TEXAS INSTRUMENTS

MSP-FET430 FLASH Emulation Tool (FET)

(For use with IAR Workbench Version 3.x)

User's Guide

October 2006

SLAU138
Mixed Signal Products

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

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

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, license, warranty or endorsement thereof.

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. Representation or reproduction of this information with alteration voids all warranties provided for an associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Resale of TI's 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, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Also see: Standard Terms and Conditions of Sale for Semiconductor Products. www.ti.com/sc/docs/stdterms.htm

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

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

Preface

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 ultra low 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 Get Started Now! chapter. This chapter will enable you to inventory your FET, and then it will instruct you to install the software and hardware, and then run the demonstration programs. Once you've been demonstrated how quick and easy it is to use the FET, we suggest that you complete the reading of this manual.

This manual describes the set-up and operation of the FET, but does not fully teach the MSP430 or the development software systems. For details of these items, refer to the appropriate TI and IAR documents listed in Chapter 1.12 Important MSP430 Documents on the CD-ROM and WEB.

This manual is applicable 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)

Below 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 pin DW-Packages)

MSP-FET430U38 (for MSP430 devices in 38 pin DA-Packages)

MSP-FET430U40 (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)

This tool 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.

This is an example of a caution statement.

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

CAUTION

This is an example of a warning statement.

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

WARNING

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.

Contents

Figures

Tables

Chapter 1

Get Started Now!

This chapter will enable you to inventory your FET, and then it will instruct you to install the software and hardware, and then run the demonstration programs.

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.

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 2x7 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 2x7 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 2x7 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 2x7 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 2x7 pin male connector is also present on the PCB.

One 25-conductor cable.

One 14-conductor cable.

MSP-FET430P120: Four PCB 1x14 pin headers (Two male and two female).

MSP-FET430P140: Eight PCB 1x16 pin headers (Four male and four female).

MSP-FET430P410: Eight PCB 1x16 pin headers (Four male and four female).

MSP-FET430P430: Eight PCB 1x20 pin headers (Four male and four female).

MSP-FET430P440: Eight PCB 1x25 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 http://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, ‘U40, ‘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 2x7 pin male connector on the other end of the case.

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 2x7 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 2x7 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 2x7 pin male connector is also present on the PCB.

MSP-FET430U40: One 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 2x7 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 2x7 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 2x7 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 2x7 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 2x7 pin male connector is also present on the PCB.

One USB-Cable.

One 14-conductor cable.

MSP-FET430U14: Four PCB 1x7 pin headers (Two male and two female).

MSP-FET430U28: Four PCB 1x14 pin headers (Two male and two female).

MSP-FET430U38: Four PCB 1x19 pin headers (Two male and two female).

MSP-FET430U40: Eight PCB 1x10 pin headers (Four male and four female).

MSP-FET430U48: Four PCB 2x24 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-FET430U40: MSP430F2370IRHA
MSP-FET430U48: MSP430F4270IDL
MSP-FET430U64: MSP430F417IPM and MSP430F169IPM
MSP-FET430U80: MSP430FG439
MSP-FET430U100: 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 http://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 is used to refer 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 above documents (and this document) can be accessed using: START--> PROGRAMS--> IAR SYSTEMS--> IAR EMBEDDED WORKBENCH KICKSTART FOR MSP430 V3

Kickstart is compatible with Windows 98, 2000, ME, NT4.0, and XP. However, the USB-FET-Interface works only with Windows 2000 and 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 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) 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. Pictorials 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 SW. **If for any reason the Install Wizard starts, respond to the prompts and point the wizard to the driver files which are located in directory: <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 where 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} (1.8V–3.6V); target can be supplied with up to 100 mA.

1.10 Hardware Installation, MSP-FET430Uxx (‘U14, ‘U28, ‘U38, ‘U40, ‘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 V_{CC}) near the 2x7 pin male connector are in place. Pictorials of the Target Socket module and its parts are presented in Appendix B.

1.11 “Flash”ing 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) Use FILE--> OPEN WORKSPACE to open the file at: <Installation Root>\Embedded Workbench x.x\430\FET_examples\fet_projects.eww. The workspace window will open.
3) Click on the tab at the bottom of the workspace window that corresponds to your tool (FETxxx) and desired language (assembler or C).
4) Use PROJECT--> OPTIONS--> FET Debugger--> Setup--> Connection to select the appropriate port: LPTx for the parallel FET Interface or TI USB FET for the USB Interface or for the eZ430.
5) Use 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) Use 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.

   Refer to FAQ, Debugging #1) if C-SPY is unable to communicate with the device.
7) Use DEBUG--> GO to start the application. The LED should flash!
8) Use DEBUG--> STOP DEBUGGING to stop debugging, to exit C-SPY, and to return to the Workbench.
9) Use 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 available at the time of production have been provided on the CD-ROM included with this tool. The MSP430 web site (www.ti.com/msp430) will contain the latest version of these documents.

From the MSP430 main page on the CD-ROM, navigate to: Literature--> MSP430 Literature--> Data Sheets, to access the MSP430 device data sheets.

From the MSP430 main page on the CD-ROM, navigate to: Literature--> MSP430 Literature--> User's Guides, to access the MSP430 device User's Guides and tools.

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

Chapter 2

Development Flow

This chapter discusses how to use Kickstart to develop your 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 on-line via HELP.

Read Me Firsts 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 will not generate an assembly code list file.

The linker will link a maximum of 4K bytes of code originating from C source (but an unlimited amount of code originating from assembler source).

The simulator will input 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. Please 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 will be applied to the selected file (and 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)

In order to most easily debug a C project, disable optimization (C/C++ COMPILER--> CODE--> OPTIMIZATIONS--> SIZE--> NONE (BEST DEBUG SUPPORT))

Enable the generation of debug information in the compiler output (C/C++ COMPILER --> OUTPUT--> GENERATE DEBUG INFO)

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)

In order to debug the project using C-SPY, specify a compatible format (LINKER--> OUTPUT--> FORMAT--> DEBUG INFO [WITH TERMINAL IO])

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: LPTx for the parallel FET Interface or TI USB FET for the USB Interface.

Enable the Device Description file. This file makes C-SPY “aware” of the specifics of the device it is debugging. This file will correspond to the specified target device (DEBUGGER--> SETUP--> DEVICE DESCRIPTION--> OVERRIDE DEFAULT)

Enable the erasure of the Main and Information memories before object code download (FET DEBUGGER--> SETUP--> DOWNLOAD CONTROL--> ERASE MAIN AND INFORMATION MEMORY)

In order to maximize system performance during debug, disable Virtual Breakpoints (FET DEBUGGER--> SETUP --> USE VIRTUAL BREAKPOINTS), and disable all System Breakpoints (FET DEBUGGER--> SETUP --> SYSTEM BREAKPOINTS ON)

Note: Use of Factory Settings to quickly configure a project

It is possible to 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: 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) To debug on the hardware, select DEBUGGER --> SETUP--> DRIVER--> FET DEBUGGER
9) Specify the active parallel port used to interface to the FET if not LPT1 (FET DEBUGGER --> SETUP--> CONNECTION--> LPTx) or specify the USB port (FET DEBUGGER --> SETUP--> CONNECTION--> TI USB FET)

Note: Avoid the use of absolute pathnames when referencing files.

Instead, use the relative pathname keywords $TOOLKIT_DIR$ and $PROJ_DIR$. Refer to the IAR documentation for a description of these keywords. The use of relative pathnames will permit projects to be moved easily, and projects will not require modification when IAR systems are upgraded (say, from Kickstart, or Baseline, to Full).

2.2.2 Creating a Project from Scratch

The following 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. Refer to Project Settings above. Also, the MSP430 IAR Embedded Workbench IDE User Guide presents a more comprehensive overview of the process.

1) Start the Workbench (START--> PROGRAMS--> IAR SYSTEMS--> IAR EMBEDDED WORKBENCH KICKSTART FOR MSP430 V3--> KICKSTART IAR EMBEDDED WORKBENCH).

2) Create a new text file (FILE--> NEW--> SOURCE/TEXT).

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. Simply 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 text file (FILE--> SAVE).

   It is recommended that assembler text file 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). Specify a workspace name and press SAVE.

6) Create a new project (PROJECT--> CREATE NEW PROJECT). Specify a project name and press CREATE

7) Add the text file to the project (PROJECT--> ADD FILES). Select the text file and press OPEN. Alternatively, double-click on the text 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”. In order to view assembler files (.s43), select “Assembler Files” in the “Files of type” drop-down menu.

8) Configure the project options (PROJECT--> OPTIONS). For each of the listed 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), select DEBUGGER --> SETUP--> DRIVER--> FET DEBUGGER

   Specify the active port used to interface to the FET (FET DEBUGGER --> SETUP--> CONNECTION)

8) Build the project (PROJECT--> REBUILD ALL).

9) Debug the application using C-SPY (PROJECT--> DEBUG). This will start C-SPY, and C-SPY will get control of the target, erase the target memory, program the target memory with the application, and reset the target.

   Refer to FAQ, Debugging #1) if C-SPY is unable to communicate with the device.

10) Use DEBUG--> GO to start the application.

11) Use DEBUG--> STOP DEBUGGING to stop the application, to exit C-SPY, and to return to the Workbench.

12) Use 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; refer to the IAR document *Step by step migration for EW430 x.xx*. This document can be located 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). Refer to 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:

1. **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).**

2. **There is no runtime checking of the stack. The stack can overflow the 50 reserved bytes and possible overwrite the other segments. No error will be output.**

The supplied .xcl files can be easily modified to tune the size of the stack to the needs of the application; simply edit -D_STACK_SIZE=xx to allocate xx bytes for the stack. Note that the .xcl file will also reserve 50 byes for the heap if required (say, 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. Select: PROJECT--> OPTIONS--> LINKER--> OUTPUT--> FORMAT--> OTHER--> MSP430-TXT. Intel and Motorola formats can also be selected.

Refer to 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\fet_projects.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 FET tools. Debug and Release versions are provided for each of the projects.

<Installation Root>\Embedded Workbench x.x\430\FET_examples\code_examples.eww conveniently organizes the code examples into a workspace. The workspace contains assembler and C projects of the code for each of the FET tools. 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 Design Resources.

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

Refer to 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, refer to Table 2-1 below). 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--> SETUP--> 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 breakpoint (or “Non-Realtime”). During Non-Realtime mode, the PC effectively 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 JTAG†	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		

† The 2-wire JTAG debug interface is also referred to as Spy-Bi-Wire interface

2.3.2 Using Breakpoints

If C-SPY is started with greater than N breakpoints set and virtual breakpoints are disabled, a message will be output that informs 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.

RESET'ing a program temporarily requires a breakpoint if PROJECT--> OPTIONS--> DEBUGGER--> SETUP--> RUN TO is enabled. Refer to FAQ, Debugging #31).

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. Refer to FAQ, Debugging #32).

If, while processing a breakpoint, an interrupt becomes active, C-SPY will stop at the first instruction of the interrupt service routine. Refer to FAQ, Debugging #25).

2.3.3 Using Single Step

When debugging an assembler file, STEP OVER, STEP OUT, and NEXT STATEMENT operate like STEP INTO; 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 CALL'ed function.

When debugging an assembler file, a (true) STEP OVER a CALL instruction that executes the CALL'ed function at full device speed can be synthesized by placing a breakpoint after the CALL and GO'ing (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 will be placed after the function reference and a GO will be implicitly executed. This will cause the function to be executed at full speed. If no hardware breakpoints are available, the function will be 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 will place – if possible - a hardware breakpoint after the CALL instruction, and then execute GO. The CALL'ed function will execute at full device speed. If no hardware breakpoint is available prior to the GO, the CALL'ed function will be executed in Non-Realtime mode. In either case, execution will stop 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 will stop at the first instruction of the interrupt service routine. Refer to FAQ, Debugging #25).

