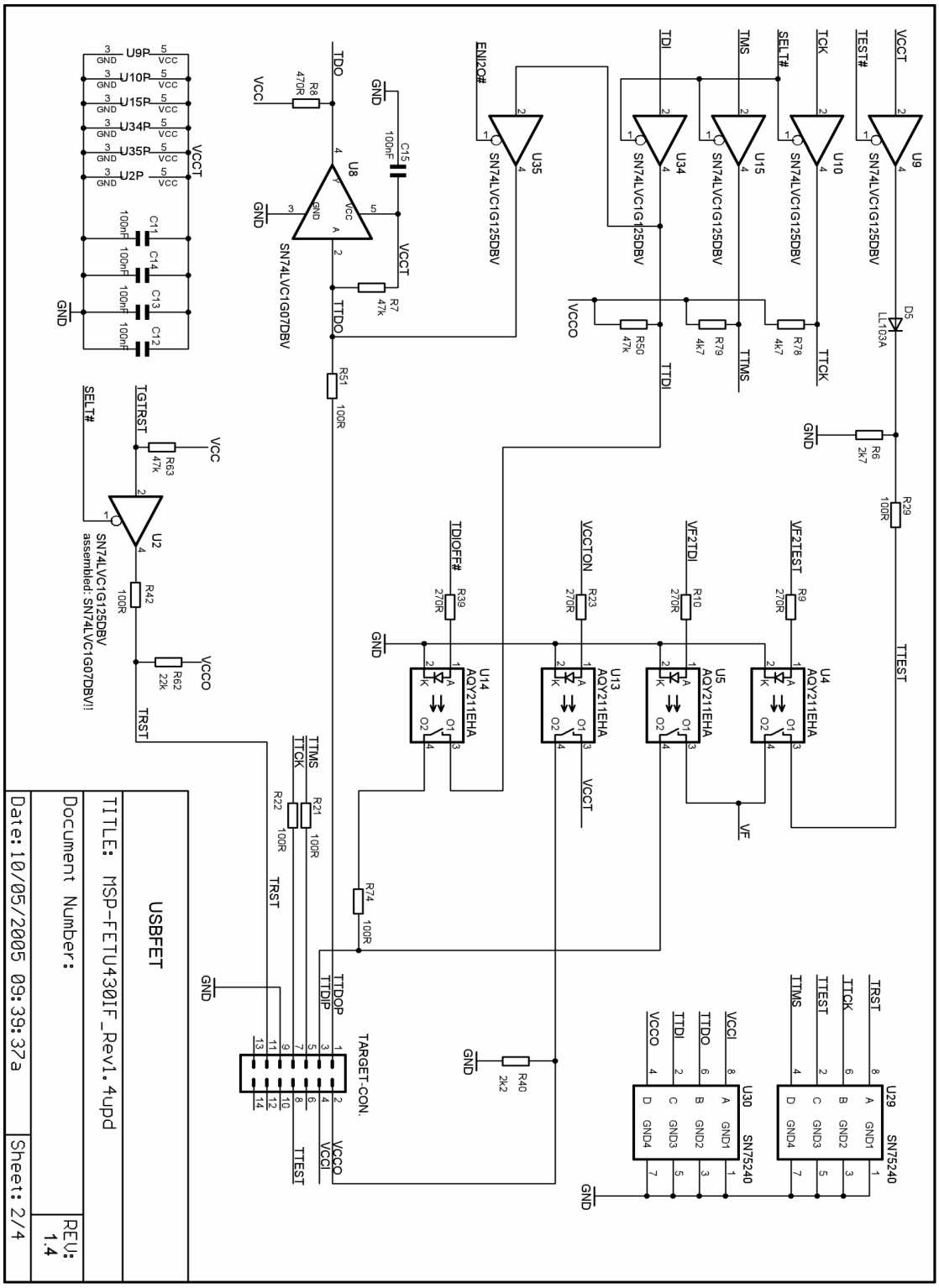


Figure B-22. MSP-FET430UIF USB Interface, Schematic (2 of 4)

