

MSP430 Family Programming Adapter Manual

User's Guide

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 acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor 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.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

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 semiconductor 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, warranty or endorsement thereof.

Preface

Read This First

About This Manual

This document describes the hardware and software installation and setup. It explains operation and EPROM programming

The manual assumes that you have set up the Windows operating environment and that you are familiar with the basic terminology and procedures for using Microsoft Windows. To set up Windows or review basic Windows information, see your Windows documentation.

How to Use This Manual

This document contains the following chapters:

- ☐ Chapter 1 Installation and Setup
- ☐ Chapter 2 Operation
- ☐ Chapter 3 Hardware
- ☐ Chapter 4 EPROM Programming

Notational Conventions

This document uses the following conventions.

- ☐ Program listings, program examples, and interactive displays are shown in a special typeface similar to a typewriter's. Examples use a **bold version** of the special typeface for emphasis; interactive displays use a **bold version** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

Here is a sample program listing:

```
0011 0005 0001      .field    1, 2
0012 0005 0003      .field    3, 4
0013 0005 0006      .field    6, 3
0014 0006           .even
```

Here is an example of a system prompt and a command that you might enter:

```
C:  csr -a /user/ti/simuboard/utilities
```

- In syntax descriptions, the instruction, command, or directive is in a **bold typeface** font and parameters are in an *italic typeface*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a directive syntax:

.asect *section name, address*

.asect is the directive. This directive has two parameters, indicated by *section name* and *address*. When you use **.asect**, the first parameter must be an actual section name, enclosed in double quotes; the second parameter must be an address.

- Square brackets (**[** and **]**) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don't enter the brackets themselves. Here's an example of an instruction that has an optional parameter:

LALK *16-bit constant [, shift]*

The LALK instruction has two parameters. The first parameter, *16-bit constant*, is required. The second parameter, *shift*, is optional. As this syntax shows, if you use the optional second parameter, you must precede it with a comma.

Square brackets are also used as part of the pathname specification for VMS pathnames; in this case, the brackets are actually part of the pathname (they are not optional).

- Braces (**{** and **}**) indicate a list. The symbol **|** (read as *or*) separates items within the list. Here's an example of a list:

{ * | *+ | *- }

This provides three choices: *****, ***+**, or ***-**.

Unless the list is enclosed in square brackets, you must choose one item from the list.

- Some directives can have a varying number of parameters. For example, the **.byte** directive can have up to 100 parameters. The syntax for this directive is:

.byte *value₁ [, ... , value_n]*

This syntax shows that **.byte** must have at least one value parameter, but you have the option of supplying additional value parameters, separated by commas.

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

Trademarks

Microsoft Windows is a trademark of Microsoft.

Contents

1	Installation and Setup	1-1
1.1	Installing the Software	1-2
1.2	Installing the Hardware	1-3
2	Operation	2-1
2.1	Programming the MSP430 Devices	2-2
2.1.1	Procedure	2-2
2.1.2	Error Messages	2-2
3	Hardware	3-1
3.1	Specifications	3-2
3.2	Basic Hints	3-2
3.3	Programming Adapter Target Connector Signals	3-3
3.4	Programming Adapters	3-6
3.5	MSP-PRG430A	3-7
3.6	MSP-PRG430B	3-8
3.7	Circuit Diagram of MSP-PRG430C	3-10
3.7.1	Upgrade of MSP-PRG430B to MSP-PRG430C	3-12
3.8	Circuit Diagram of MSP-PRG430D	3-13
3.8.1	Upgrade of MSP-PRG430B to MSP-PRG430D	3-15
3.8.2	Upgrade of MSP-PRG430C to MSP-PRG430D	3-15
3.9	Location of Components, MSP-PRG430B	3-16
3.10	Interconnection of MSP-PRG430B/430C to MSP430C313DL/430P313SDL, MSP430C311SDL/P315SDL or 'E313FZ	3-17
3.11	Interconnection of MSP-PRG430B, or '430C to MSP430C325PG, C325PM MSP430P325PG or 'P325PM	3-23
3.12	Interconnection of MSP-PRG430B/430C/430D to MSP430C336PJM/337PJM or MSP430E337CQFP	3-31
3.13	Interconnection of MSP-PRG430C to MSP430C111DW, MSP430C112DW, MSP430P112DW, or MSP430E112JL	3-34
4	EPROM Programming	4-1
4.1	EPROM Operation	4-2
4.1.1	Erase	4-2
4.1.2	Programming Methods	4-2
4.1.3	EPROM Control Register EPCTL	4-3
4.1.4	EPROM Protect	4-4
4.2	FAST Programming Algorithm	4-4
4.3	Programming an EPROM Module Through a Serial Data Link Using the JTAG Feature	4-5
4.4	Programming an EPROM Module With Controller's Software	4-6
4.5	Code	4-8

Figures

1-1	ADT430 Program Icons	1-2
2-1	MSP430 Program Device Dialog Box	2-2
2-2	Communication Error Box	2-2
2-3	Communication Error Box for Blown Fuse	2-3
2-4	Erase Check Error Message	2-4
2-5	Data Error	2-4
3-1	9-Pin Sub-D at the Programming Adapter	3-3
3-2	14-Pin Connector at the End of the Interconnect Cable	3-3
3-3	MSP-PRG430A Circuit Diagram	3-7
3-4	MSP-PRG430B Circuit Diagram, Part 1	3-8
3-5	MSP-PRG430B Circuit Diagram, Part 2	3-9
3-6	MSP-PRG430C Circuit Diagram, Part 1	3-10
3-7	MSP-PRG430C Circuit Diagram, Part 2	3-11
3-8	MSP-PRG430D Circuit Diagram, Part 1	3-13
3-9	MSP-PRG430D Circuit Diagram, Part 2	3-14
3-10	MSP-PRG430B Components	3-16
3-11	MSP-PRG430B Used to Program the MSP430P313DL Device	3-17
3-12	MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P313DL Device	3-18
3-13	MSP-PRG430B Used to Program the MSP430P315SDL Device	3-19
3-14	MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P315SDL Device	3-20
3-15	MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P313DL Device	3-21
3-16	MSP-PRG430B Used to Program the MSP430E313FZ Device	3-22
3-17	MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430E313FZ Device	3-22
3-18	MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430E313FZ Device	3-23
3-19	MSP-PRG430B Used to Program the MSP430P325PG Device	3-24
3-20	MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P325PG or MSP430P325APG Device	3-24
3-21	MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P325PG or MSP430P325APG Device	3-25
3-22	MSP-PRG430B Used to Program the MSP430C325PM or MSP430P325PM Device	3-26
3-23	MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P325PM or MSP430P325APM Device	3-27
3-24	MSP-PRG430B or MSP-PRG430C or PRG430D Used to Program the MSP430P325PM or MSP430P325APM Device	3-28
3-25	MSP-PRG430B Used to Program the MSP430E325FZ Device	3-29
3-26	MSP-PRG430B or MSP-PRG430C Used to Program the MSP430E325FZ Device	3-30

3-27	MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430E325FZ Device	3-31
3-28	MSP-PRG430B Used to Program the MSP430P325PM or MSP430P325APM Device	3-32
3-29	MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430C336PJM, MSP430P337PJM, or the MSP430E337 Device	3-32
3-30	MSP-PRG430B or MSP-PRG430C Used to Program the MSP430C336PJM, MSP430P337PJM, or the MSP430E337 Device	3-33
3-31	MSP-PRG430D or Modified MSP-PRG430C or Modified MSP-PRG430B	3-34
3-32	MSP-PRG430C or Upgraded MSP-PRG430	3-35
4-1	EPROM Control Register EPCTL	4-3
4-2	EPROM Programming With Serial Data Link	4-5
4-3	EPROM Programming With Controller's Software	4-6

Tables

2-1	MSP430 Function Buttons and Descriptions	2-3
2-2	Error Messages	2-4
3-1	MSP430 Hardware Specifications	3-2
3-2	Target Connector Signal Functions	3-3
3-3	Programming Adapter Signal Levels	3-5
3-4	Programming Adapter Versions Compatibility and Upgrades	3-6

Examples

4-1	MSP430 On-Chip Program Memory Format	4-3
4-2	Fast Programming Subroutine	4-4
4-3	Programming EPROM Module With Controller's Software	4-7
4-4	Subroutine	4-7

Installation and Setup

This chapter describes the process of installing and programming the hardware and software for the MSP430 family of microcontrollers.