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, say:

```
 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) Use 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) Press the Add Watch button

6) Close the Quick Watch window

7) For the created entry in the Watch Window, click on the + symbol. This will 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-n-dropping then 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. Refer to FAQ, Debugging #23).

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 (say, following a breakpoint hit, a single step, or a STOP/escape).

Although registers can be monitored in the Watch Window, VIEW--> REGISTER is a superior method.

Chapter 3

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, you can use the C-SPY debugger and an FET hardware JTAG interface such as the MSP-FET430PIF and MSP-FET430UIF to program and debug code on your own target board. In addition, the connections will also support the GANG430 or PRGS430 production programmers, thus providing 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 JTAG mode (Spy-Bi-Wire). While 4-wire JTAG mode is generally supported on all MSP430 devices, 2-wire JTAG mode is available on selected devices only. Refer to Table 2-1 above 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}, then 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 which 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 (Figure 3-1), the connection of the target RST signal to the JTAG connector is optional and not required when using 4-Wire JTAG communication mode capable-only devices. However, when using 2-Wire JTAG communication mode capable devices in 4-Wire JTAG mode, the RST connection must be made. The MSP430 development tools and device programmers perform a target reset through issuing a JTAG command to gain control over the device. However, in the case this should be unsuccessful, the RST signal of the JTAG connector may be used by the development tool or device programmer as an additional way to assert a device reset.

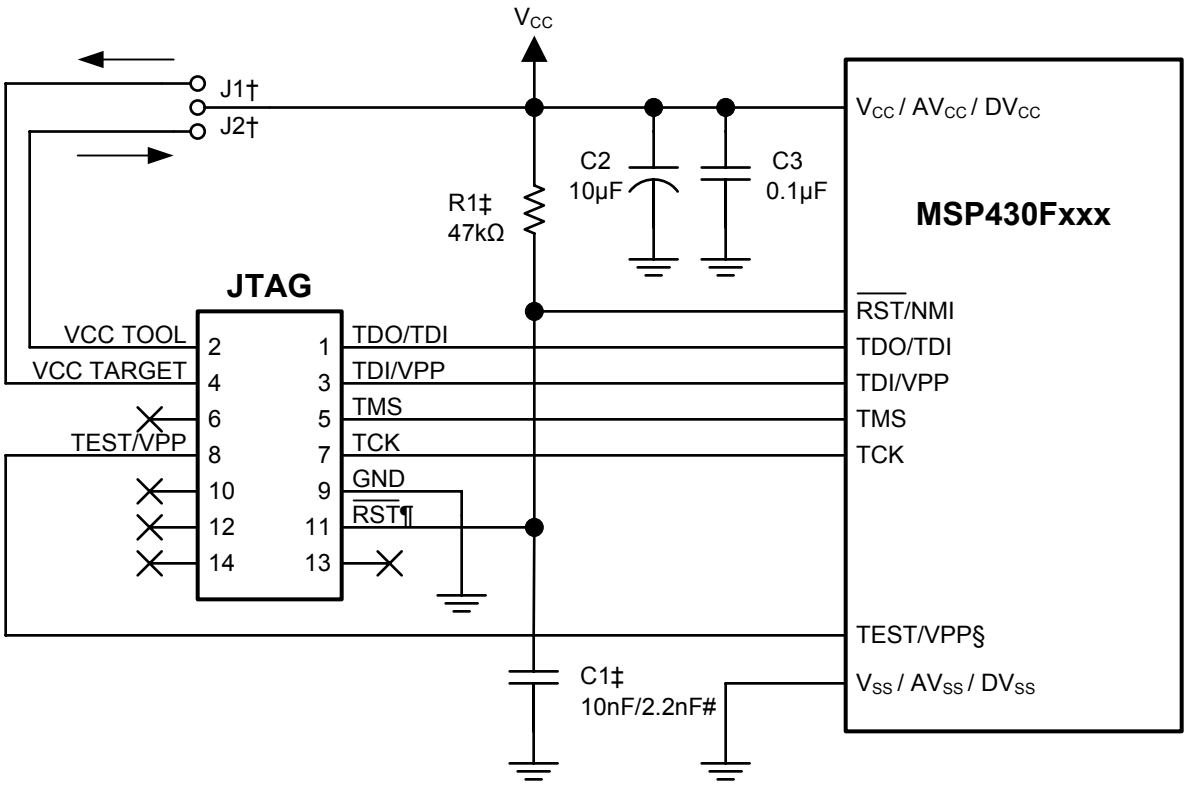


Figure 3-1. Signal Connections for 4-Wire JTAG Communication

† Make either connection J1 in case a local target power supply is used OR connection J2 to power target from the debug/programming adapter.

‡ The RST/NMI pin R1/C1 configuration is device family dependent. Refer to the respective MSP430 Family User's Guide for the recommended configuration.

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

¶ The connection to the JTAG connector RST pin is optional when using 4-Wire JTAG communication mode capable-only devices and not required for device programming or debugging. However, this connection is required when using 2-Wire JTAG communication mode capable devices in 4-Wire JTAG mode.

When using 2-Wire JTAG 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).

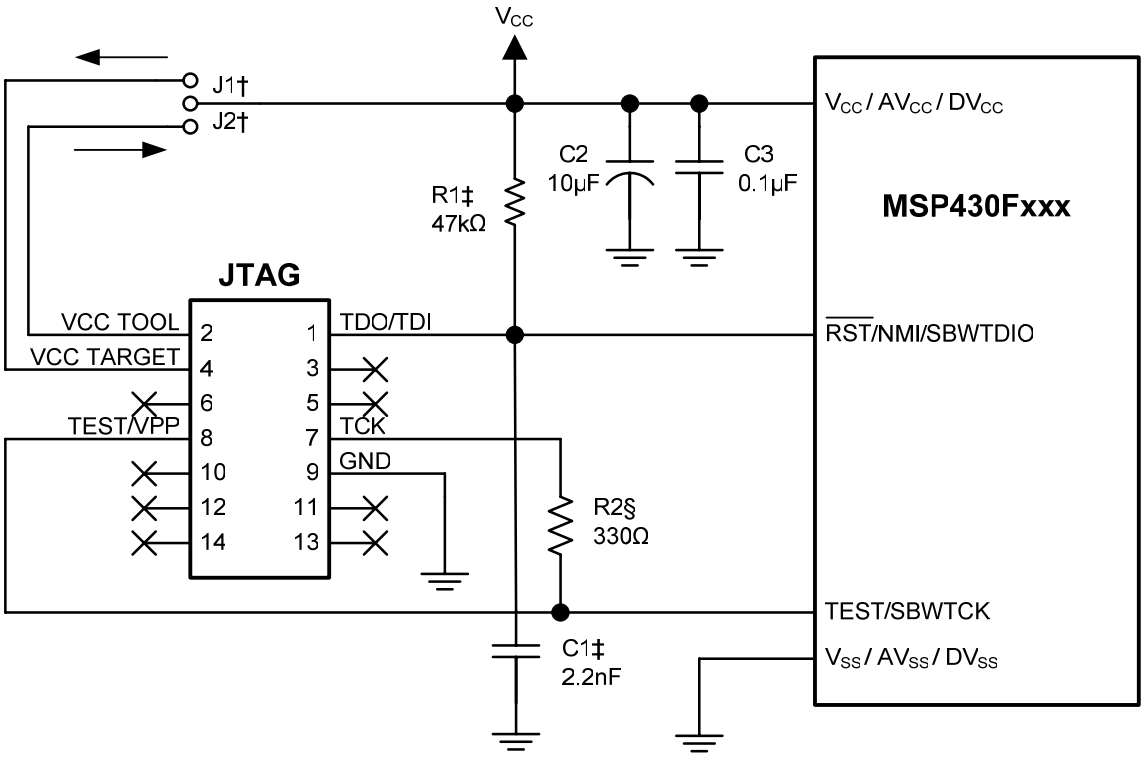


Figure 3-2. Signal Connections for 2-Wire JTAG Communication (Spy-Bi-Wire)

† Make either connection J1 in case a local target power supply is used OR connection J2 to power target from the debug/programming adapter.

‡ **Note that the device RST/NMI/SBWTDIO pin is used in 2-wire mode for bi-directional communication with the device during JTAG access 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).**

§ 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) and the connection TEST/VPP must not be made.

3.2 External Power