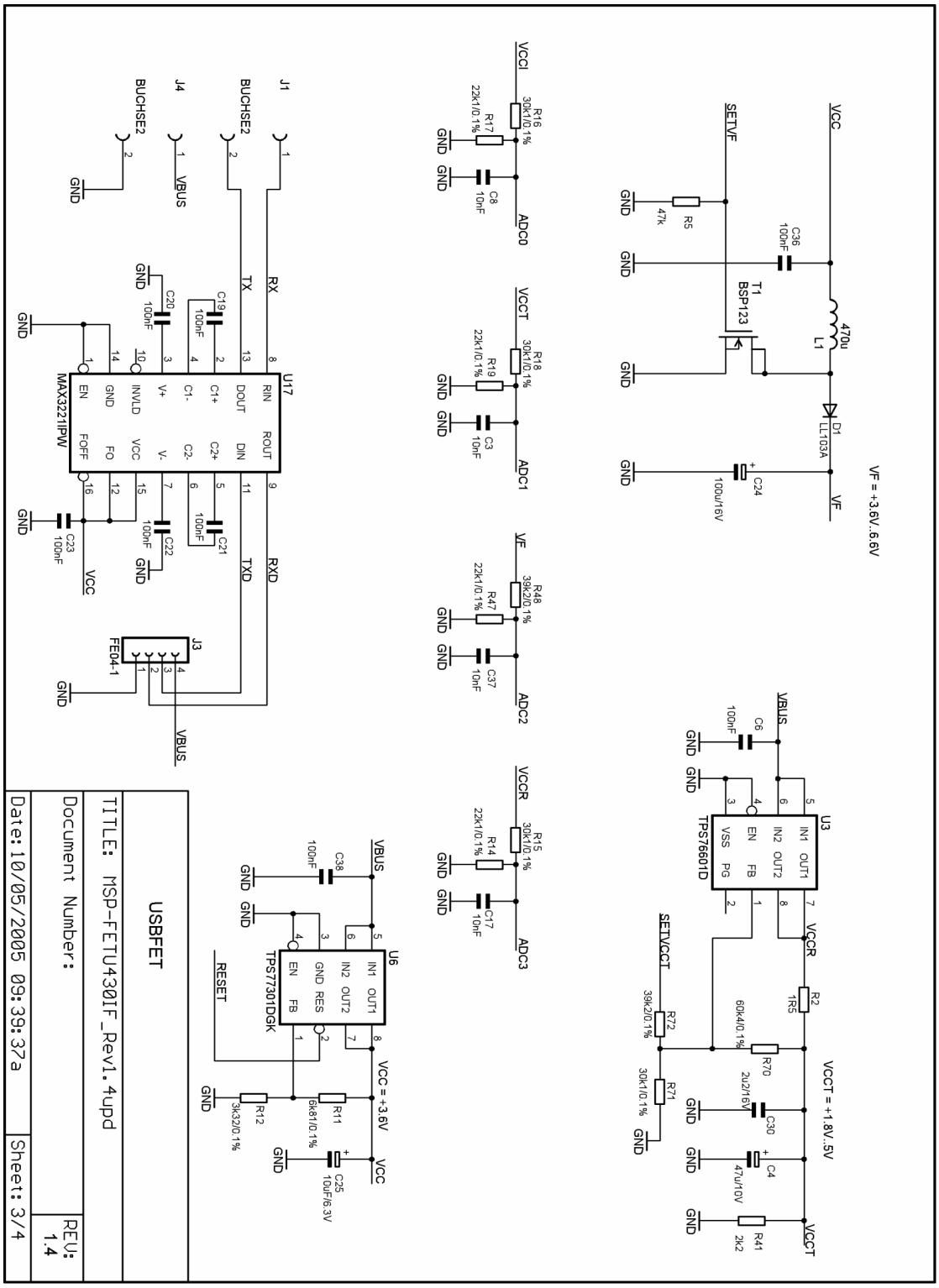


Figure B-23. MSP-FET430UIF USB Interface, Schematic (3 of 4)