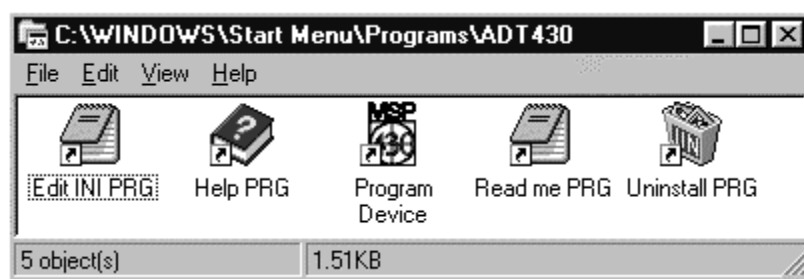
Topic	Page
1.1 Installing the Software	1-2
1.2 Installing the Hardware	1-3

1.1 Installing the Software

To install the MSP-PRG430D software, perform the following steps:

- 1) Insert the MSP-PRG430D installation disk and click Setup.exe under Microsoft® Windows™.
- 2) Follow the setup instructions on the screen. The setup program guides you through the installation process.
- 3) After you run setup, the MSP430 Program icons are displayed. Double-click the Read me PRG icon, shown in Figure 1–1, to obtain important information about the program device hardware and software. The ADT430 Program Icons window appears.

Figure 1–1. ADT430 Program Icons



- 4) The appropriate program group and icons are added to the Windows program manager.
- 5) To start the programming adapter software, double-click the Program Device icon in the ADT430 program group.

1.2 Installing the Hardware

To install the programming adapter hardware, perform the following steps:

- 1) Using the 25-pins SUB-D connector, connect the programming adapter to the parallel port (LPT1–LPT3) of the PC.
- 2) Connect an external power supply to the programming adapter. The voltage of the power supply must be between 14 V and 20 V dc and must provide a minimum of 200 mA of power. The center terminal of the supply connector at the programming adapter is the plus pole.
- 3) The red LED on the programming adapter lights if the power supply is properly connected. If the LED does not light and the power supply is properly connected check the F1 fuse on the programming adapter printed wire board (PWB).
- 4) The MSP430 devices, in a socket or on a PWB, should be connected to the programming adapter through the 9-pin cable.

If only the device itself is connected, the programming adapter provides a supply voltage V_{CC} of ≈ 5 V at pin 6 of the 9-pin SUB-D connector or pin 2 of the 14-pin connector. The signal name is VCC_MSP.

If the device is supplied externally, as in the case of in-circuit programming, the external VCC should be connected to pin 7 of the 9-pin SUB-D connector or pin 4 of the 14-pin connector. This configuration ensures that the voltage level of the programming signals is adjusted to the external VCC. The signal name is MSP_VCC_IN.

Note:

Almost all standard power supplies yield a higher output voltage than specified. Most universal 12-V power supplies can be used as the external power supply for the programming adapter.



Operation

This section describes the programming procedure for MSP430 devices and error messages you may encounter during the procedure.

Topic	Page
2.1 Programming the MSP430 Devices	2-2

2.1 Programming the MSP430 Devices

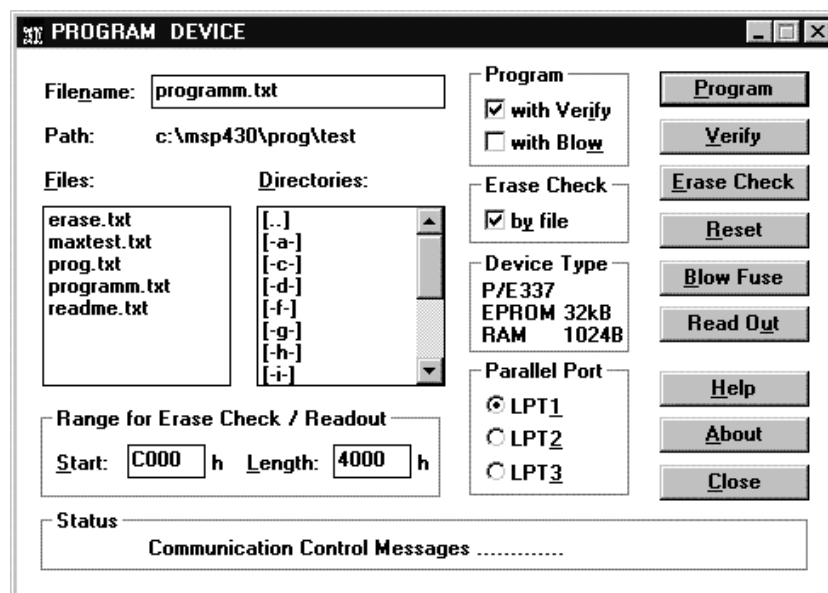
2.1.1 Procedure

To program the MSP430 devices, perform the following steps:

Double-click the Program Device icon in the ADT430 program group. The MSP430 program device dialog box appears.

The Status line at the bottom of the window shows the actual or the most recent activity (see Figure 2–1). The status line flashes the message *In Progress* while a function is active.

Figure 2–1. MSP430 Program Device Dialog Box



2.1.2 Error Messages

If JTAG communication is not accessible, the following message showing possible reasons for communication errors appears (see Figure 2–2).

Figure 2–2. Communication Error Box



If the fuse is already blown, the following error message appears (see Figure 2–3).

Figure 2–3. Communication Error Box for Blown Fuse



A common way to program an MSP430 device is by selecting the file with the appropriate data from the Program Device dialog box (Figure 2–1) and selecting the Parallel Port LPTx button. The device configuration and memory type are selected automatically according to the connected device. Clicking the Program button starts the programming operation.

Table 2–1 describes the function buttons for different options and combinations for the MSP430 Program Device dialog box.

Table 2–1. MSP430 Function Buttons and Descriptions

Button Name	Description
Program	An object code is programmed to the on-chip memory without Verify and without Blow. The code protection fuse is left intact.
Program with Verify	Each section is verified after it is programmed.
Program with Blow	The code protection fuse is blown after the complete object code, with or without Verify, is programmed. This action irreversibly disables future access (read or programming) to the on-chip memory.
Verify	A verification of the selected object file and the program memory contents of the device is performed.
Erase Check	An erase check is performed depending on the values entered in the Range for Erase Check/Readout field.
Erase Check by file	Only the memory locations corresponding to the selected object file are checked.
Reset	A software reset of the chip is generated.
Blow Fuse	Any access (read or programming) to the on-chip memory is disabled irreversibly.
Read Out†	The content of the connected device to the file selected in the Filename field is read out. The desired memory range is selected in the Range for Erase Check/Readout field.
Help	Help is available for programming MSP430 devices, and for using command buttons, selectors, and the used object file format.

† Some older devices without security fuses are read-protected.

For general error messages, such as Erase Check (Figure 2–4), additional message boxes appear.

Figure 2–4. Erase Check Error Message



Table 2–2. Error Messages

Error Type	Error Message
Data Error	Invalid file format (No 'txt'–File)!
File Open Error	File cannot be opened !
File not found	Filename
Communication Error	Please check hardware !
Input Error	Memory Range parameters must be even !
Blow Fuse Error	Fuse has not been blown !
Verification Error	First error at xxxh.
Read access denied	Type xxxx for read–out not released P/E 313 (3784–10)
Erase Check Error	First unerased word at xxxh.