The PC parallel port can only source a limited amount of current. Owing to the ultra low power capability of the MSP430, a stand-alone 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. Refer to the schematics and pictorials of the MSP-FET430X110 and the Target Socket modules presented in 38) 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 1.8V and 5.0V in steps of 0.1V. 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 adjusts the level of the JTAG signals to external V_{CC} automatically. Only pin 2 (MSP-

FET430UIF supplies target) OR pin 4 (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). Again, refer to the Target Socket module schematics in 38).

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 Notes 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 Notes, or BSL tools can be purchased from 3rd parties. Refer to the MSP430 web site for the Application Notes and a list of MSP430 3rd party tool developers.

Texas Instruments suggests that MSP430Fxxx customers design their circuits with the BSL in mind (i.e., we suggest providing access to these signals, e.g. via a header).

Refer to FAQ, Hardware #9) for a second alternative to sharing the JTAG and port pins.

The BSL tool requires the following device signals:

RST/NMI
TEST†
TCK†
GND
VCC
P1.1
P2.2 or P1.0‡

† If present on device.

‡ ‘1xx / ‘2xx devices use pins P1.1 and P2.2 for the BSL. ‘4xx devices use pins P1.0 and P1.1 for the BSL.

Appendix A

Frequently Asked Questions

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

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) 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.

3) The 14-conductor **cable** connecting the FET Interface module and the Target Socket module **must not exceed 8 inches (20 centimeters) in length.**

4) The signal assignment on the **14-conductor cable** is **identical** for the **parallel port interface** and the **USB FET**.

5) **To utilize the on-chip ADC voltage references, C6** (10uF, 6.3V, low leakage) **must be installed** on the Target Socket module.

6) **Crystals/resonators Q1 and Q2** (if applicable) **are not provided** on the Target Socket module. For MSP430 devices which contain user selectable loading capacitors, the effective capacitance is the selected capacitance plus 3pF (pad capacitance) divided by two.

7) **Crystals/resonators have no effect upon the operation of the tool and C-SPY** (as any required clocking/timing is derived from the internal DCO/FLL).

8) **On 20-pin and 28-pin devices** with multiplexed port/JTAG pins (P1.4-P1.7), **it is required that "RELEASE JTAG ON GO" be selected in order to use these pins in their port capacity.** Refer to C.1.2 EMULATOR--> RELEASE JTAG ON GO for additional information regarding this mechanism.

9) **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 (say, without shared JTAG and port pins) will port effortlessly to another (assuming an equivalent set of peripherals).

10) **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.

11) **The device current increases by approximately 10uA 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.

12) 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: http://www.enplas.com

    Wells-CTI: http://www.wellscti.com/

    Yamaichi: http://www.yamaichi.us/