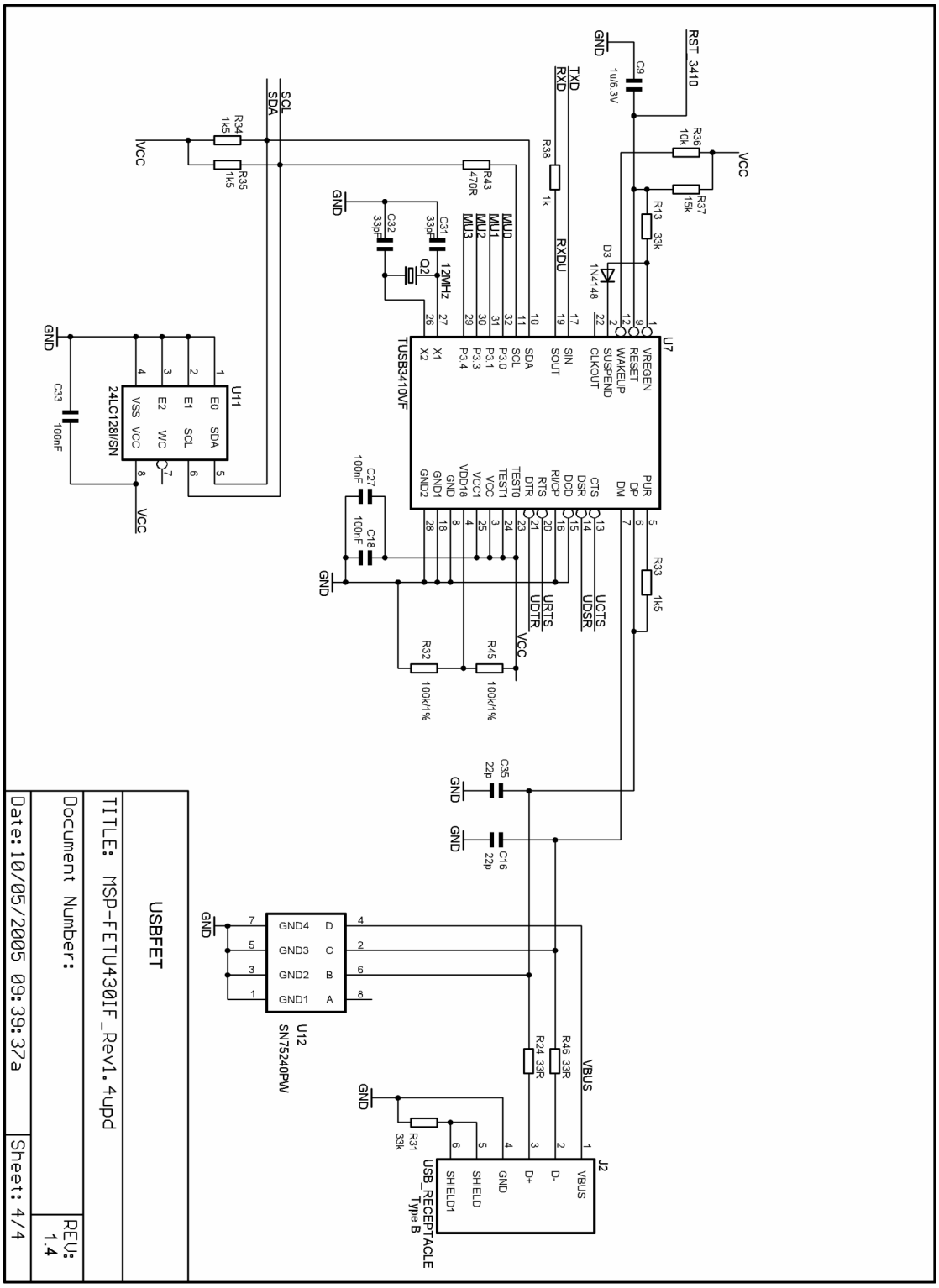


Figure B-24. MSP-FET430UIF USB Interface, Schematic (4 of 4)