Figure 2–5. Data Error



Hardware

This chapter describes hardware for the MSP430 family of microcontrollers, including hardware specifications, various components of the programming adapters, and its connection of the programming adapter to the MSP430 device families.

Topic	Page
3.1 Specifications	3-2
3.2 Basic Hints	3-2
3.3 Programming Adapter Target Connector Signals	3-3
3.4 Programming Adapters	3-6
3.5 MSP-PRG430A	3-7
3.6 MSP-PRG430B	3-8
3.7 Circit Diagram of MSP-PRG430C	3-10
3.8 Circit Diagram of MSP-PRG430D	3-13
3.9 Location of Components, MSP-PRG430B	3-16
3.10 Interconnection of MSP-PRG430B/430C to MSP430C313DL/ 430P313SDL, MSP430C311SDL/P315SDL or 'E313FZ	3-17
3.11 Interconnection of MSP-PRG430B or '430C to MSP430C325PG, C325PM, MSP430P325PG or 'P325PM	3-23
3.12 Interconnection of MSP-PRG430B/430C/430D to MSP430C336PJM/P337PJM or MSP430E337CQFP	3-31
3.13 Interconnection of MSP-PRG430C to MSP430C111dW, MSP430C112DW, MSP430P112DW or MSP430E112JL	3-34

3.1 Specifications

The specifications for the MSP430 hardware are shown in Table 3–1.

Table 3–1. MSP430 Hardware Specifications

Temperature range	10°C–45°C
Humidity	40%–70%
Power supply	14 V–20 V, 200 mA minimum
Dimensions	(W)140 mm × (H)30 mm × (D)70 mm

3.2 Basic Hints

The basic hints useful for programming MSP430 devices or MSP430 devices on printed wire boards (PWB).

- ☐ All VCC pins of an MSP430 device are tied together and connected to the most positive terminal of the supply.
- ☐ All VSS pins of an MSP430 device are tied together and connected to the most negative terminal of the supply.
- ☐ The interface should supply the MSP430 with proper conditions according to the device datasheet in terms of current, voltage levels, and timing conditions.
- ☐ MSP430x3xx family: Six interconnections are needed as minimum:
TMS, TCK, TDI/VPP, TDO/TDI, VSS, and XOUT.
- ☐ MSP430x11x family: Seven interconnections are needed as minimum:
TMS, TCK, TDI, TDO/TDI, VSS, Test/VPP, and XOUT.
- ☐ Short cables to interconnect the interface to the MSP430 device or PWB; less than 20 cm is recommended.
- ☐ Ensure low-impedance interconnections: Especially for the path of the programming and fuse blow voltage – TDI/VPP (MSP430x3xx family) or Test/VPP (MSP430x11x family).
- ☐ When a device with a transparent window is programmed, the window should be already covered with an opaque label while the device is programmed. Since ambient light contains the correct wavelength for erasure, keep the transparent window covered after the device is programmed.

3.3 Programming Adapter Target Connector Signals

The target connector signals for the programming adapter ensure communication between programming adapter and MSP430 devices and supply low energy to systems without extra supply sources.

Figure 3–1 and Figure 3–2 show the target connector signals for the programming adapter.

Figure 3–1. 9-Pin Sub-D at the Programming Adapter

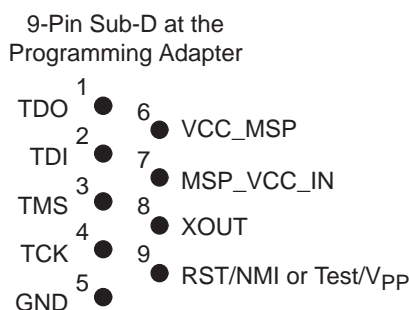


Figure 3–2. 14-Pin Connector at the End of the Interconnect Cable

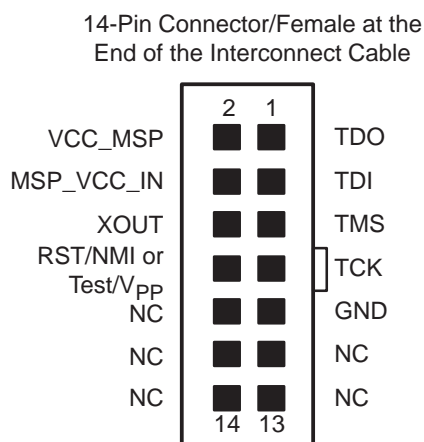


Table 3–2 lists the target connector signals and describes their requirement statuses and functions.

Table 3–2. Target Connector Signal Functions

Signal/Terminal Name	Required	Function/Comment
TMS	Mandatory	Test mode select functions according to IEE1149.1.
TCK	Mandatory	Test clock functions according to IEE1149.1.
TDI/VPP	Mandatory	Test data input functions according to IEE1149.1, but with additional programming voltage.
TDO/TDI	Mandatory	Test data output functions according to IEE1149.1, but additional data input is used when programming voltage is applied by TDI/VPP.

Table 3–2. Target Connector Signal Functions (Continued)

Signal/Terminal Name	Required	Function/Comment
GND	Mandatory	GND is the most negative terminal.
VCC_MSP (see Note 1)	Defined by application	Voltage source is used with MSP430 devices or PWBs. The voltage is ~ 5 V when MSP_VCC_IN is open, or the voltage follows the level applied to MSP_VCC_IN.
MSP_VCC_IN (see Note 1)	Defined by application	Used with function of VCC_MSP.
XOUT	Mandatory	Signal supplies the MSP430 system with clock signals.
RST/NMI (see Note 2)		Signal connects to a pullup resistor to VCC_MSP to prevent unexpected reset conditions.
Test/VPP (see Note 3)	Mandatory	Signal used with MSP430x11x devices to select pin or JTAG function or to apply VPP.

- Notes:**
- 1) The signals VCC_MSP and VCC_MSP_IN may not be connected.
 - 2) The RST/NMI signal is valid for MSP-PRG430A and MSP-PRG430B.
 - 3) The Test/VPP signal is valid for MSP-PRG430C, and MSP-PRG430D.

The output signal levels of the programming adapter depend on the conditions at the MSP_VCC_IN line as indicated in Table 3–3.

- ☐ The RST/NMI terminal of the device must be high; otherwise the access to the device via JTAG system may fail.
- ☐ The programming procedure (handling of the SW) is described in the manual that was included with the programming adapter package.
- ☐ The connections from the MSP430 terminals must follow EMI rules; such as short lines and ground planes. If TMS line receives one negative pulse by EMI strike, the fuse current is activated (with fuse version 1.0). The fuse current flows from TDI/VPP pin to GND (or VSS).