13) **Supply current measurement on Target Socket modules.** On each module a jumper connects Vcc with Vcc430. If this jumper is removed and a ampere-meter is connected to the jumper pins, the supply current of the module can be measured. As the pull-up resistor (47k) on the Reset-line is connected to Vcc, the MSP430 device sees a marginal voltage at pin RST/NMI if Vcc is present and the jumper is open. Therefore Vcc 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. Refer to 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 will reset the device if not disabled or properly handled by your application. Refer to FAQ, Program Development #14).

3) **When adding source files to a project, do not add files that are #include’ed by source files that have already been added to the project** (say, an .h file within a .c or .s43 file). These files will be 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 will not start otherwise, and an error message will be 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.** Contact TI if you wish the source code for these libraries so that this behavior can be disabled.

9) **It is possible to mix assembler and C programs within the Workbench.** Refer to the Assembler Language Interface chapter of the C/C++ Compiler Reference Guide from IAR.

10) The Workbench can produce an object file in Texas Instruments .TXT format. **C-SPY cannot input an object file in Texas Instruments .TXT format**. An error message will be output in this case.

11) **The example programs giving in the Kickstart documentation (i.e., Demo, Tutor, etc.) are not correct.** The programs will work only in the simulator. However, the programs will 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 the C-statement: “WDTCTL = WDTPW + WDTHOLD;”, or “mov.w # WDTPW+WDTHOLD,&WDTCTL” in assembler.

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 a good thing 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 will go 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 can be located in: <Installation Root>\Embedded Workbench x.x\430\doc\migration.htm

18) **Assembler projects must reference the code segment** (RSEG CODE) in order **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 will fail and a linker error will be 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
    Refer to 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--> 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). Refer to FAQ, Debugging #7) later in this document. For users of IBM Thinkpads, please try port specifications LPT2 and LPT3 despite the fact that 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 (say, printer drivers, ZIP drive drivers, etc.) in the case 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 with using 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 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.

4) **C-SPY can debug applications that utilize interrupts and low power modes**. Refer to FAQ, Debugging #25).

5) **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 in order to access device registers and memory. Any displayed register/memory fields will then be updated.

6) **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.

7) **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. Refer FAQ, Debugging #1) for additional hints on solving communication problems between C-SPY and the device.

8) **C-SPY may assert 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 RST/NMI is not asserted (low), C-SPY sets the logic driving RST/NMI to high-impedance, and RST/NMI is pulled high via a resistor on the PCB.