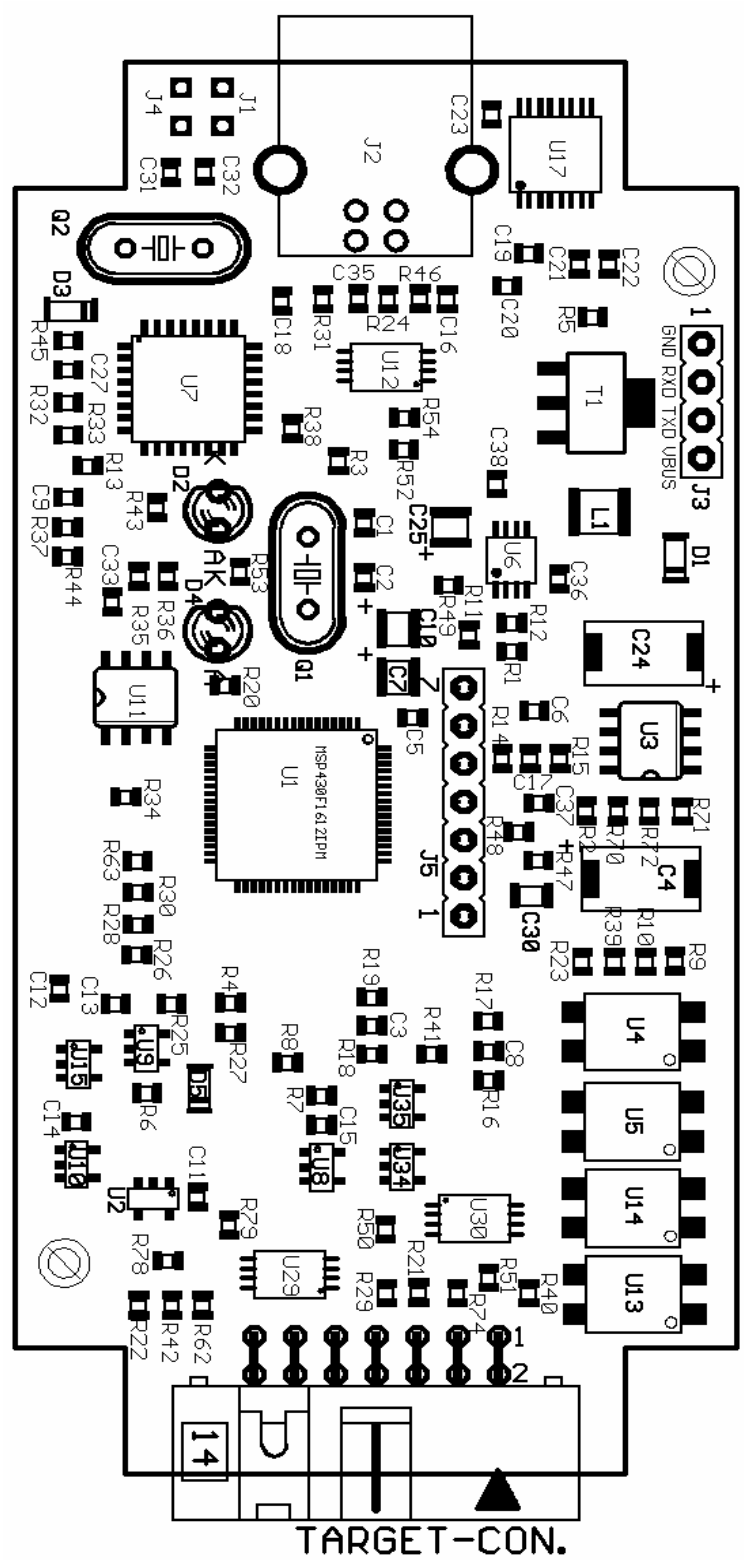


Figure B-25. MSP-FET430UIF USB Interface, PCB

B.2 MSP-FET430UIF Revision History

Revision 1.3

- Initial released hardware version

Assembly change on 1.3 (May 2005)

- R29, R51, R42, R21, R22, R74: value changed from 330R to 100R

Changes 1.3 to 1.4 (Aug 2005)

- J5: VBUS and RESET additionally connected
- R29, R51, R42, R21, R22, R74: value changed from 330R to 100R
- U1, U7: F1612 can reset TUSB3410; R44 = 0R added
- TARGET-CON: pins 6, 10, 12, 13, 14 disconnected from GND
- Firmware-upgrade option through BSL: R49, R52, R53, R54 added; R49, R52 are currently DNP
- Pullups on TCK and TMS: R78, R79 added
- U2: Changed from SN75LVC1G125DBV to SN75LVC1G07DBV

Assembly change on 1.4 (January 2006)

- R62: not populated

Note: Using a locally powered target board with hardware revision 1.4

Using an MSP-FET430UIF interface hardware revision 1.4 with populated R62 in conjunction with a locally powered target board is not possible. In this case, the target device RESET signal is pulled down by the FET tool. It is recommended to remove R62 to eliminate this restriction. This component is located close to the 14-pin connector on the MSP-FET430UIF PCB. See Figure B-25 for the exact location.

FET-Specific Menus

Topic **Page**

C.1 Menus

C.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.

C.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 tri-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 will be determined (i.e., was a breakpoint reached?) (see FAQ Debugging #12).

C.1.3 Emulator → Resynchronize JTAG

Regain control of the device.

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

C.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.

C.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.

C.1.6 Emulator → Breakpoint Usage

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

C.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/Step Into) (see FAQ Debugging #19).

C.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.

See Appendix D.

C.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 can also be appended to the file.

C.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 will be triggered when the breakpoints are encountered in the specified order.

C.1.11 Emulator → State Storage Control

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

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

C.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.

C.1.13 Emulator → Sequencer Control

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

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

C.1.14 Emulator → "Power on" Reset

Cycle power to the device to effect a reset.

C.1.15 Emulator → GIE on/off

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

C.1.16 Emulator → Leave Target Running

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

C.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 will be grayed out.

80-Pin MSP430F44x and MSP430F43x Device Emulation

80-pin MSP430F44x and MSP430F43x devices can be emulated by the 100-pin MSP430F449 device. Table D-1 lists where the pin signals of an 80-pin device appear on the pins of an MSP-TS430PZ100 target socket module.