Table 3–3. Programming Adapter Signal Levels

Signal/Pin	Signal/Pin Levels	VCC_MSP	MSP_VCC_IN
TMS	VCC_MSP	~ 5 V	Open
TCK	VCC_MSP	~ 5 V	Open
TDI/VPP	VCC_MSP	~ 5 V	Open
TDO/TDI	VCC_MSP	~ 5 V	Open
XOUT	VCC_MSP	~ 5 V	Open
RST/NMI (see Note 2)	Pull up to VCC_MSP	~ 5 V	Open
Test/VPP (see Note 2)	VSS or VCC_MSP or VPP	~ 5 V	Open
TMS	~Vext	~Vext	Vext (ext. applied voltage)
TCK	~Vext	~Vext	Vext (ext. applied voltage)
TDI/VPP	~Vext	~Vext	Vext (ext. applied voltage)
TDO/TDI	~Vext	~Vext	Vext (ext. applied voltage)
XOUT	~Vext	~Vext	Vext (ext. applied voltage)
RST/ NMI (see Note 2)	Pull up to ~Vext	~Vext	Vext (ext. applied voltage)
Test/VPP (see Note 2)	VSS or Vext or VPP	~Vext	Vext (ext. applied voltage)
TMS	See Note 1	See Note 1	Rext (ext. resistor to GND)
TCK	See Note 1	See Note 1	Rext (ext. resistor to GND)
TDI/VPP	See Note 1	See Note 1	Rext (ext. resistor to GND)
TDO/TDI	See Note 1	See Note 1	Rext (ext. resistor to GND)
XOUT	See Note 1	See Note 1	Rext (ext. resistor to GND)
RST/NMI (see Note 2)	See Note 1	See Note 1	Rext (ext. resistor to GND)
Test/VPP (see Note 2)	VSS or VCC_MSP (See Note 1) or VPP		

Notes: 1) $\sim 5V \cdot [R_{ext}/(R_{ext}+100\text{ k}\Omega)]$ 5 V are as accurate as the TL750LC05CLP (in the PRG430x).

2) The signal is either RST/NMI or Test/VPP. Test/VPP is needed in the '11x's shared JTAG pin implementation. RST/NMI is not always needed. During JTAG access the RST/NMI terminal must be high. If not, no access is possible, and the JTAG function is reset.

3.4 Programming Adapters

There are four versions of the programming adapter: MSP-PRG430A, MSP-PRG430B, MSP-PRG430C, and MSP-PRG430D.

Table 3–4 lists the correlations between the programming adapter versions and the programming adapter devices and indicates each adapter's upgrade capabilities.

Table 3–4. Programming Adapter Versions Compatibility and Upgrades

Prg. Adapter Device	MSP-PRG430A	MSP-PRG430B	MSP-PRG430C	MSP-PRG430D
Upgrade (see Note 1)	Modified	(⇒ MSP-PRG430C)	(⇒ MSP-PRG430D) (see note 3)	
SW version valid	2.10 (see Note 2)	2.10B	3	3.02
MSP430x111	No	No	Yes	Yes
MSP430x112	No	No	Yes	Yes
MSP430x311	No	Yes	Yes	Yes
MSP430x312	Yes	Yes	Yes	Yes
MSP430x313	Yes	Yes	Yes	Yes
MSP430x314	Yes	Yes	Yes	Yes
MSP430x315	No	Yes	Yes	Yes
			(see Note 3)	(see Note 3)
MSP430x323	Yes	Yes	Yes	Yes
MSP430x325	Yes	Yes	Yes	Yes
MSP430x336	Yes	Yes	Yes	Yes
MSP430x337	Yes	Yes	Yes	Yes

Notes:

- 1) Upgrading to the HW and SW is strongly recommended.
- 2) SW version 2.10B is compatible, but it is not recommended.
- 3) The low power EPROM devices require an increased VPP. Programming adapter version D delivers the increased programming voltage VPP. The programming adapter version B and C can be updated easily to version D. Please refer to Paragraph 3.10 or 3.10.2 and also refer to *Read Me PRG*.

3.5 MSP-PRG430A

The PRG430A can be used with software version 2.10. Burning the fuse is not supported. Two pins at resistor network RN2 must be connected: RN2.4 with RN2.6. The resistor R6 should be modified from 220 Ω to 27 Ω . Figure 3-3 shows the circuit diagram for MSP-PRG430A.

Figure 3–3. MSP-PRG430A Circuit Diagram

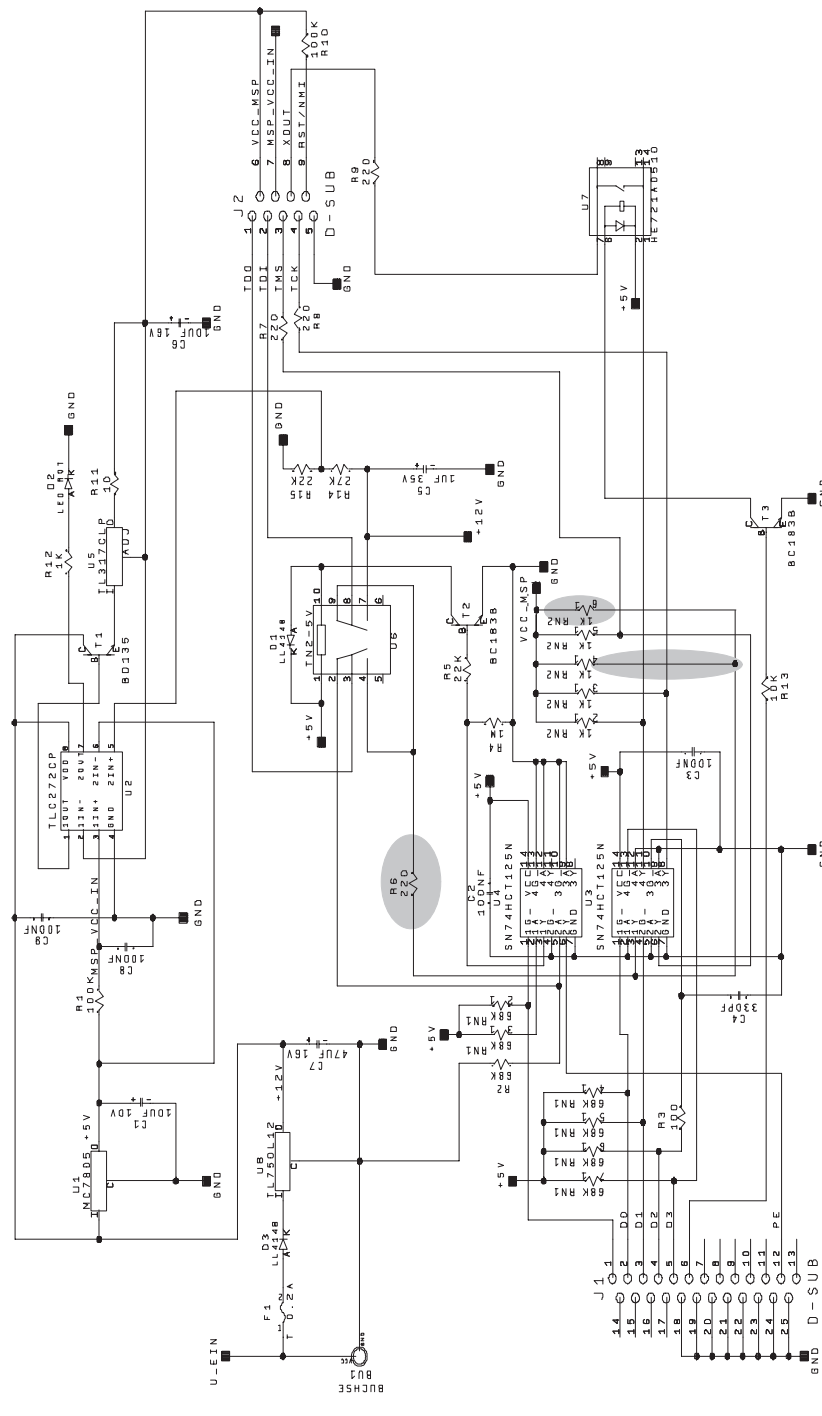
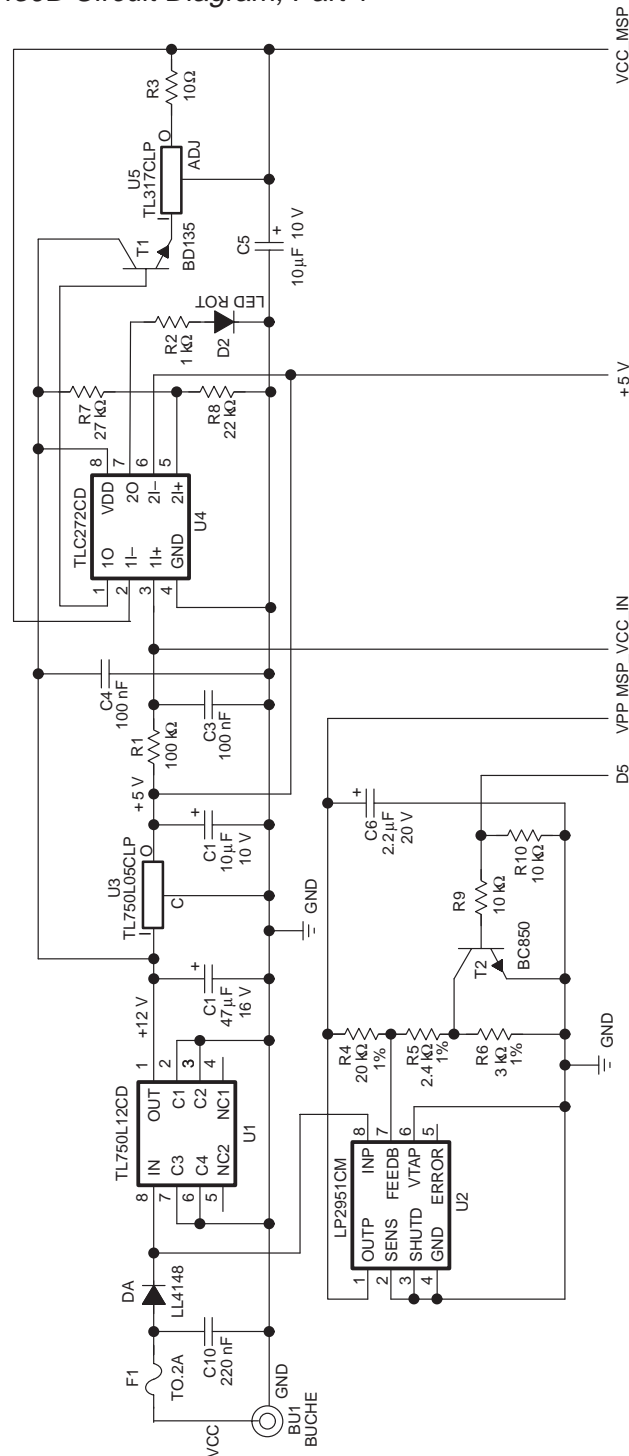