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

   Within C-SPY, EMULATOR--> "POWER ON" RESET will cycle the power to the target to generate a power-on reset.

9) **C-SPY can debug a device whose program reconfigures the function of the RST/NMI pin to NMI.**

10) **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.

11) **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. Refer to FAQ, Debugging #13) and Hardware #11).

12) **Most C-SPY settings** (breakpoints, etc.) **are preserved between sessions.**

13) **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. Refer to FAQ, Debugging #11) and Hardware #11).

14) 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.

15) 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.

16) **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.

17) 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 only be used in the address ranges as specified by the device data sheet.

18) 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 effected 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 which support Clock Control (EMULATOR--> ADVANCED--> CLOCK CONTROL) can further minimize these effects by selecting to stop the clock(s) during debugging.

    Refer to FAQ, Debugging #23).

19) **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 INITNEW 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.

20) When programming the Flash, **do not set a breakpoint on the instruction immediately following the write to Flash operation.** A simple work-around to this limitation is to follow the write to Flash operation with a NOP, and set a breakpoint on the instruction following the NOP. Refer to FAQ, Debugging #22).

21) The **Dump Memory length specifier is restricted to four hexadecimal digits (0-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).

22) 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 will update its memory window with erroneous information.** A work around 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. Refer to FAQ, Debugging #20).

23) **Peripheral bits that are cleared when read during normal program execution** (i.e., Interrupt Flags) **will be 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).

24) **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.

25) **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 will always process an active and enabled interrupt before processing non-ISR (i.e., mainline) code. A work-around for this behavior is, while within the ISR, to disable the GIE bit *on the stack* so that interrupts will be disabled after exiting the ISR. This will permit the non-ISR code to be debugged (but without interrupts). Interrupts can later be re-enabled 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).

26) **The base** (decimal, hexadecimal, etc.) **property of Watch Window variables is not preserved between C-SPY sessions**; the base reverts to Default Format.

27) On devices equipped with a Data Transfer Controller (DTC), **the completion of a data transfer cycle will preempt a single step of a low power mode instruction**. The device will advance beyond the low power mode instruction only after an interrupt is processed. Until an interrupt is processed, it will appear that the single step has no effect. A work around to this situation is to set a breakpoint on the instruction following the low power more instruction, and then execute (GO) to this breakpoint.