Note: The MSP-TS430PZ100 must be modified as indicated. See Section C.1.8, Emulator → Advanced → Emulation Mode to enable the emulation mode.

Topic **Page**

D.1 F4xx/80-Pin Signal Mapping

Table D-1. F4xx/80-pin Signal Mapping

F4xx/80-pin Signal	F4xx/80-pin Pin Number	MSP430-TS430PZ100 Pin Number	Connection Required Between Indicated Pins of MSP430-TS430PZ100 Socket
DVcc1	1	1	
P6.3/A3	2	2	
P6.4/A4	3	3	
P6.5/A5	4	4	
P6.6/A6	5	5	
P6.7/A7	6	6	
VREF+	7	7	
XIN	8	8	
XOUT	9	9	
VeREF+	10	10	
VREF-/VeREF-	11	11	
P5.1/S0	12	12	
P5.0/S1	13	13	
P4.7/S2	14	14	14-46
P4.6/S3	15	15	15-47
P4.5/S4	16	16	16-48
P4.4/S5	17	17	17-49
P4.3/S6	18	16	18-50
P4.2/S7	19	19	19-51
P4.1/S8	20	20	20-62
P4.0/S9	21	21	21-63
S10	22	22	
S11	23	23	
S12	24	24	
S13	25	25	
S14	26	26	
S15	27	27	
S16	28	28	
S17	29	29	
P2.7/ADC12CLK/S18	30	30	
P2.6/CAOUT/S19	31	31	
S20	32	32	
S21	33	33	
S22	34	34	
S23	35	35	
P3.7/S24	36	36	36-64
P3.6/S25	37	37	37-65
P3.5/S24	38	38	38-66
P3.4/S27	39	39	39-67
P3.3/UCLK0/S28	40	40	40-68
P3.2/SOMI0/S29	41	41	41-69
P3.1/SIMO0/S30	42	42	42-70

Table D-1. F4xx/80-pin Signal Mapping (continued)

F4xx/80-pin Signal	F4xx/80-pin Pin Number	MSP430-TS430PZ100 Pin Number	Connection Required Between Indicated Pins of MSP430-TS430PZ100 Socket
P3.0/STE0/S31	43	43	43-71
COM0	44	52 (1)	
P5.2/COM1	45	53	
P5.3/COM2	46	54	
P5.4/COM3	47	55	
R03	48	56	
P5.5/R13	49	57	
P5.6/R23	50	58	
P5.7/R33	51	59	
DVcc2	52	60	
DVss2	53	61	
P2.5/URXD0	54	74 (1)	
P2.4/UTXD0	55	75	
P2.3.TB2	56	76	
P2.2/TB1	57	77	
P2.1/TB0	58	78	
P2.0/TA2	59	79	
P1.7/CA1	60	80	
P1.6/CA0	61	81	
P1.5/TACLK/ACLK	62	82	
P1.4/TBCLK/SMCLK	63	83	
P1.3/TBOUTH/SVSOUT	64	84	
P1.2/TA1	65	85	
P1.1/TA0/MCLK	66	86	
P1.0/TA0	67	87	
XT2OUT	68	88	
XT2IN	69	89	
TDO/TDI	70	90	
TDI	71	91	
TMS	72	92	
TCK	73	93	
RST/NMI	74	94	
P6.0/A0	75	95	
P6.1/A1	76	96	
P6.2/A2	77	97	
Avss	78	98	
DVss1	79	99	
Avcc	80	100	

(1) Note discontinuity of pin numbering sequence

MSP-FET430UIF Installation Guide

This section describes the hardware installation process of the MSP-FET430UIF USB debug interface on a PC running Windows XP. The installation procedure for a Windows 2000 system is very similar and, therefore, not shown here.

Topic **Page**

E.1 Hardware Installation

1. Connect the MSP-FET430UIF USB Debug Interface with a USB cable to a USB port of your PC.
2. Windows now should recognize the new hardware as an "MSP430 USB FET x.xx.xx" (see Figure E-1).

Figure E-1. WinXP Hardware Recognition

3. The Hardware Wizard starts automatically and opens the "Found New Hardware Wizard" window.
4. Select "Install from a list or specific location (Advanced)" (see Figure E-2).