Figure 3–4. MSP-PRG430B Circuit Diagram, Part 1



3.7 Circuit Diagram of MSP-PRG430C

Figure 3–6. MSP-PRG430C Circuit Diagram, Part 1

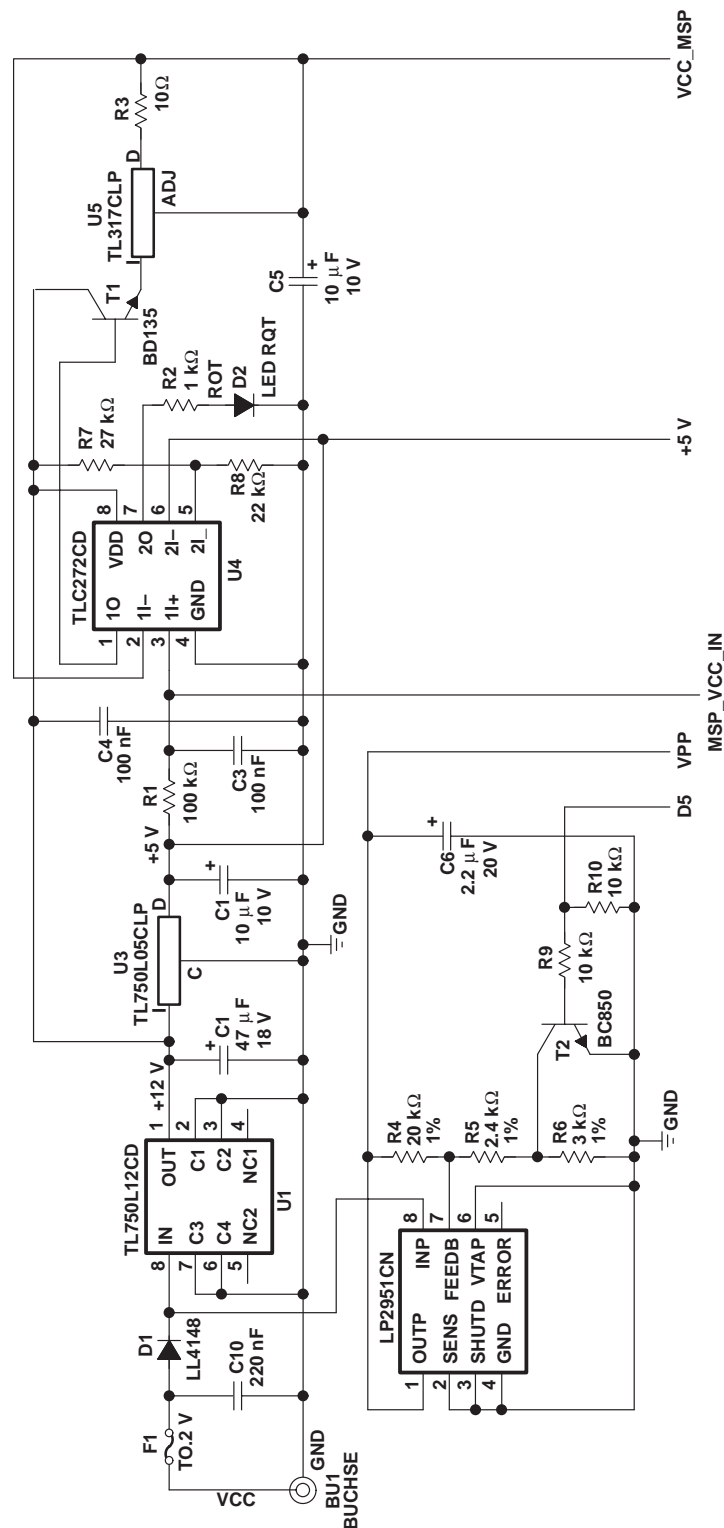
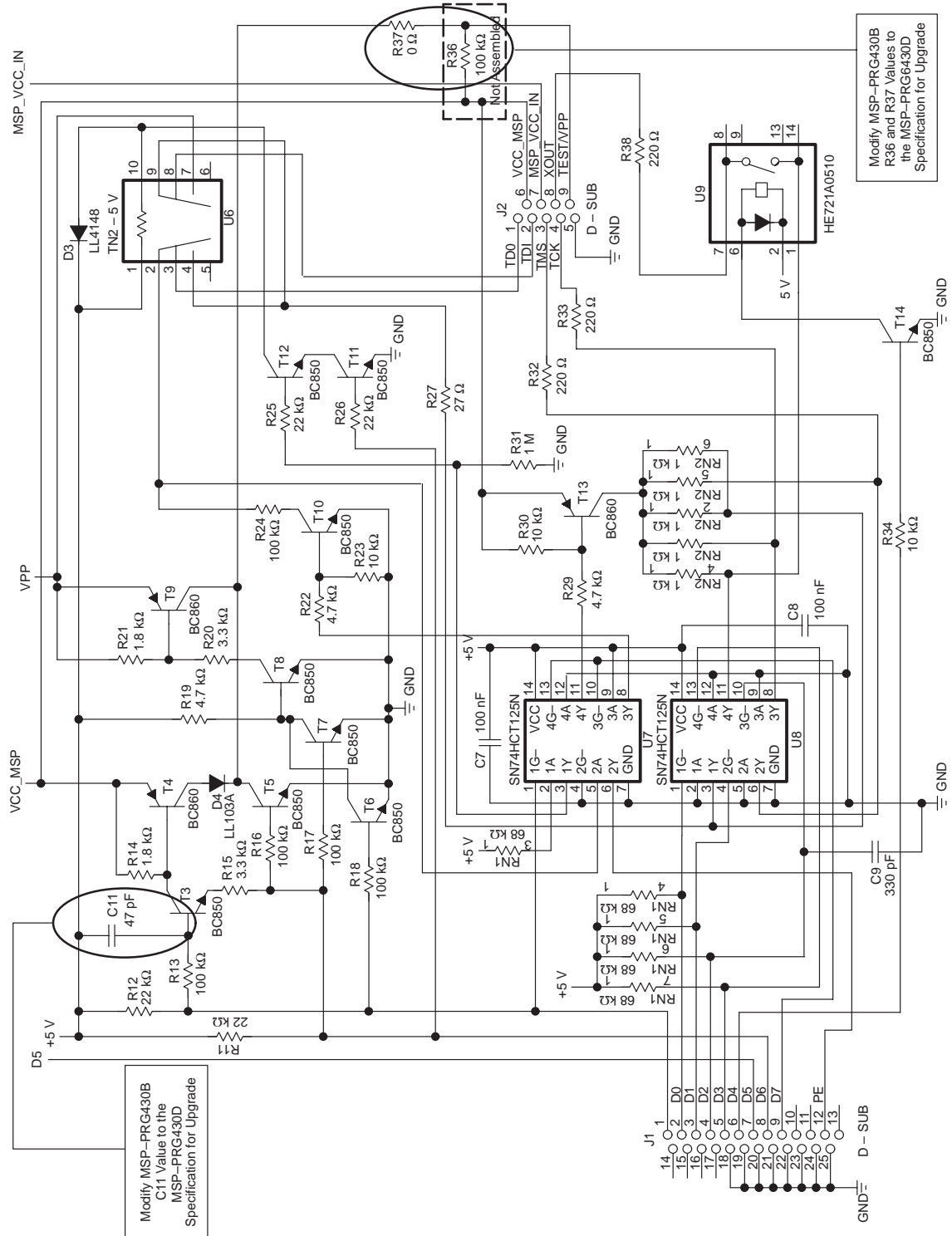