28) **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.

29) 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.

30) **It's 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.

31) RESET'ing a program temporarily requires a breakpoint if PROJECT--> OPTIONS--> DEBUGGER--> SETUP--> RUN TO is enabled. If N or more breakpoints are set, RESET will set a virtual breakpoint and will run 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 will indicate that the program is running, and C-SPY windows may be blank (or may not be correctly updated).

32) RUN TO CURSOR temporarily requires a breakpoint. If N breakpoints are set and virtual breakpoints are disabled, **RUN TO CURSOR will incorrectly use a virtual breakpoint.** This results in very slow program execution.

33) **The simulator is a CPU core simulator only**; peripherals are not simulated, and interrupts are statistical events.

34) 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.** Refer to the C-SPY documentation for a description of this complex breakpoint mechanism.

35) **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 device.

36) **Special Function Registers** (SFRs) and the peripheral registers **are displayed in VIEW--> REGISTER**.

37) **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.

38) **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 in order to maximize programming efficiency. Consequently, it’s possible for, say, a 60K byte program to be reduced to a single chunk, and the progress bar will not be updated until the entire write operation is complete.

Appendix B

Hardware

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

Topic **Page**

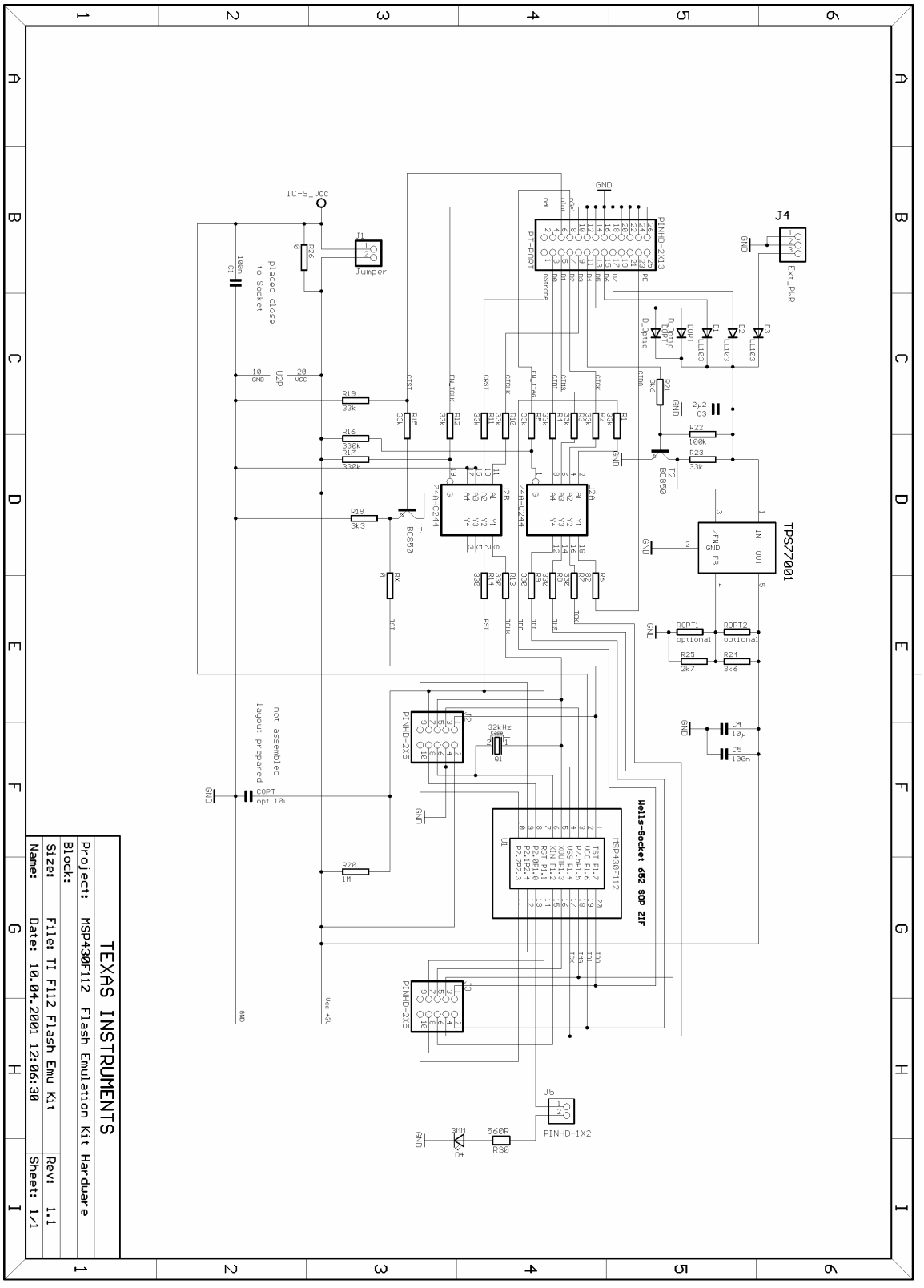
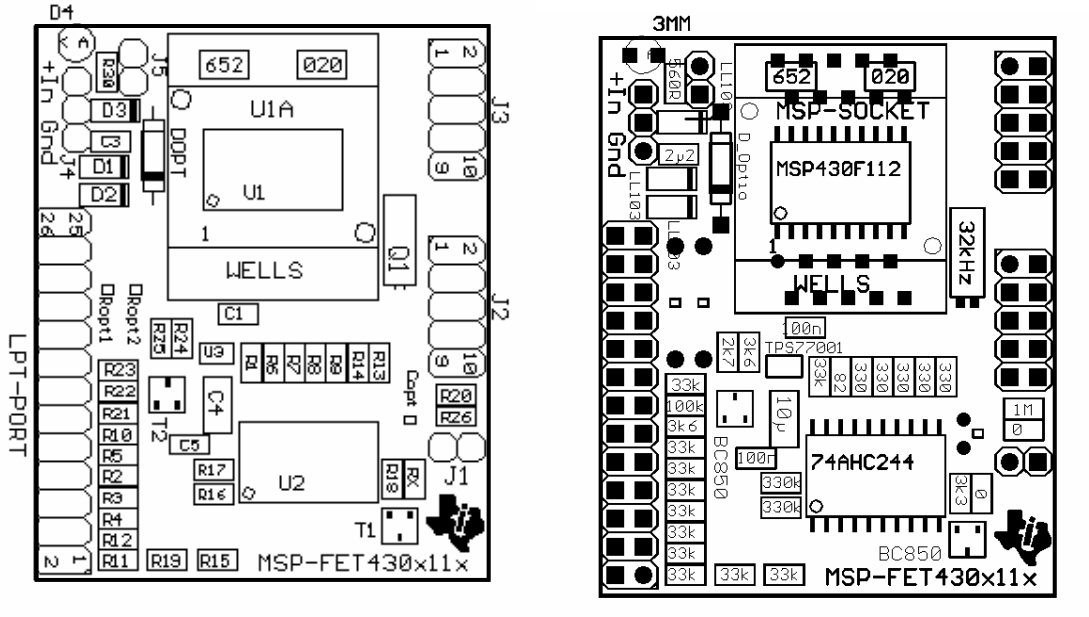
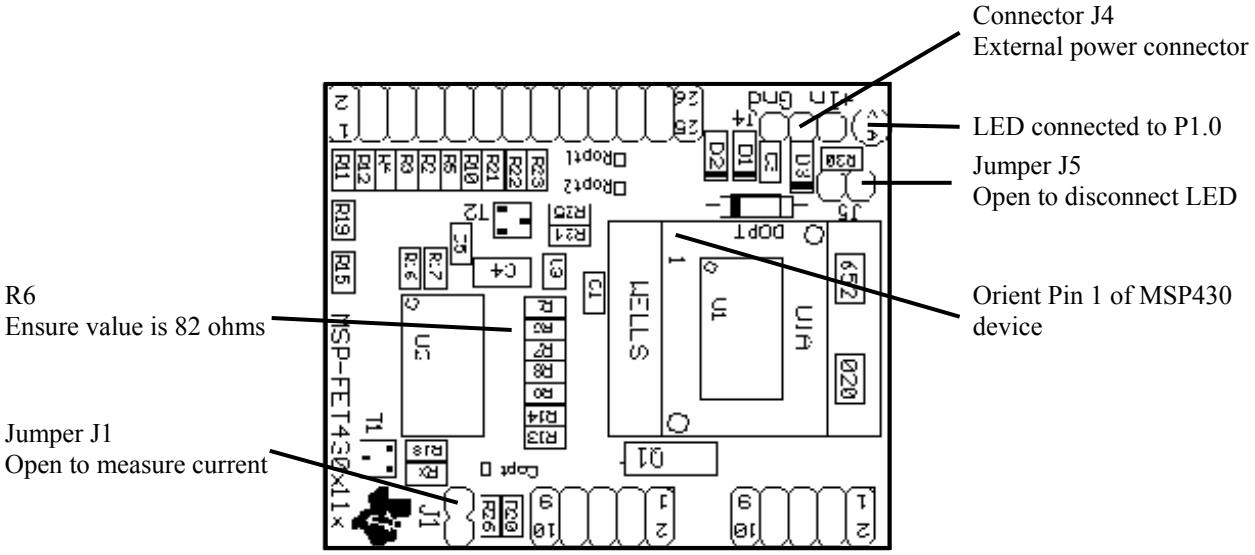


Figure B-1. MSP-FET430X110, Schematic





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 Pictorials

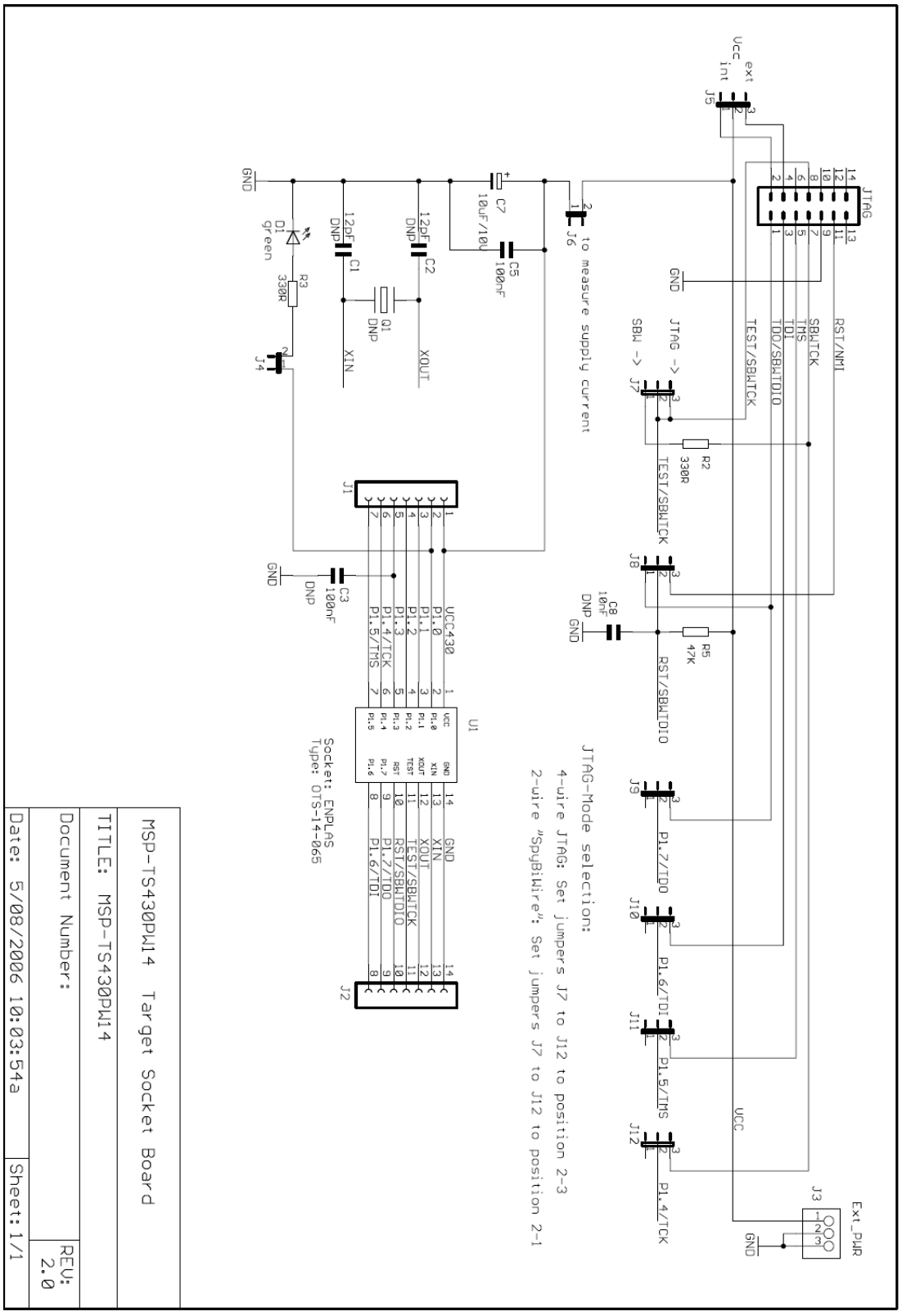


Figure B-3. MSP-TS430PW14 Target Socket module, Schematic

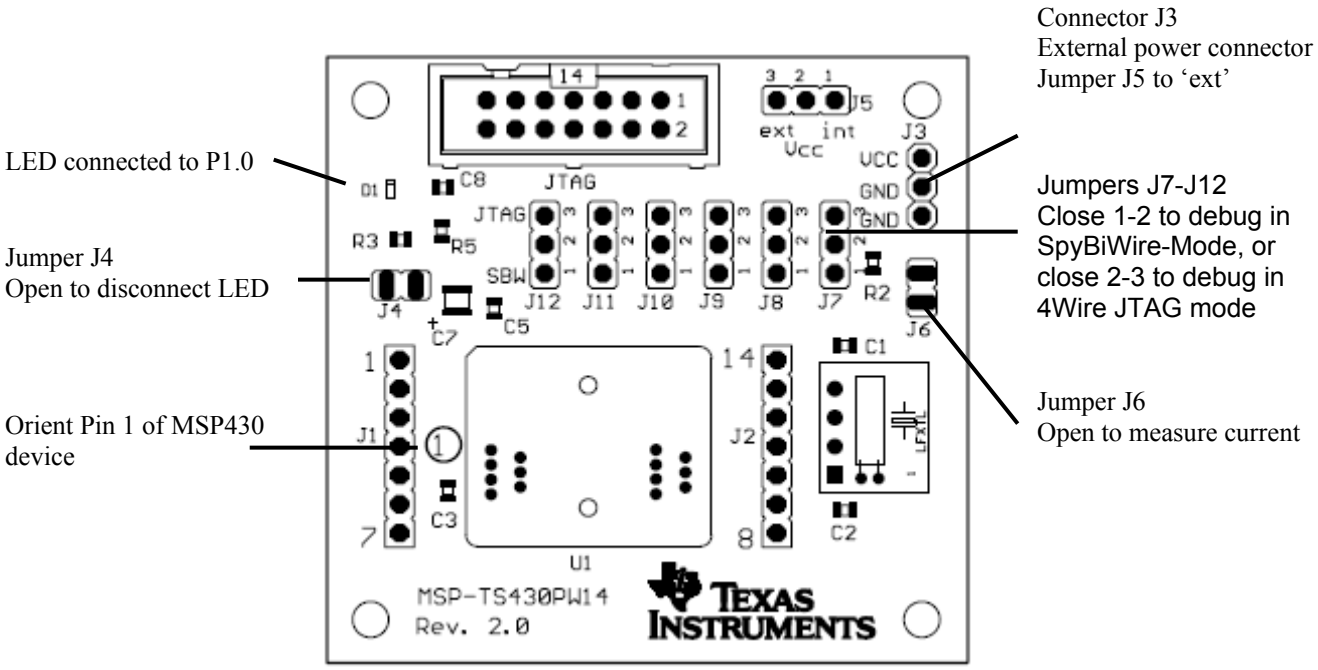
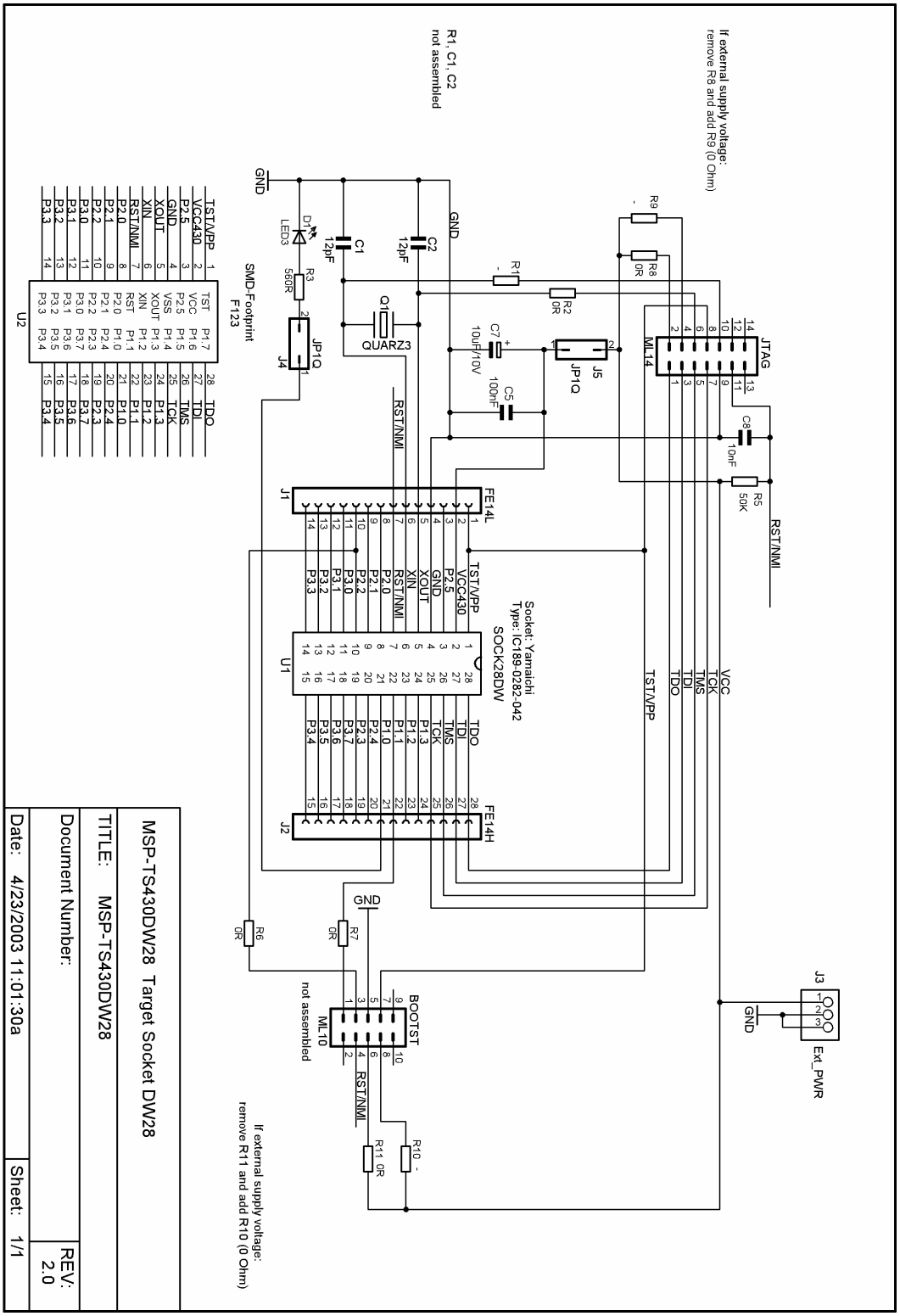


Figure B-4. MSP-TS430PW14 Target Socket module, PCB Pictorials



Note: Connections between the JTAG header and pins XOUT and XIN are no longer required, and should not be made.

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

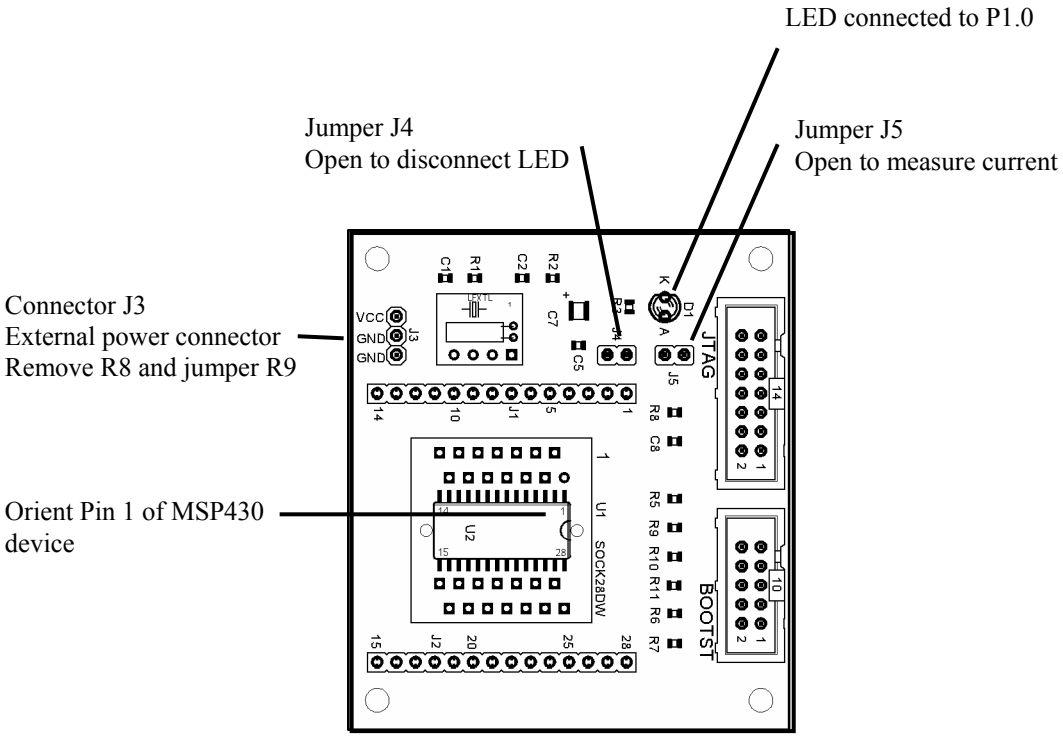


Figure B-6. MSP-TS430DW28 Target Socket module, PCB Pictorials

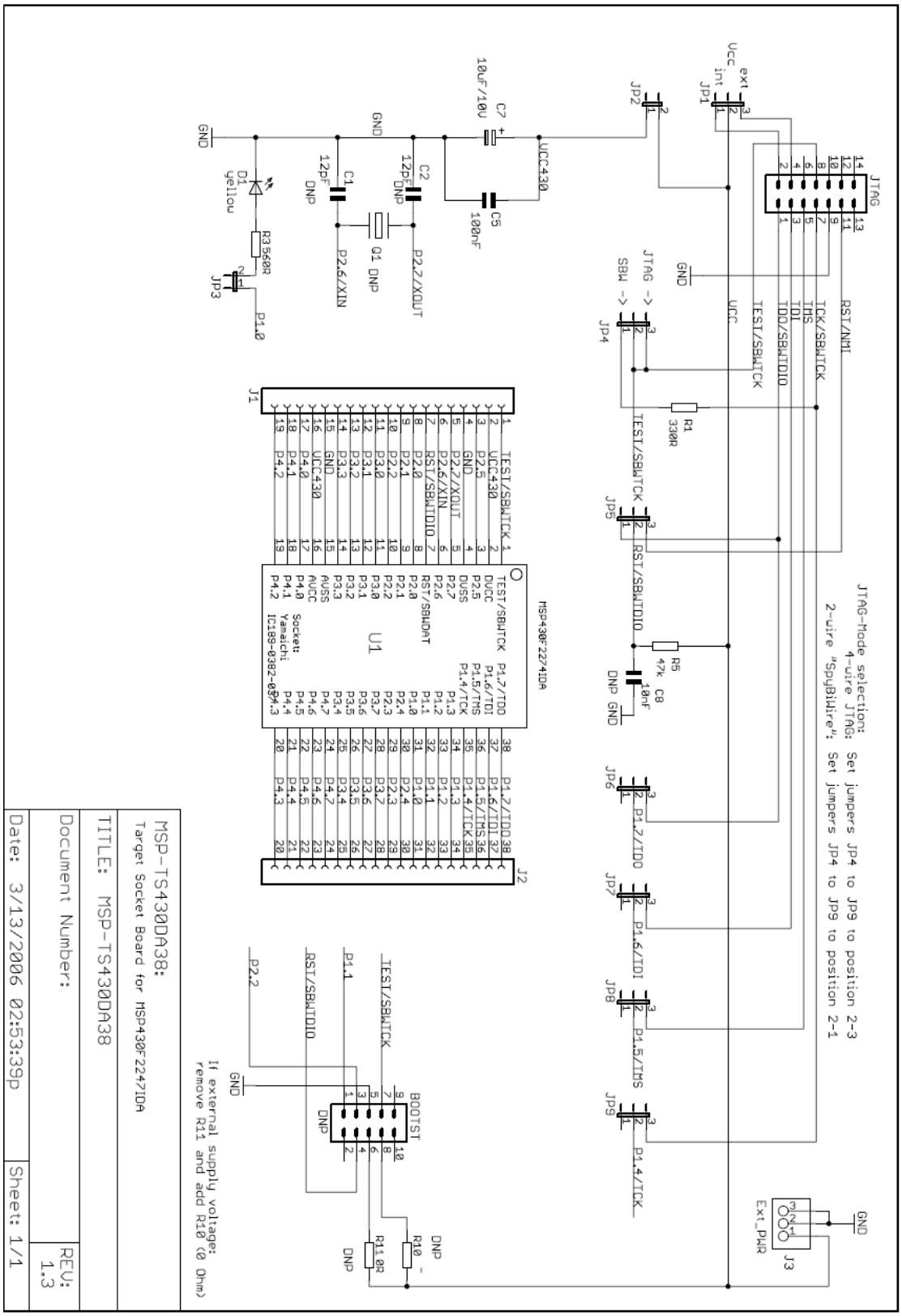


Figure B-7. MSP-TS430DA38 Target Socket module, Schematic

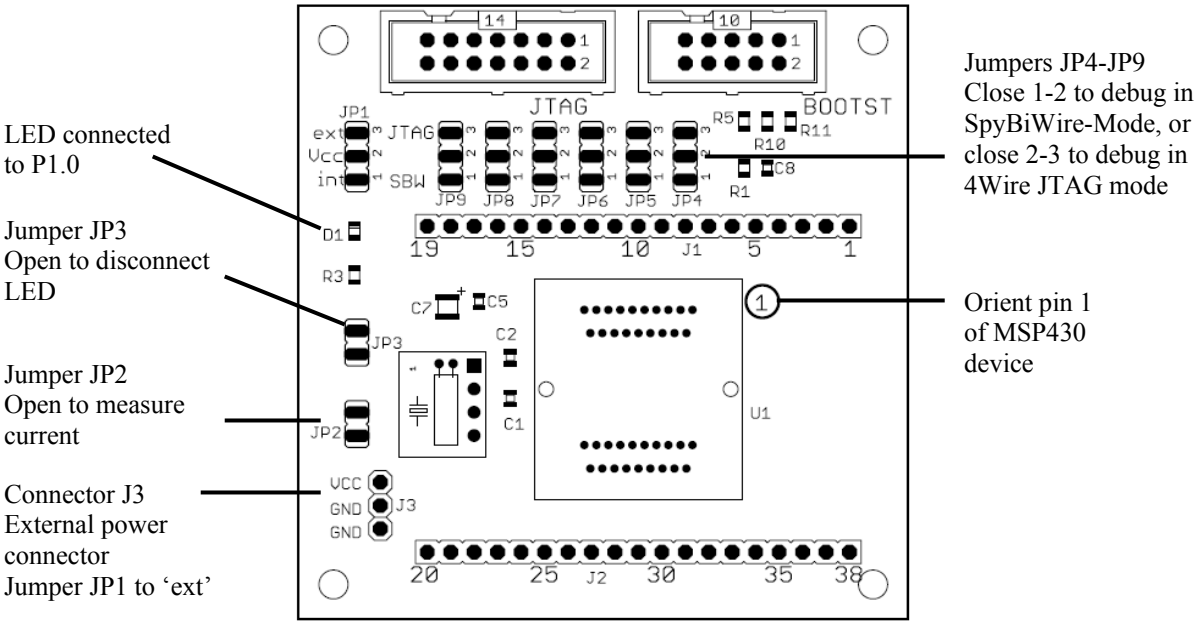


Figure B-8. MSP-TS430DA38 Target Socket module, PCB Pictorials

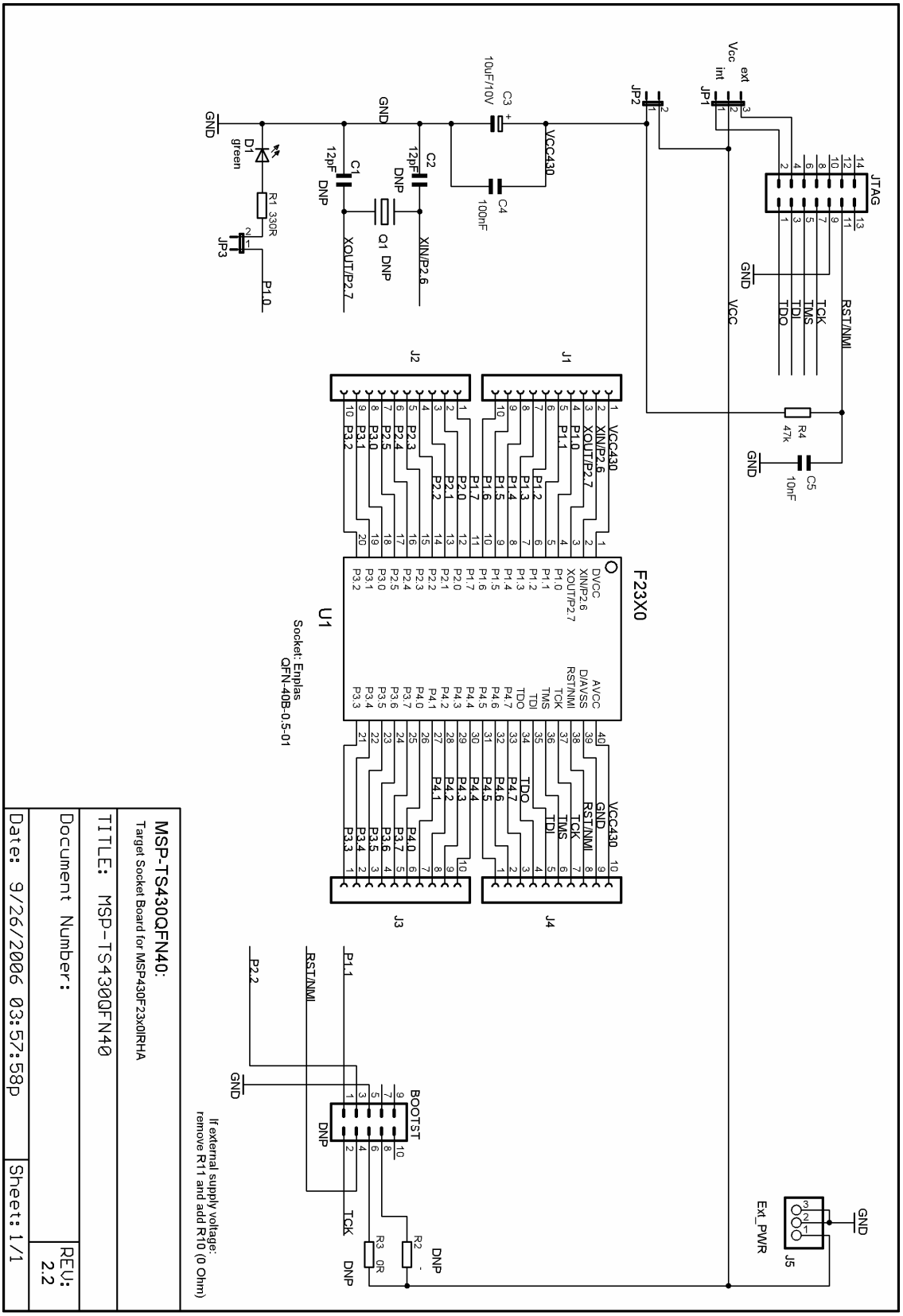


Figure B-9. MSP-TS430QFN40 Target Socket module, Schematic

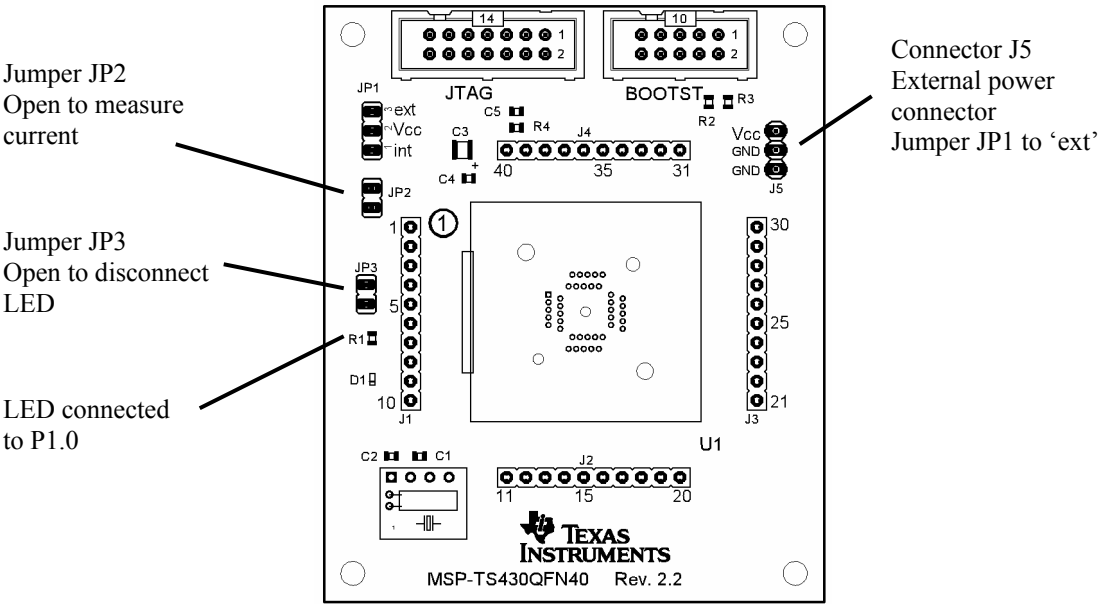


Figure B-10. MSP-TS430QFN40 Target Socket module, PCB Pictorials

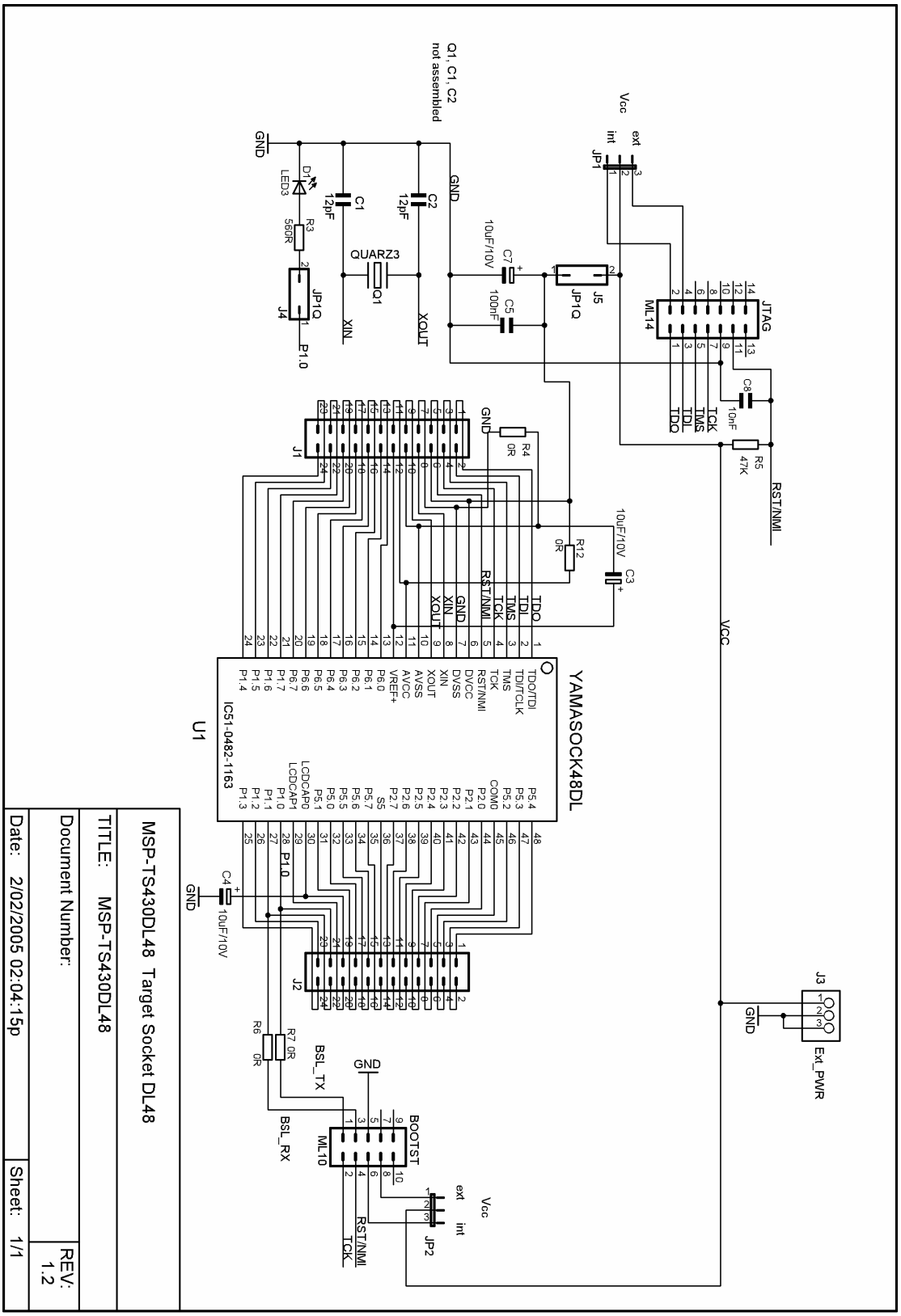


Figure B-11. MSP-TS430DL48 Target Socket module, Schematic

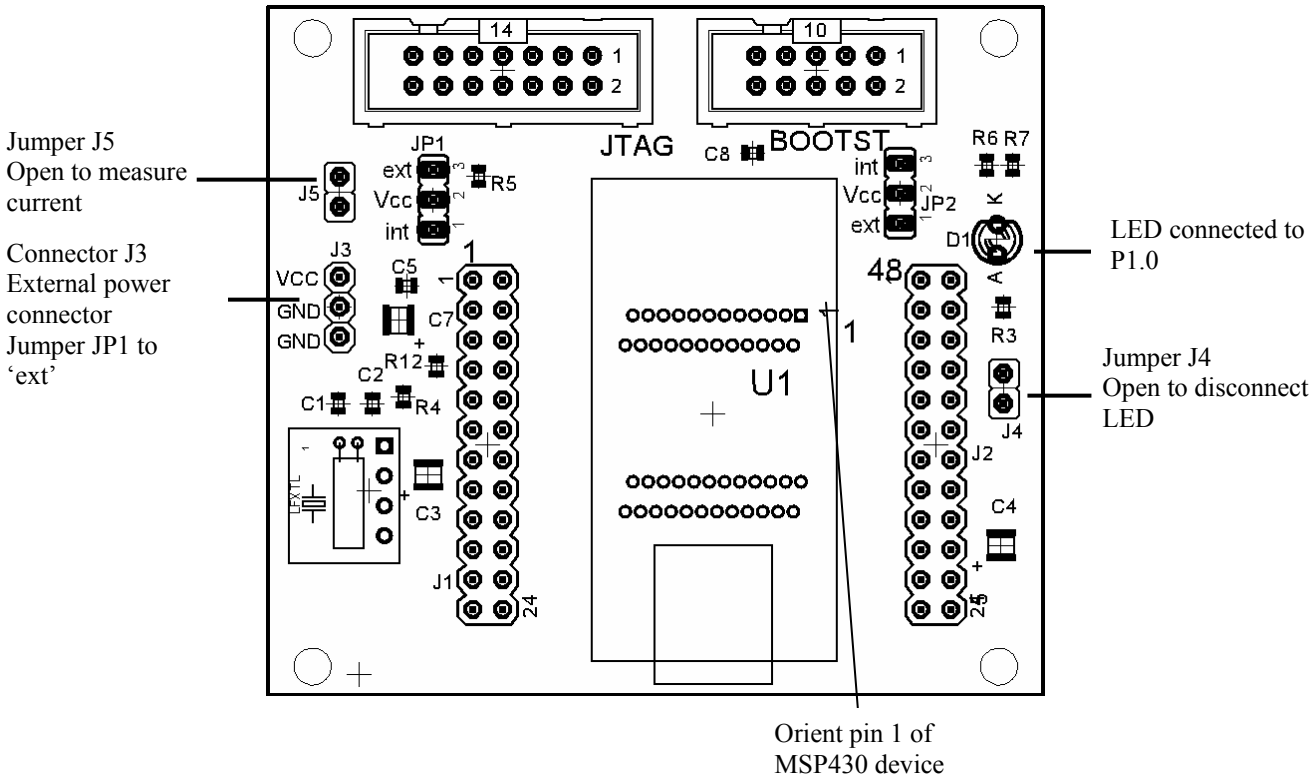
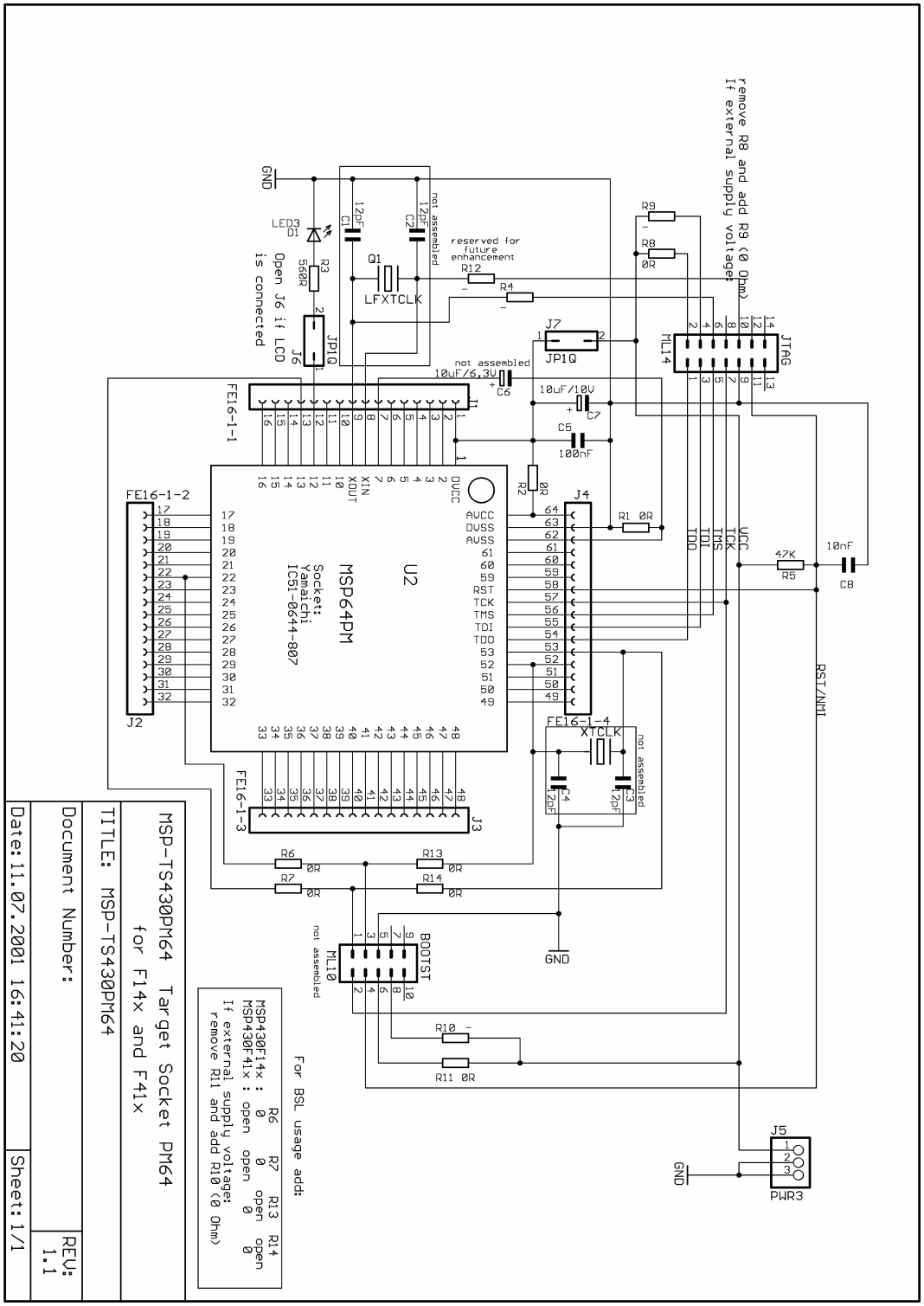


Figure B-12. MSP-TS430DL48 Target Socket module, PCB



Note: Connections between the JTAG header and pins XOUT and XIN are no longer required, and should not be made.

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

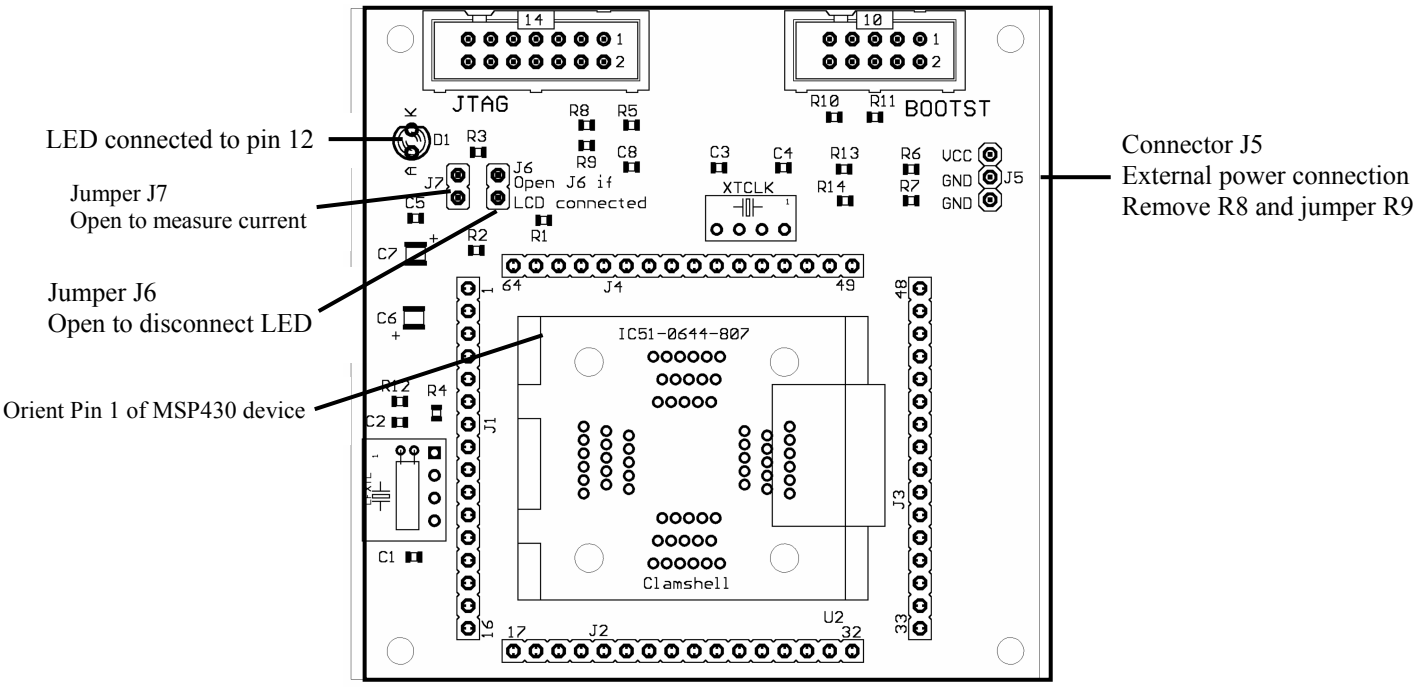


Figure B-14. MSP-TS430PM64 Target Socket module, PCB Pictorials

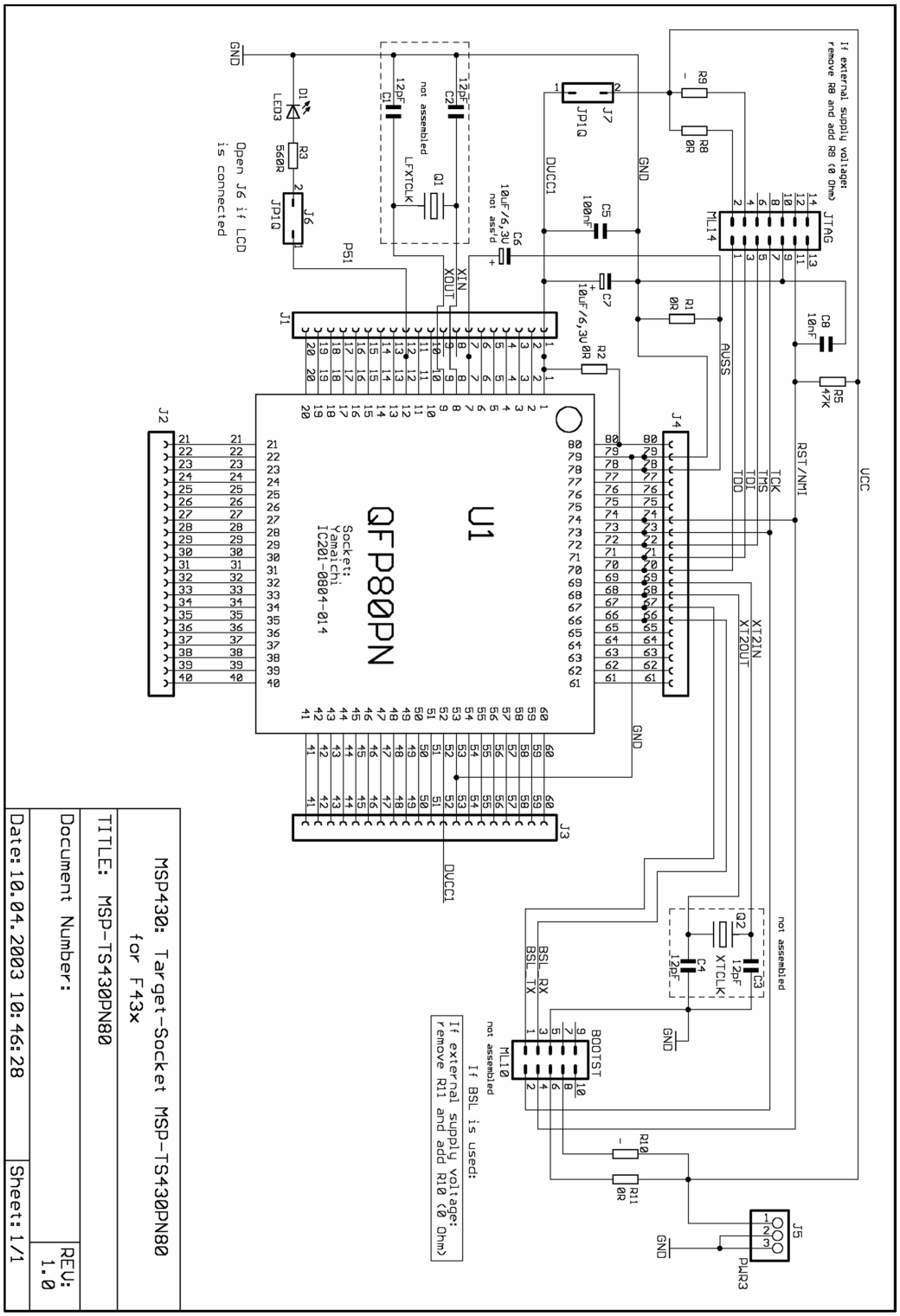


Figure B-15. MSP-TS430PN80 Target Socket module, Schematic