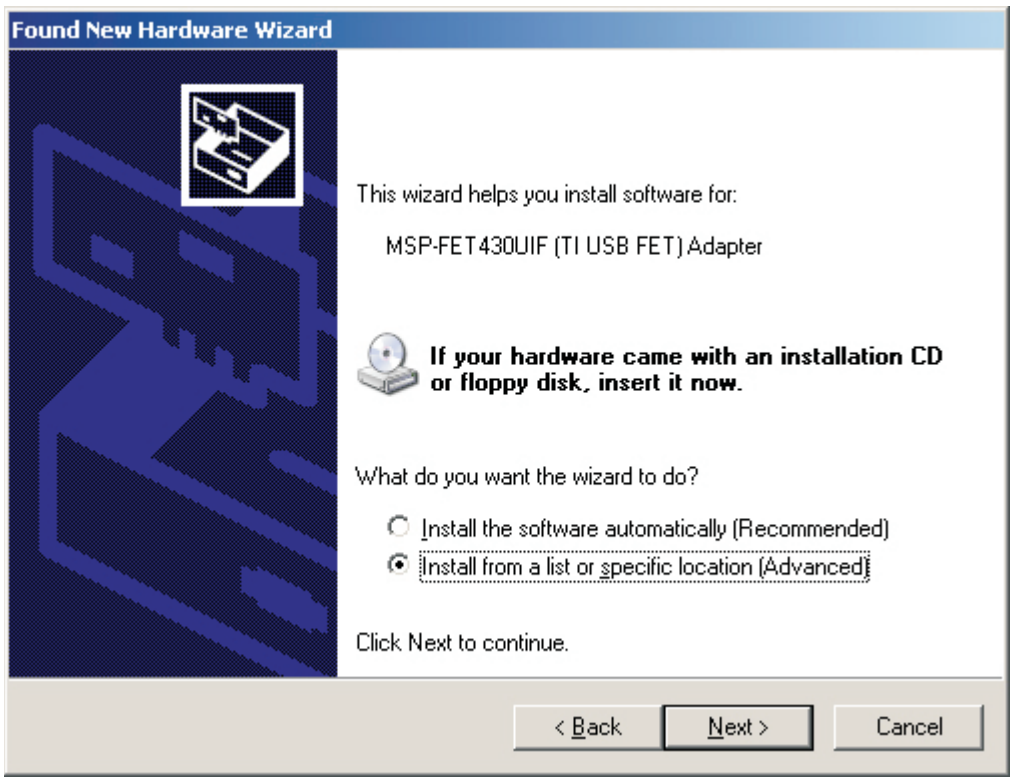


Figure E-2. WinXP Hardware Wizard

5. Browse to the folder where the driver information files are located (see Figure E-3). On a default installation the files are located in the following directory:
   C:\Program Files\IAR Systems\Embedded Workbench 4.0\430\drivers\TIUSBFET\WinXP