Figure 3–7. MSP-PRG430C Circuit Diagram, Part 2



3.7.1 Upgrade of MSP-PRG430B to MSP-PRG430C

The MSP-PRG430B programming adapter hardware can easily be upgraded to the programming adapter version C. The following modifications describe how an MSP-PRG430B can be upgraded to a MSP-PRG430C:

- ☐ Open the case of the programmer by removing the screw that is located in the center of the top surface, under the label.
- ☐ Remove resistor R36.
- ☐ Add R37. This resistor's value is 0 Ω .
- ☐ Add a capacitor of 47 pF from base of T3 to 5 V.

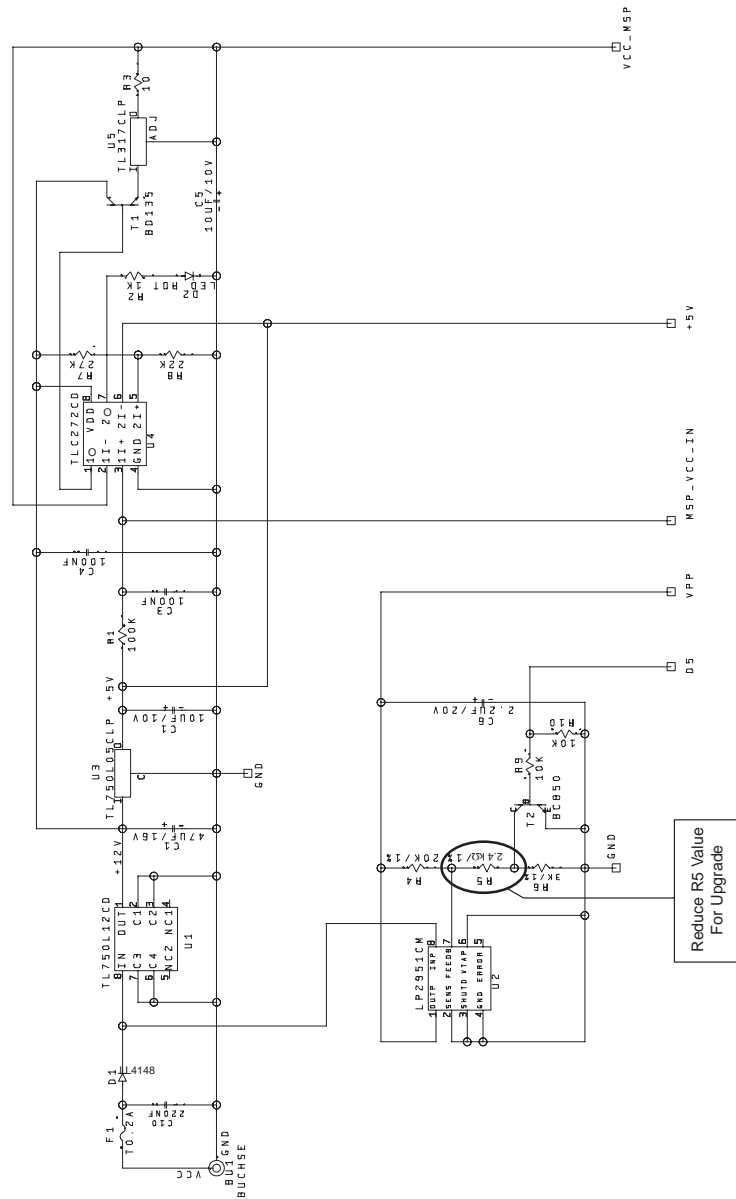
Use the correct software version after the upgrade. Make sure that the target's $\overline{\text{RST}}$ /NMI pin is not connected to pin 8 of the 14-pin connector. After the modification, the Test/VPP signal is applied to this signal line. If the pullup resistor of the MSP-PRG430B was used to keep the $\overline{\text{RST}}$ /NMI terminal high, the target must be modified for the $\overline{\text{RST}}$ /NMI terminal to remain high.

Note:

The upgrades for MSP-PRG430B to MSP-PRG430D and for MSP-PRG430C to MSP-430D are described in paragraph 3.8.1 and 3.8.2.

3.8 Circuit Diagram of MSP-PRG430D

Figure 3–8. MSP-PRG430D Circuit Diagram, Part 1





3.8.1 Upgrade of MSP-PRG430B to MSP-PRG430D

The MSP-PRG430B programming adapter hardware can easily be upgraded to the programming adapter version D. The following modifications describe how an MSP-PRG430B can be upgraded to an MSP-PRG430D:

- ☐ Open the case of the programmer by removing the screw that is located in the center of the top surface, under the label.
- ☐ Remove resistor R36.
- ☐ Add R37. This resistor's value is 0 Ω .
- ☐ Add a capacitor of 47 pF from base of T3 to 5 V.
- ☐ Connect a resistor parallel to R5 to increase VPP in the range of 12.3V to 12.7 V. See also *Read Me PRG*, installed with the latest software version.

Use the correct software version after the upgrade.