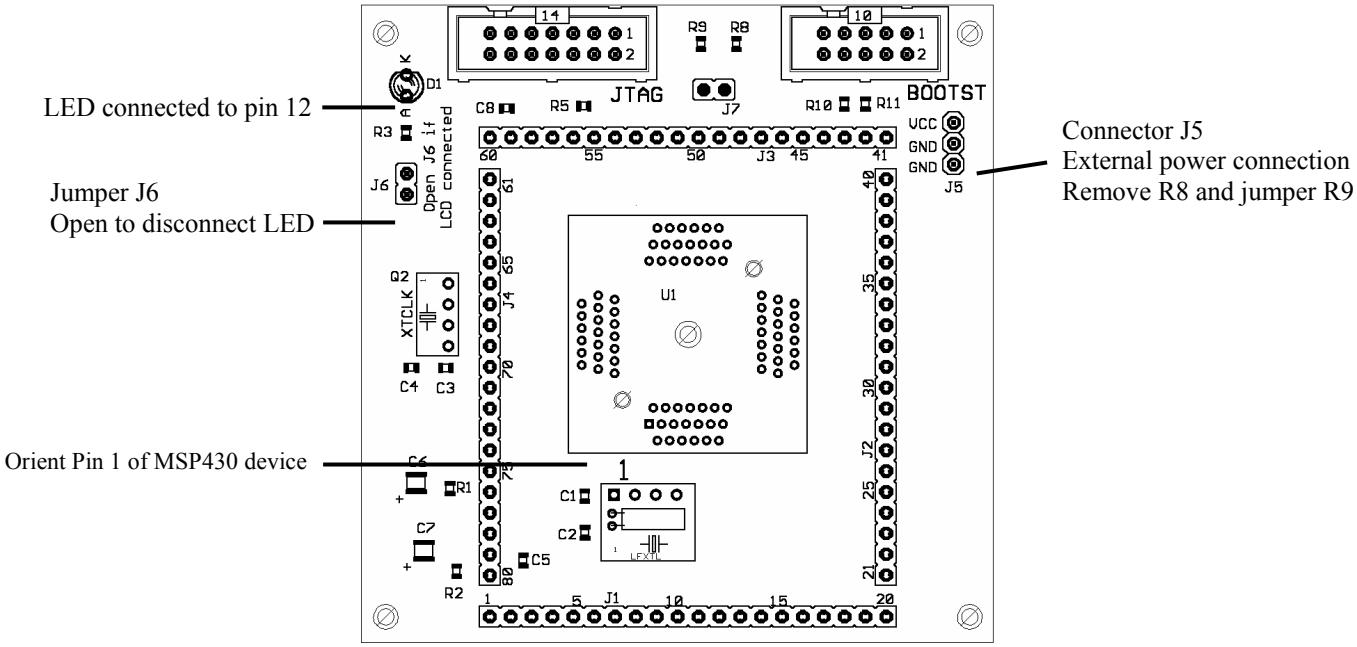
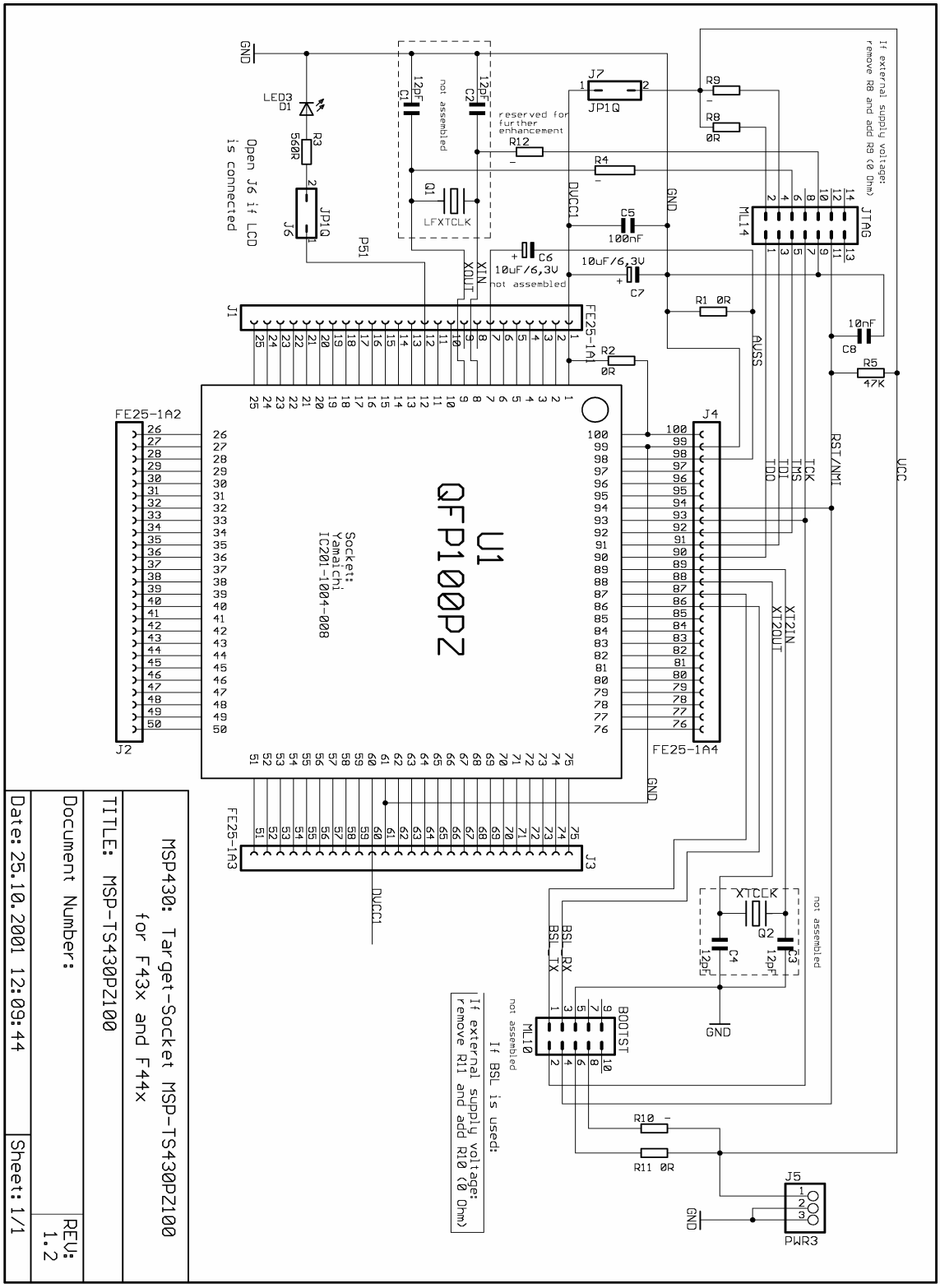


Figure B-16. MSP-TS430PN80 Target Socket module, PCB Pictorials



Note: Connections between the JTAG header and pins XOUT and XIN are no longer required, and should not be made.

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

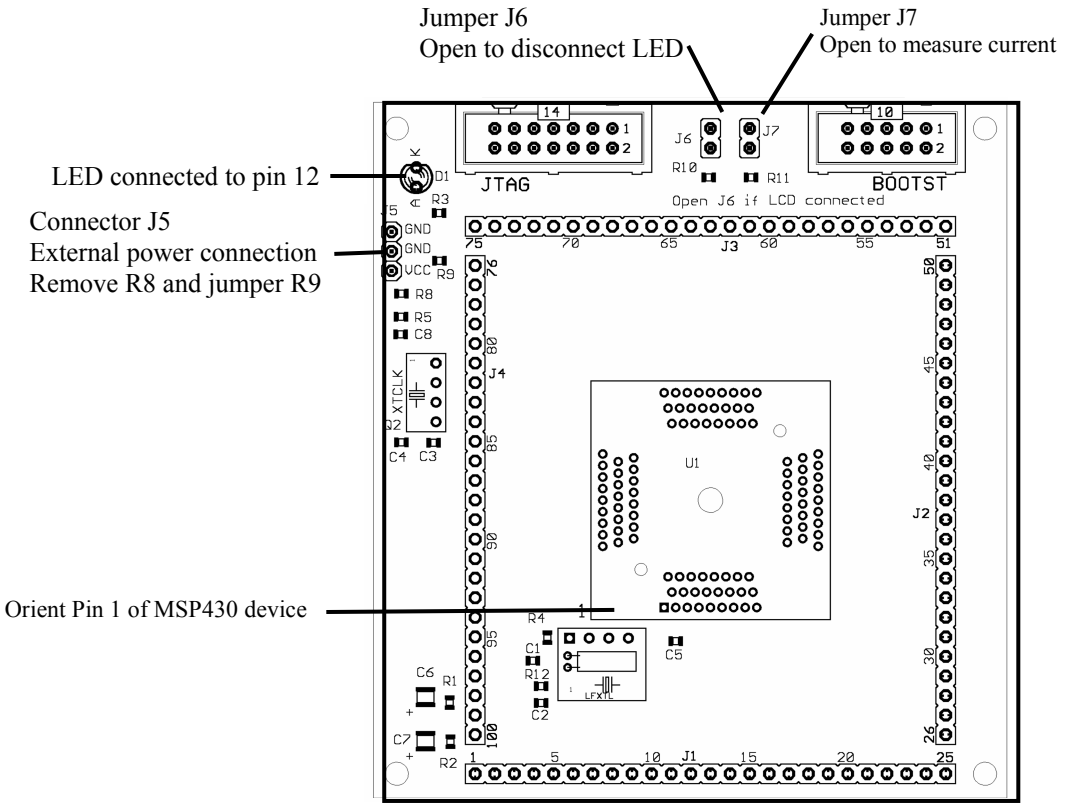


Figure B-18. MSP-TS430PZ100 Target Socket module, PCB Pictorials

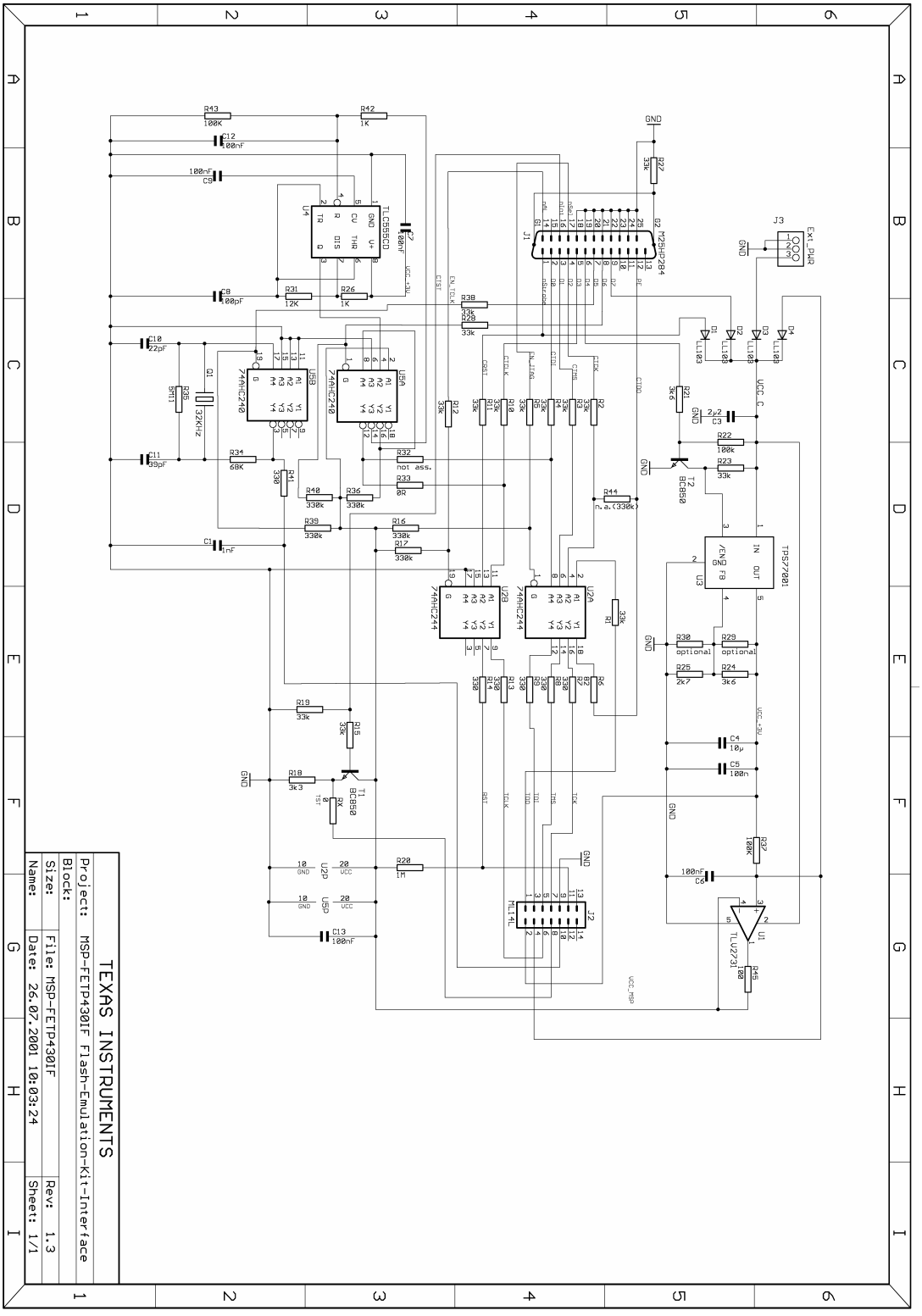


Figure B-19. MSP-FET430PIF FET Interface module, Schematic

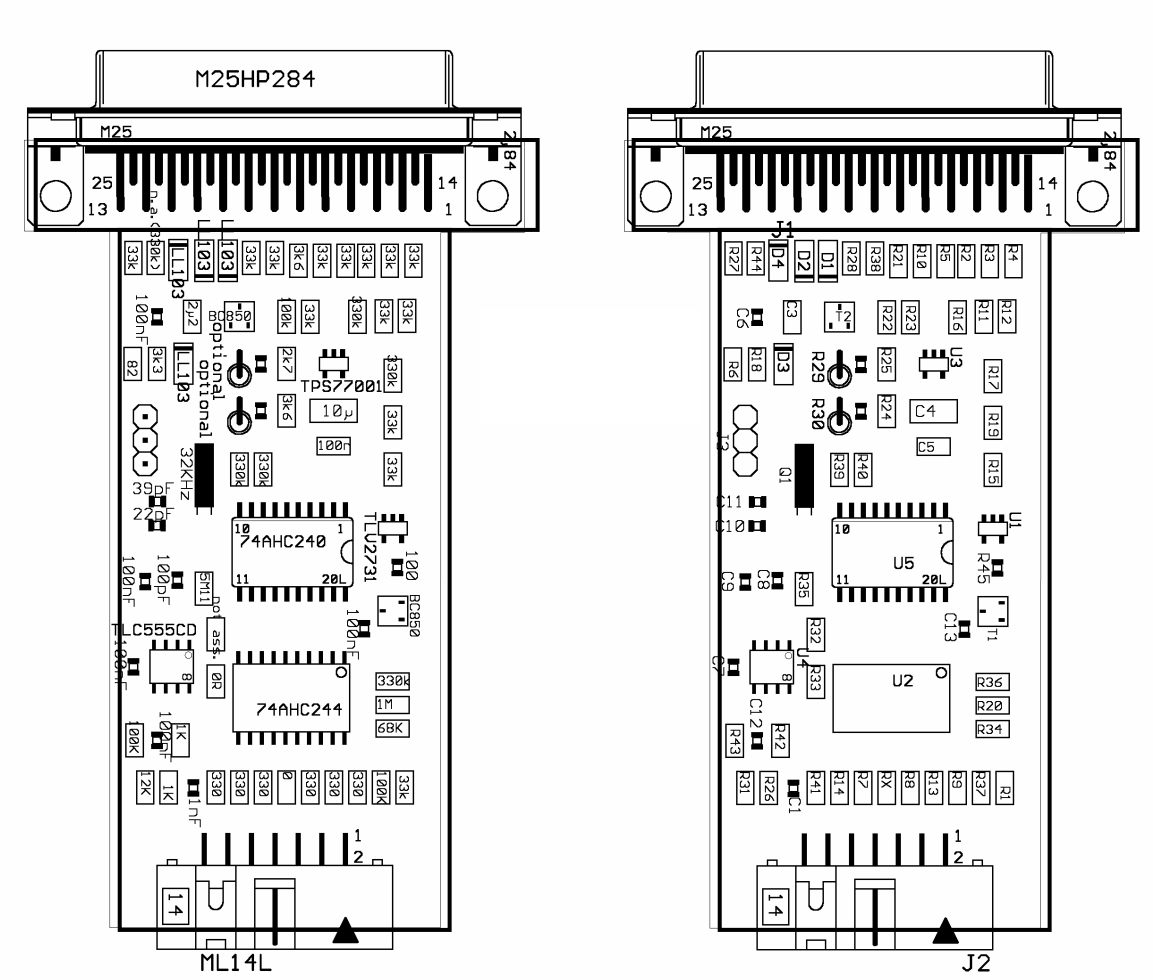


Figure B-20. MSP-FET430PIF FET Interface module, PCB Pictorials

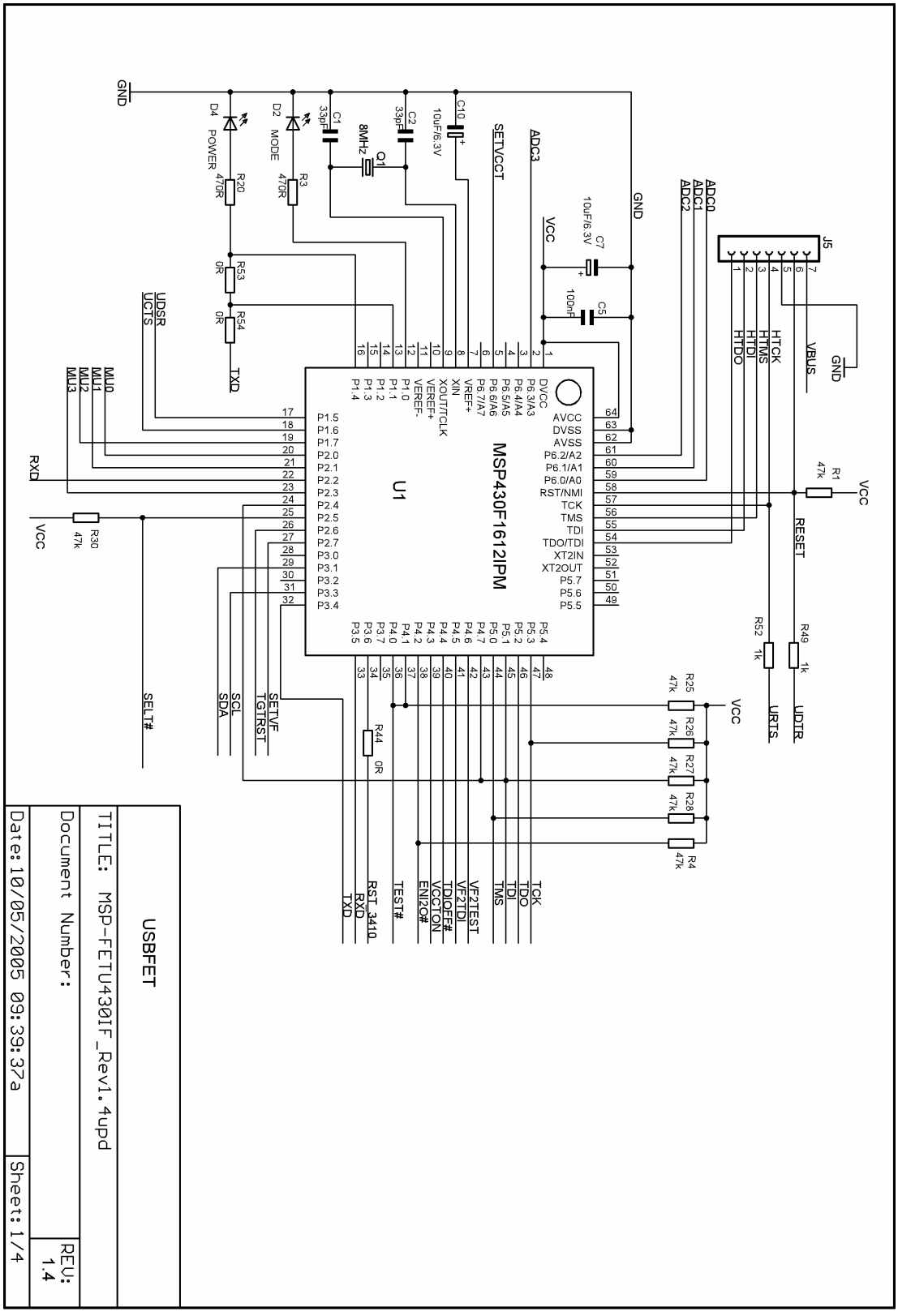
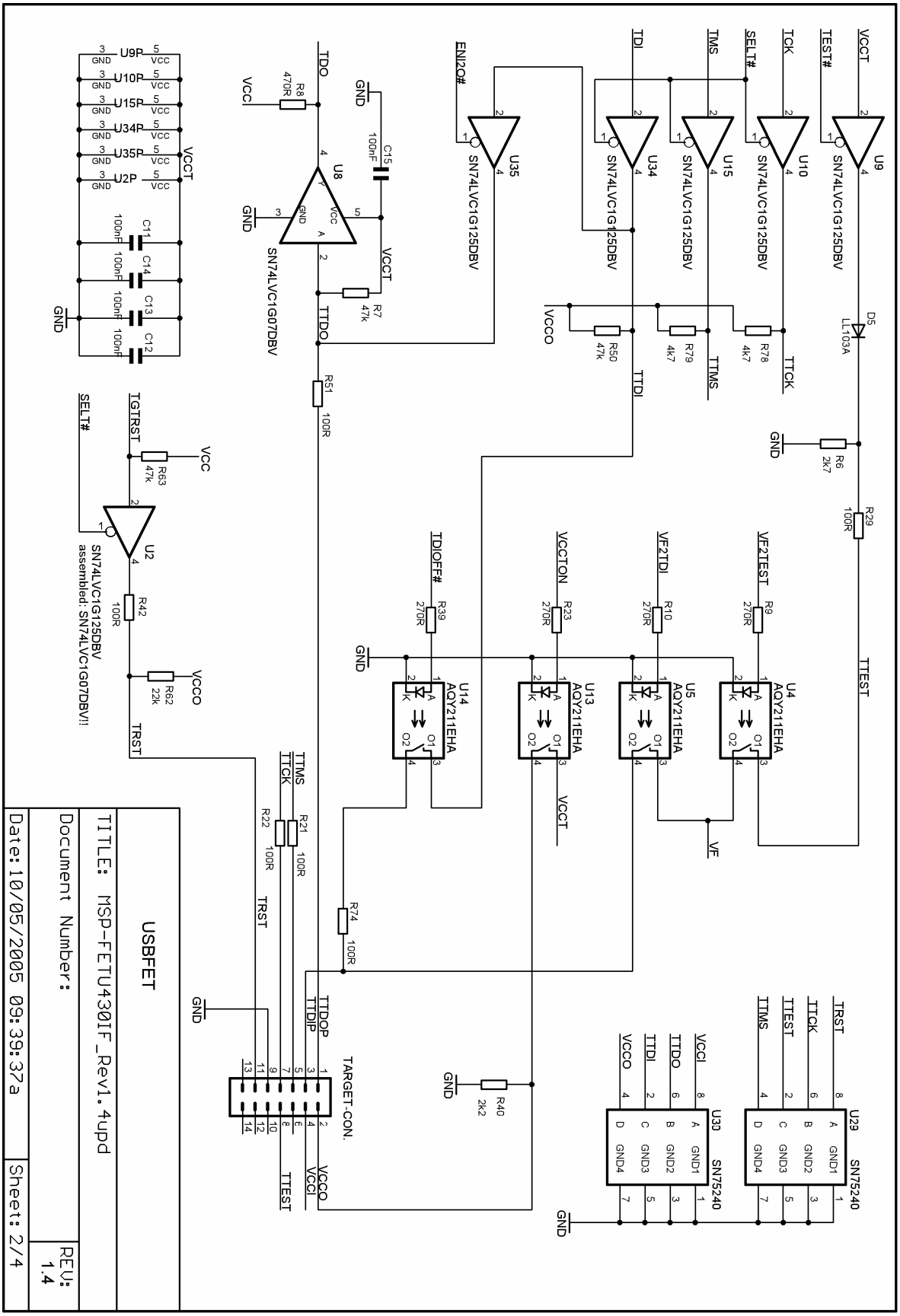
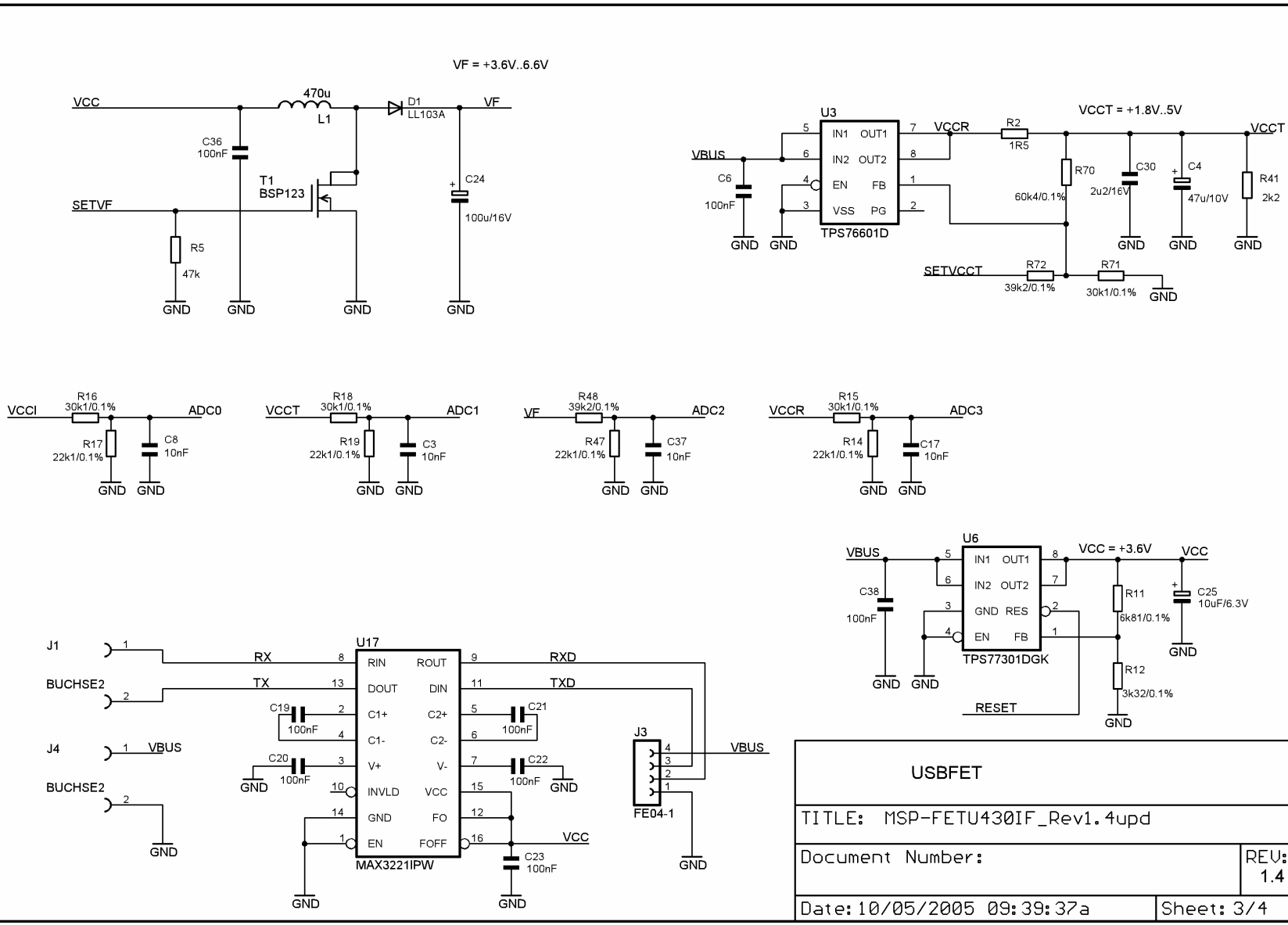
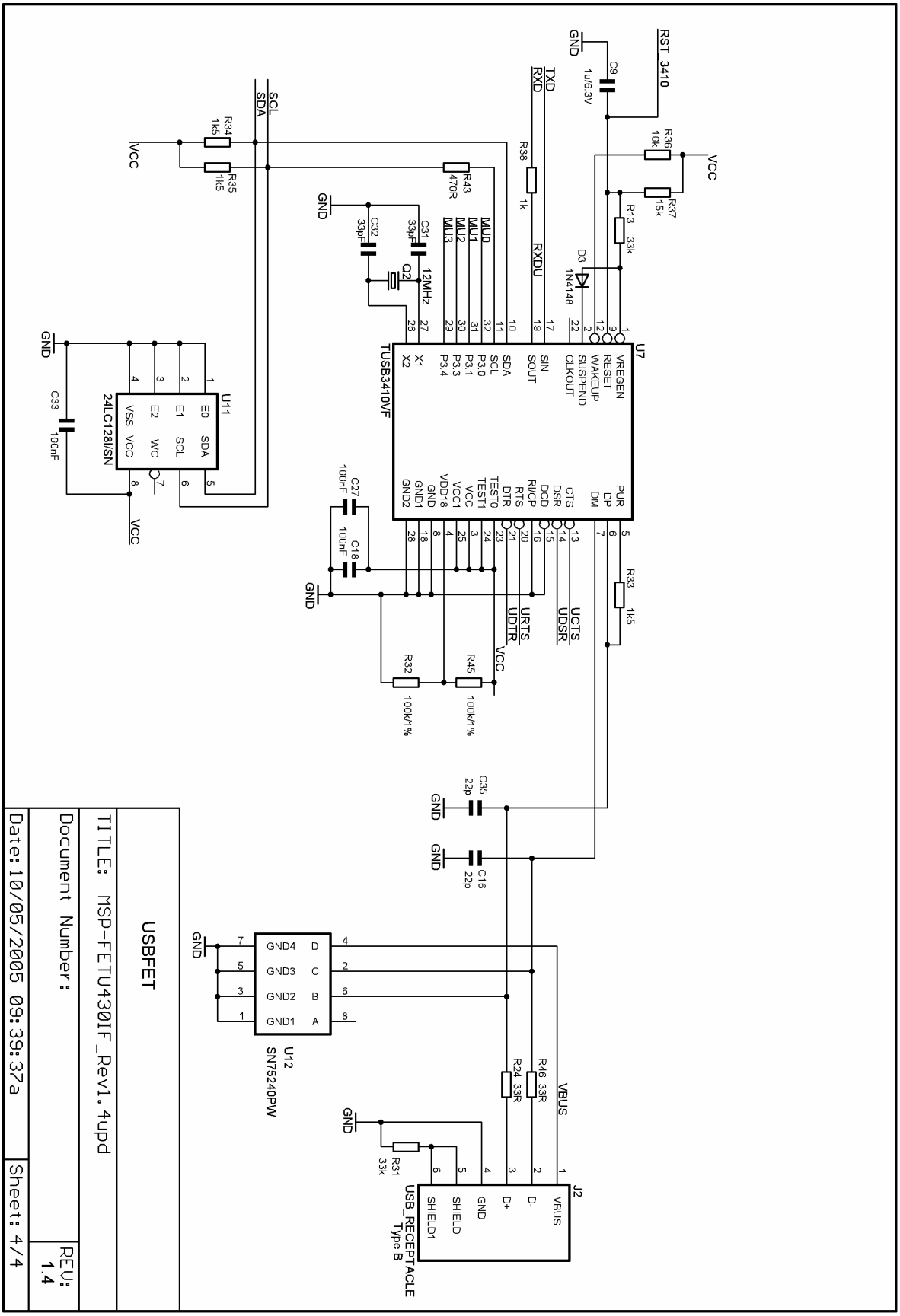


Figure B-21. MSP-FET430UIF USB Interface, Schematic


VCCT
TEST#
TCK
SELT#
TMS
TDI
ENI2C#
U9
SN74LVC1G125DBV
U10
U15
U34
U35
D5
LL103A
R6
2k7
R29
100R
TTEST
R78
4k7
TTCK
R79
4k7
TTMS
R50
47k
TTDI
VCCO
GND
TDO
R8
470R
C15
100nF
U8
SN74LVC1G07DBV
R7
47k
TTDO
R51
100R
U9P
U10P
U15P
U34P
U35P
U2P
C11
C14
C13
C12
100nF
TGTRST
R63
47k
U2
SN74LVC1G125DBV
assembled: SN74LVC1G07DBV!!
R42
100R
R62
22k
TRST
VF2TEST
VF2TDI
TDIOFF#
VCCTON
R9
R10
R39
R23
270R
U4
U5
U14
U13
AQY211EHA
VF
VCCT
R74
100R
R21
R22
TTDOP
TTDIP
TARGET-CON.
R40
2k2
TRST
TTCK
TTMS
TTDI
TTDO
VCCI
VCCO
TTEST
U29
U30
SN75240
GND1
GND2
GND3
GND4
A
B
C
D
1
2
3
4
5
6
7
8
USBFET
TITLE: MSP-FETU430IF_Rev1.4upd
Document Number:
REV: 1.4
Date: 10/05/2005 09:39:37a
Sheet: 2/4


VF = +3.6V..6.6V
VCC
470u
L1
D1
LL103A
VF
C36
100nF
T1
BSP123
C24
100u/16V
SETVF
R5
47k
GND
U3
IN1
OUT1
IN2
OUT2
EN
FB
VSS
PG
TPS76601D
VBUS
C6
100nF
VCCR
R2
1R5
VCCT = +1.8V..5V
VCCT
R70
60k4/0.1%
C30
2u2/16V
C4
47u/10V
R41
2k2
SETVCCT
R72
39k2/0.1%
R71
30k1/0.1%
VCCI
R16
30k1/0.1%
ADC0
R17
22k1/0.1%
C8
10nF
VCCT
R18
30k1/0.1%
ADC1
R19
22k1/0.1%
C3
10nF
VF
R48
39k2/0.1%
ADC2
R47
22k1/0.1%
C37
10nF
VCCR
R15
30k1/0.1%
ADC3
R14
22k1/0.1%
C17
10nF
U6
VBUS
C38
100nF
IN1
OUT1
IN2
OUT2
GND
RES
EN
FB
TPS77301DGK
VCC = +3.6V
VCC
R11
6k81/0.1%
C25
10uF/6.3V
R12
3k32/0.1%
RESET
J1
BUCHSE2
RX
TX
J4
BUCHSE2
VBUS
U17
RIN
ROUT
DOUT
DIN
C1+
C2+
C1-
C2-
V+
V-
INVLD
VCC
GND
FO
EN
FOFF
MAX3221IPW
C19
100nF
C20
100nF
C21
100nF
C22
100nF
C23
100nF
RXD
TXD
J3
FE04-1
VBUS
USBFET
TITLE: MSP-FETU430IF_Rev1.4upd
Document Number:
REV:
1.4
Date: 10/05/2005 09:39:37a
Sheet: 3/4


USBFET
TITLE: MSP-FETU430IF_Rev1.4upd
Document Number:
REV: 1.4
Date: 10/05/2005 09:39:37a
Sheet: 4/4
TUSB3410VF
U7
24LC128I/SN
U11
SN75240PW
U12
USB_RECEPTACLE Type B
J2
RST_3410
TXD
RXD
RXDU
SCL
SDA
VCC
GND
MU0
MU1
MU2
MU3
UCTS
UDSR
URTS
UDTR
VBUS
12MHz
Q2
D3
1N4148
R13 33k
R36 10k
R37 15k
R38 1k
R43 470R
R34 1k5
R35 1k5
R33 1k5
R32 100k/1%
R45 100k/1%
R24 33R
R46 33R
R31 33k
C9 1uf/6.3V
C31 33pF
C32 33pF
C33 100nF
C27 100nF
C18 100nF
C35 22p
C16 22p

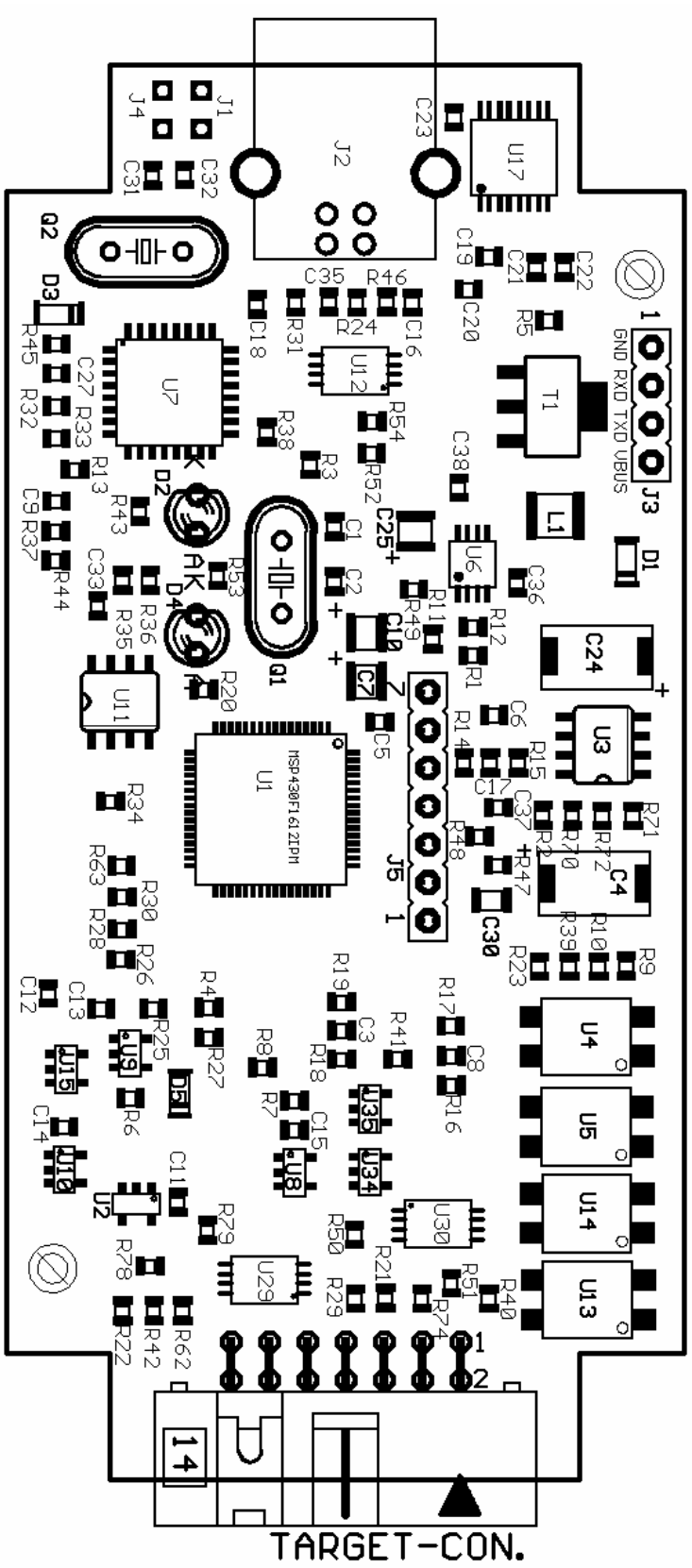


Figure B-22. MSP-FET430UIF USB Interface, PCB Pictorial

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 --> 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

Pull-ups 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. Refer to Figure B-18 on page B-22 for the exact location.

Appendix C

FET Specific Menus

This appendix describes the C-SPY menus that are specific to the FET.