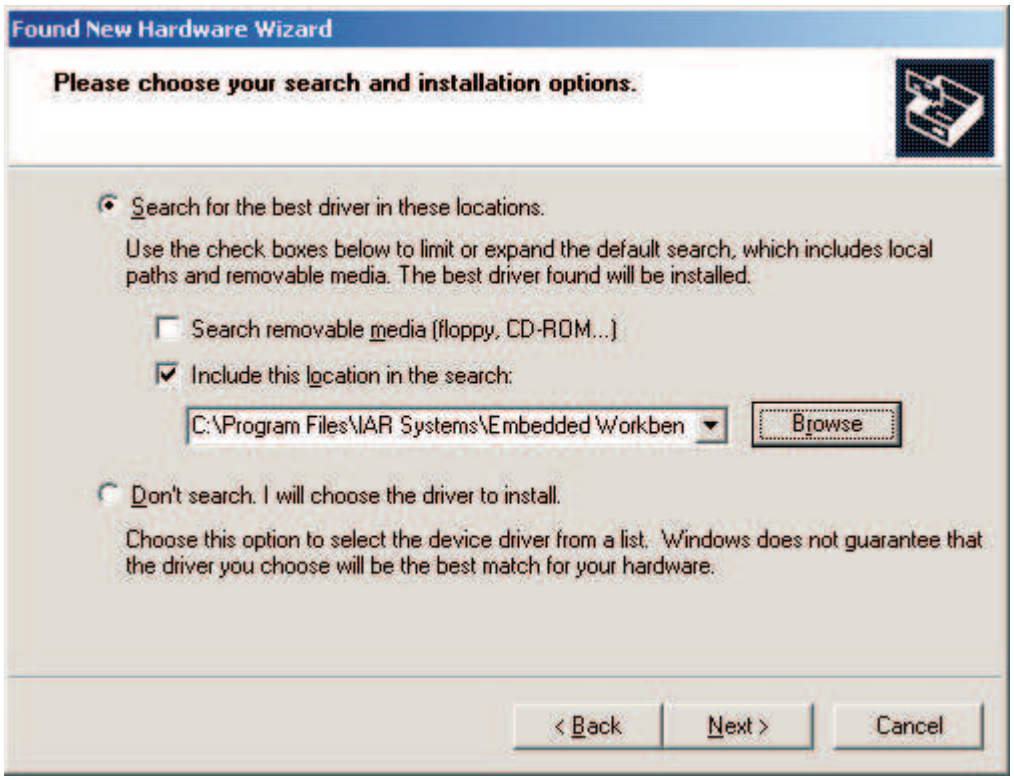


Figure E-3. WinXP Driver Location Selection Folder

6. The Wizard generates a message that an appropriate driver has been found.

7. Note that WinXP shows a warning that the driver is not certified by Microsoft®. Ignore this warning and click "Continue Anyway" (see Figure E-4).

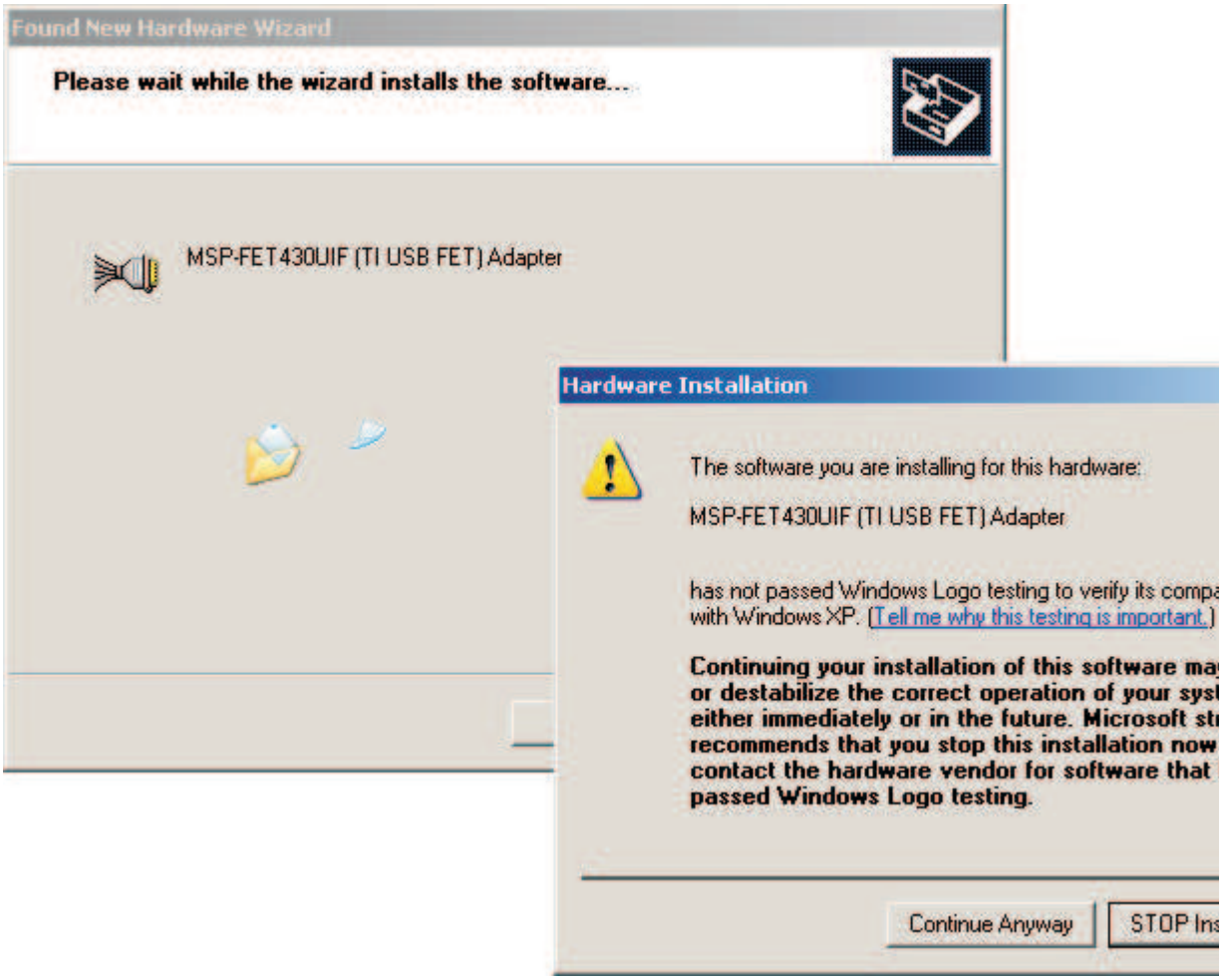


Figure E-4. WinXP Driver Installation

8. The Wizard installs the driver files.
9. The Wizard shows a message that it has finished the installation of the software for "MSP-FET430UIF (TI USB FET) Adapter".
10. After closing the Hardware Wizard, Windows automatically recognizes another new hardware device called "MSP-FET430UIF - Serial Port".
11. Depending on the current update version of the operating system, corresponding drivers are installed automatically or the Hardware Wizard opens again. **If the Wizard starts again, repeat the steps described above.**

12. The MSP-FET430UIF debug interface is installed and ready to use. The Device Manager lists a new entry as shown in Figure E-5.