3.8.2 Upgrade of MSP-PRG430C to MSP-PRG430D

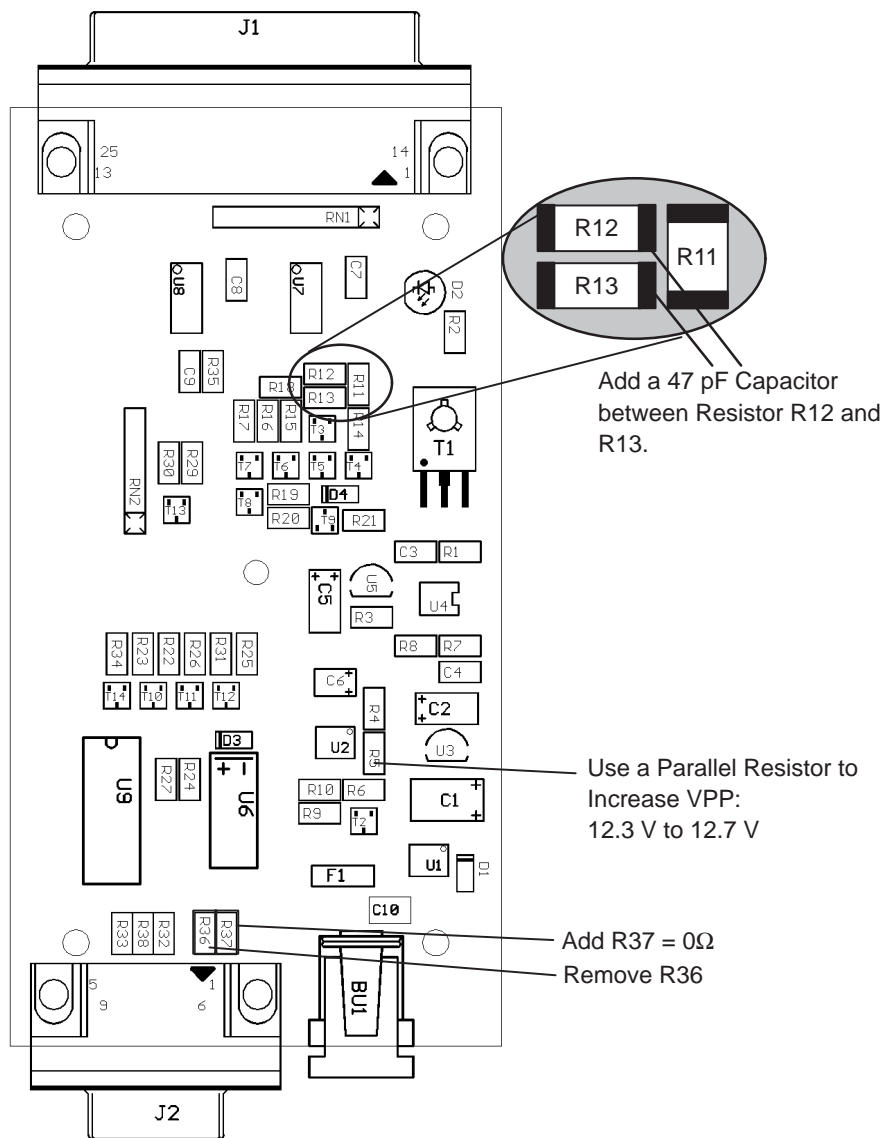
The MSP-PRG430C programming adapter hardware can easily be upgraded to the programming adapter version D. The following modifications describe how an MSP-PRG430B can be upgraded to a MSP-PRG430D:

- ☐ Open the case of the programmer by removing the screw that is located in the center of the top surface, under the label.
- ☐ Connect a resistor parallel to R5 to increase VPP in the range of 12.3V to 12.7 V. See also *Read Me PRG*, installed with the latest software version.

Use the correct software version after the upgrade.

3.9 Location of Components, MSP-PRG430B

Figure 3–10. MSP-PRG430B Components



Note: Do not use J2 Pin 9 as RST/NMI pullup. The RST/NMI pullup must now be at hardware with the target device.

3.10 Interconnection of MSP-PRG430B/430C to MSP430C313DL/430P313SDL, MSP430C311SDL/P315SDL or 'E313FZ

The circuit diagrams in Figure 3–11 show the connections required to program the MSP430P313DL device with programming adapter PRG430B in a separate socket. Since the device is not connected to a power supply in this configuration, the necessary supply comes from the PRG430B.

Figure 3–11. MSP-PRG430B Used to Program the MSP430P313DL Device

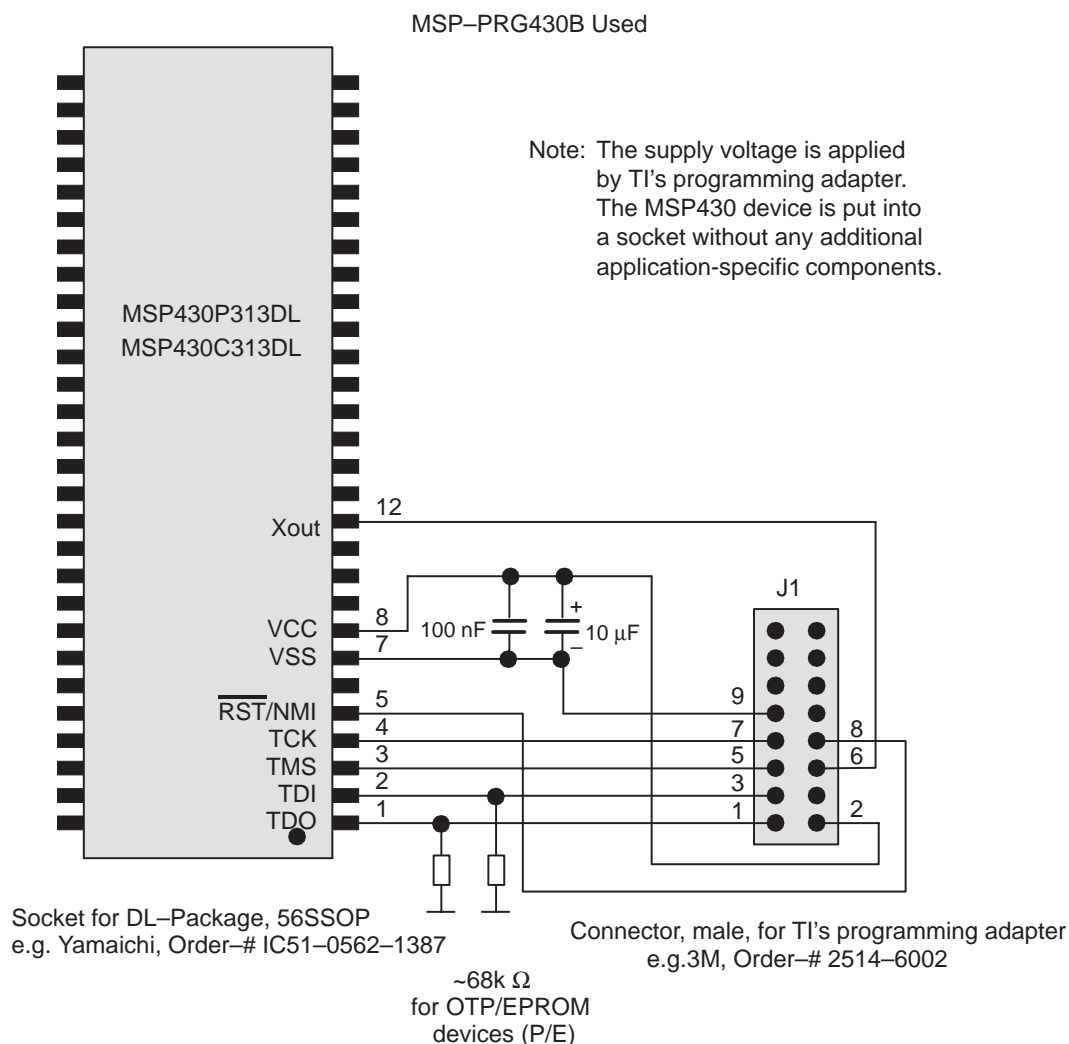
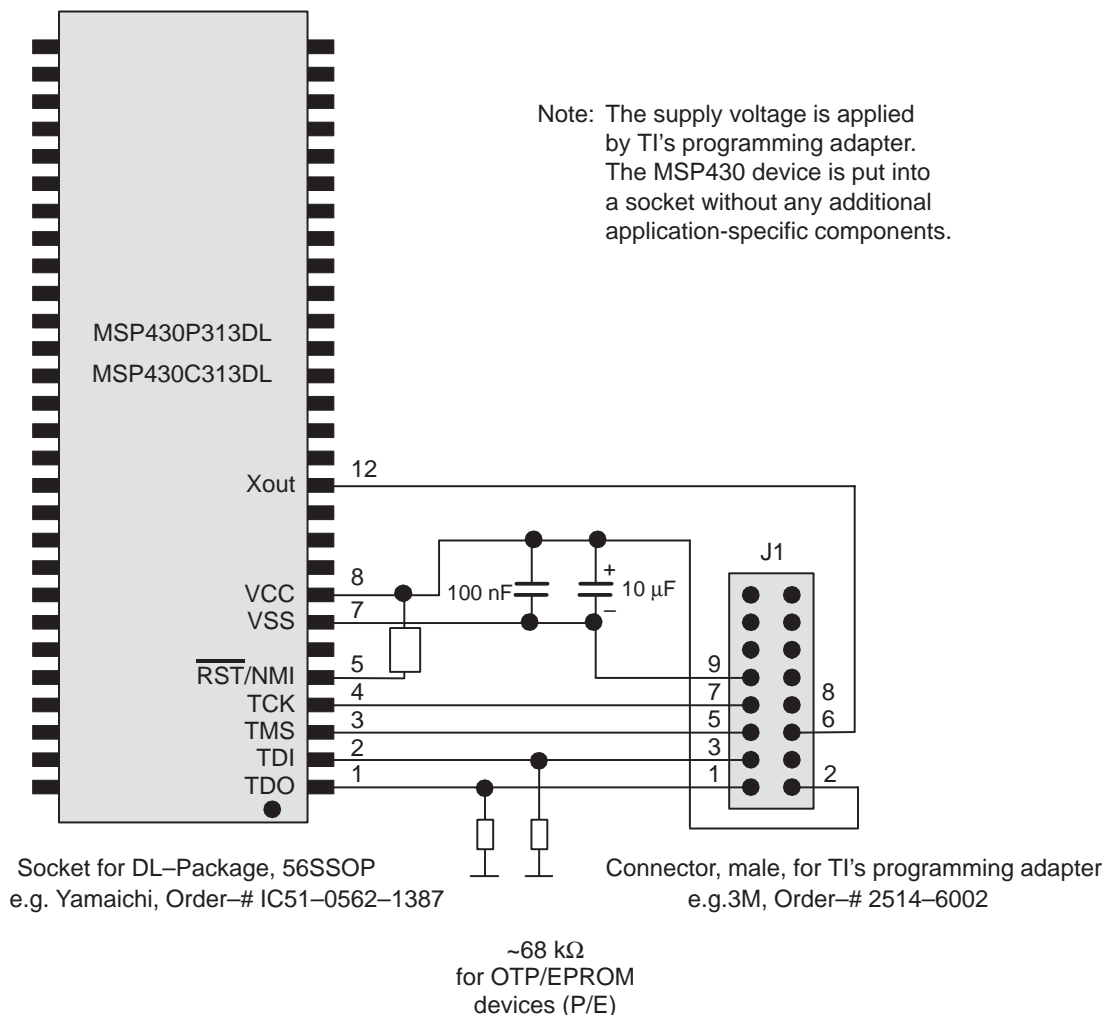