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 1.8V and 5.0V. This voltage is available on pin 2 of the 14-pin target connector to supply the target from the USB FET. If the target is supplied externally, the external supply voltage should be connected to pin 4 of the target connector, so the USB FET 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?).

Refer to FAQ, Debugging #11).

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

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

C.1.6 EMULATOR--> SHOW USED BREAKPOINTS

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). Refer to FAQ, Debugging #18).

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.

Refer to 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 present only in those devices that contain the EEM.

Refer to 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.

Refer to 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.

Refer to 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. Only in this mode the cycle counter works correctly.

C.1.18 EMULATOR--> SET VCC

On the USB FET the target supply voltage can be adjusted between 1.8V and 3.6V. This voltage is available on pin 2 of the 14-pin target connector to supply the target from the USB FET. If the target is supplied externally, the external supply voltage should be connected to pin 4 of the target connector, so the USB FET can set the level of the output signals accordingly.

Note: Availability of EMULATOR--> ADVANCED menus

Not all **EMULATOR--> ADVANCED** menus are supported by all MSP430 devices. These menus will be grayed-out.

Appendix D

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. F4xx/80-pin Signal Mapping 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. Refer to Appendix C.1.8 EMULATOR--> ADVANCED--> EMULATION MODE to enable the emulation mode.

Topic **Page**

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
P3.0/STE0/S31	43	43	43-71
COM0	44	52†	
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†
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

† Note discontinuity of pin numbering sequence

Appendix E

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” (Figure E-1).

Figure E-1. WinXP Hardware Recognition

3) The Hardware Wizard should start automatically and popup the “Found New Hardware Wizard” dialog window.

4) Instruct the Wizard to install the hardware driver from a specific location (Figure E-2).

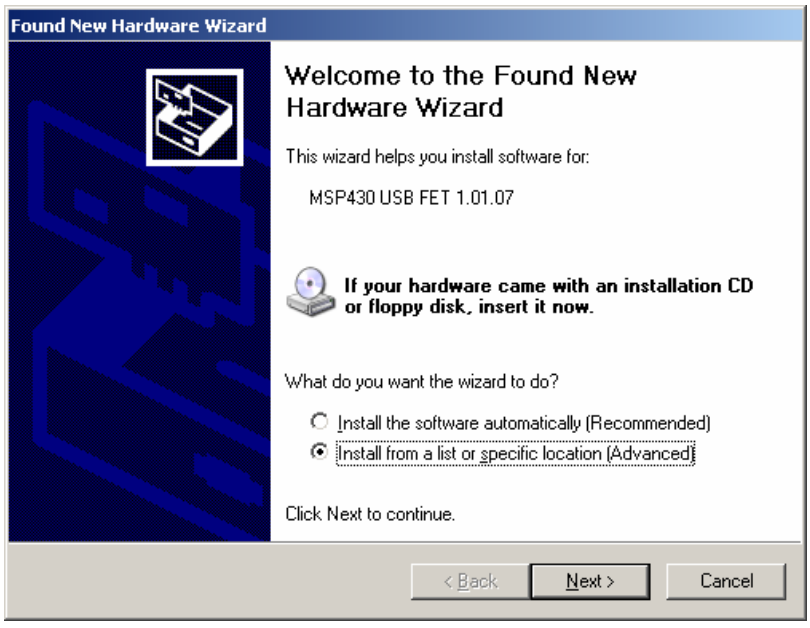


Figure E-2. WinXP Hardware Wizard

5) Point the Hardware Wizard to the folder where the corresponding driver information files are located on your hard disk.

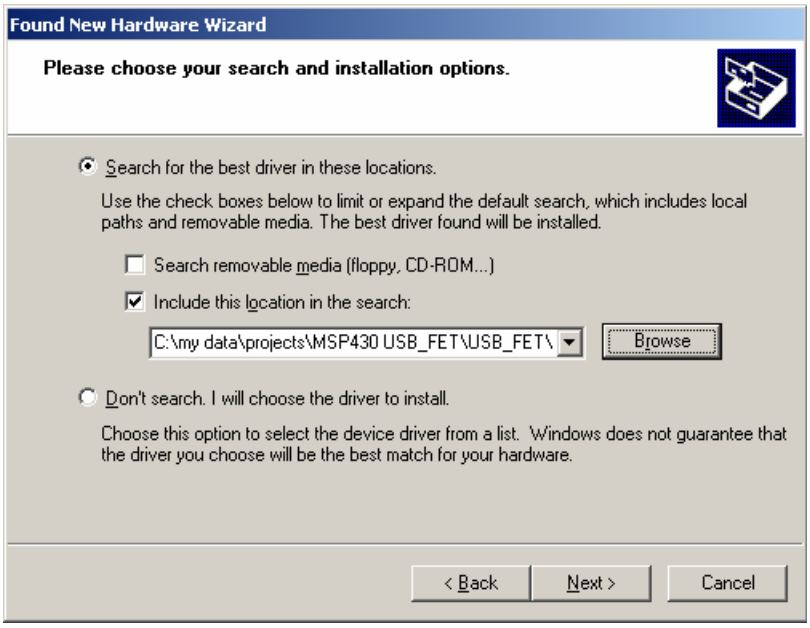


Figure E-3. WinXP Driver Location Selection Folder

6) The Wizard should generate 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” (Figure E-4).

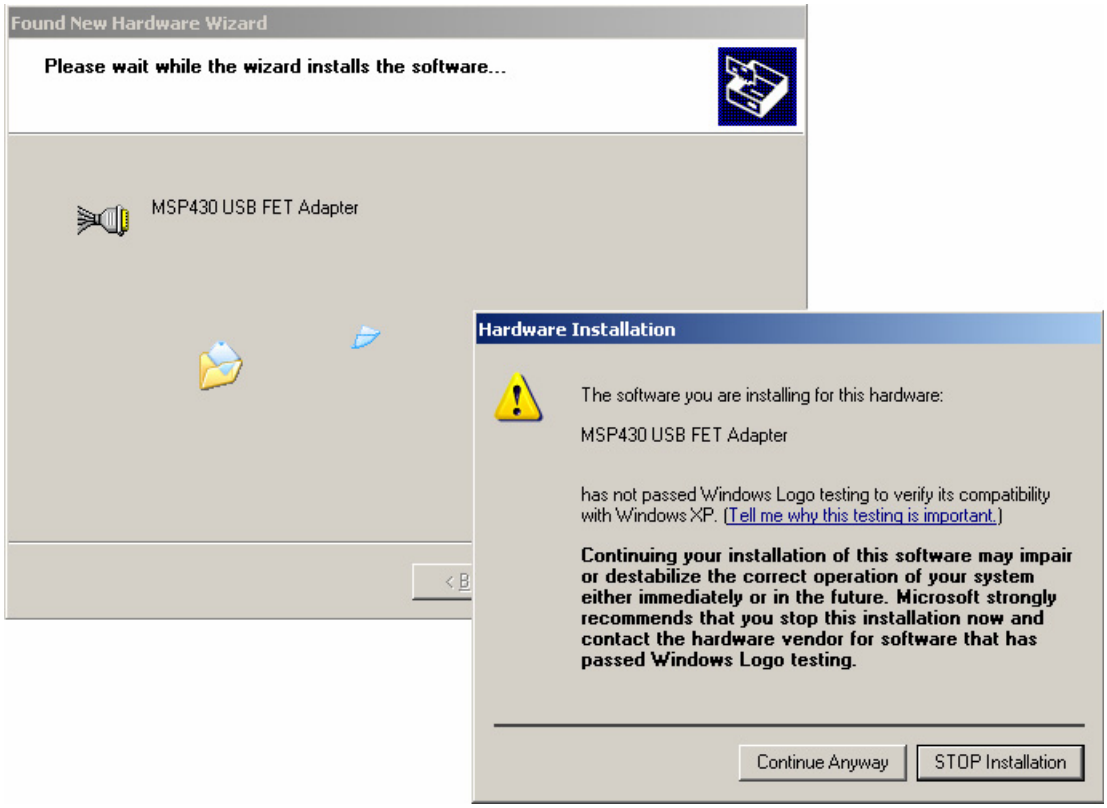


Figure E-4. WinXP Driver Installation

8) In the next step the Wizard installs the driver files.

9) The Wizard now shows a message that it has finished the installation of the software for “MSP430 USB FET Adapter”.

10) After closing the Hardware Wizard, Windows automatically recognizes another new hardware device called “Texas Instruments UMP Serial Port”.

11) Depending on the current update version of the OS corresponding drivers are installed automatically or the Hardware Wizard pops up again. In case of the Wizard is started, please repeat the steps already described above again

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

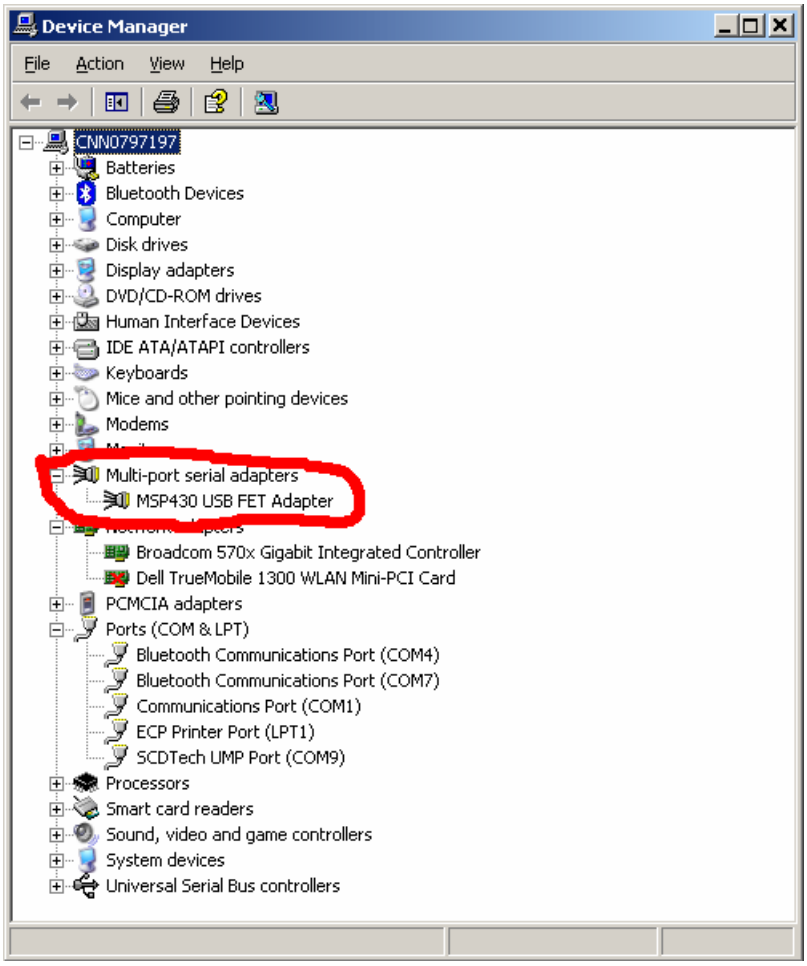


Figure E-5. Device Manager