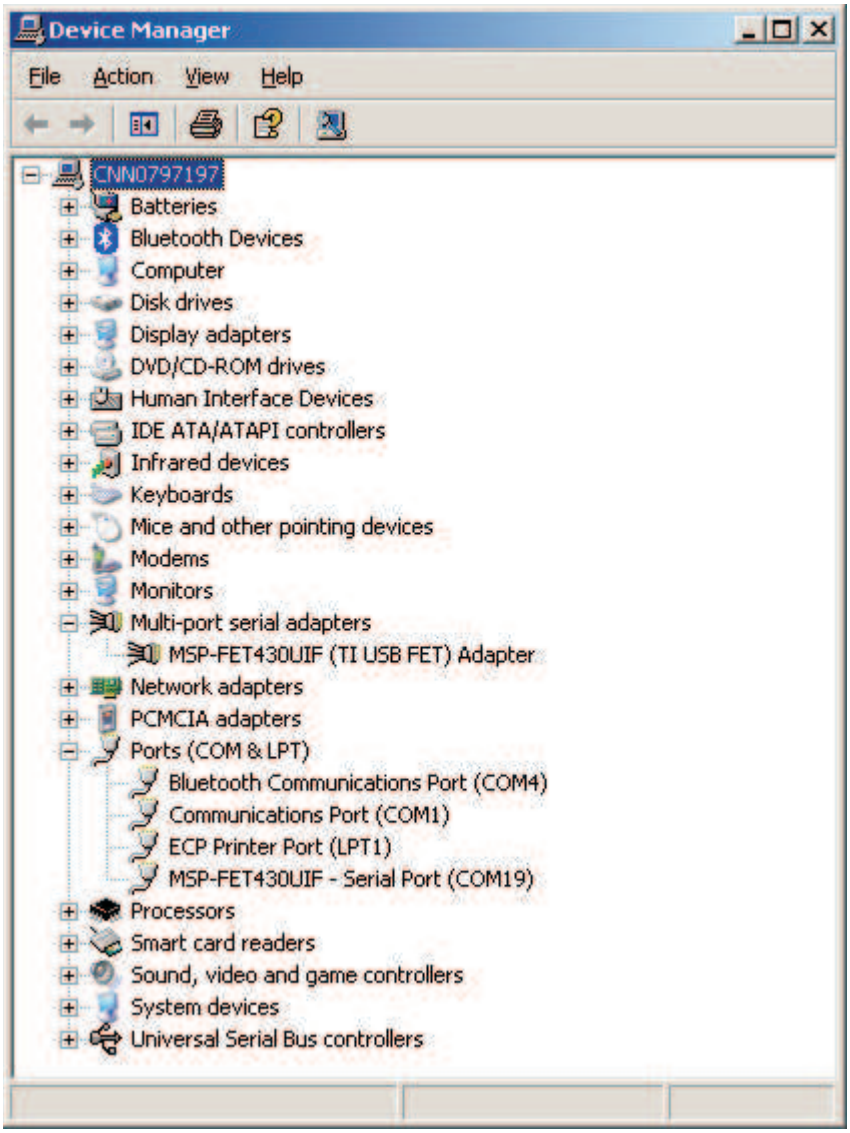


Figure E-5. Device Manager

Document Revision History

Version	Changes/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 IMPORTANT NOTICE

Texas Instruments (TI) provides the enclosed product(s) under the following conditions:

This evaluation board/kit 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. Persons handling the product(s) must have electronics training and observe good engineering practice standards. As such, the goods being provided are not intended to be complete in terms of required design–, marketing–, and/or manufacturing-related protective considerations, including product safety and environmental measures typically found in end products that incorporate such semiconductor components or circuit boards. This evaluation board/kit does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and therefore may not meet the technical requirements of these directives or other related directives.

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 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.

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. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge.

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.

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.

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 contact the TI application engineer or visit www.ti.com/esh.

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.

FCC WARNING

This evaluation board/kit 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 rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments 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.

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, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

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

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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