Figure 3–12. MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P313DL Device



The RST/NMI terminal on the MSP430 device is held high either by the resistor in the PRG430B or by an external resistor when PRG430C or PRG430D is used. In a noisy environment, consider using an additional capacitor from RST/NMI to VSS.

The RST/NMI terminal on the MSP430 device is held high either by the resistor in the PRG430B or by an external resistor when PRG430C or PRG430D is used. In a noisy environment, consider using an additional capacitor from RST/NMI to VSS.

The circuit diagrams in Figure 3–13 show the connections required to program the MSP430P315SDL device or to burn its fuse with programming adapter PRG430B in a separate socket. Since the device is not connected to a power supply in this configuration, the necessary supply comes from the PRG430B.

Figure 3–13. MSP-PRG430B Used to Program the MSP430P315SDL Device

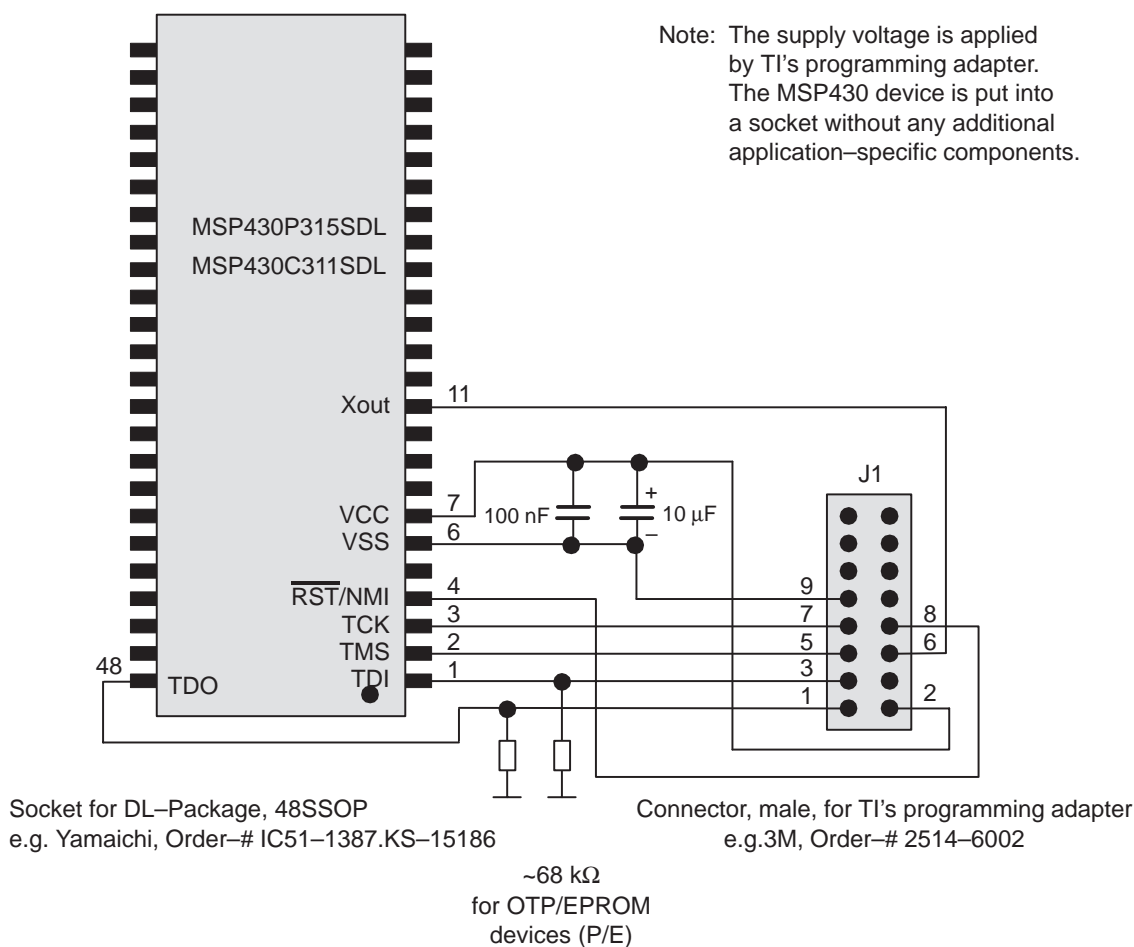
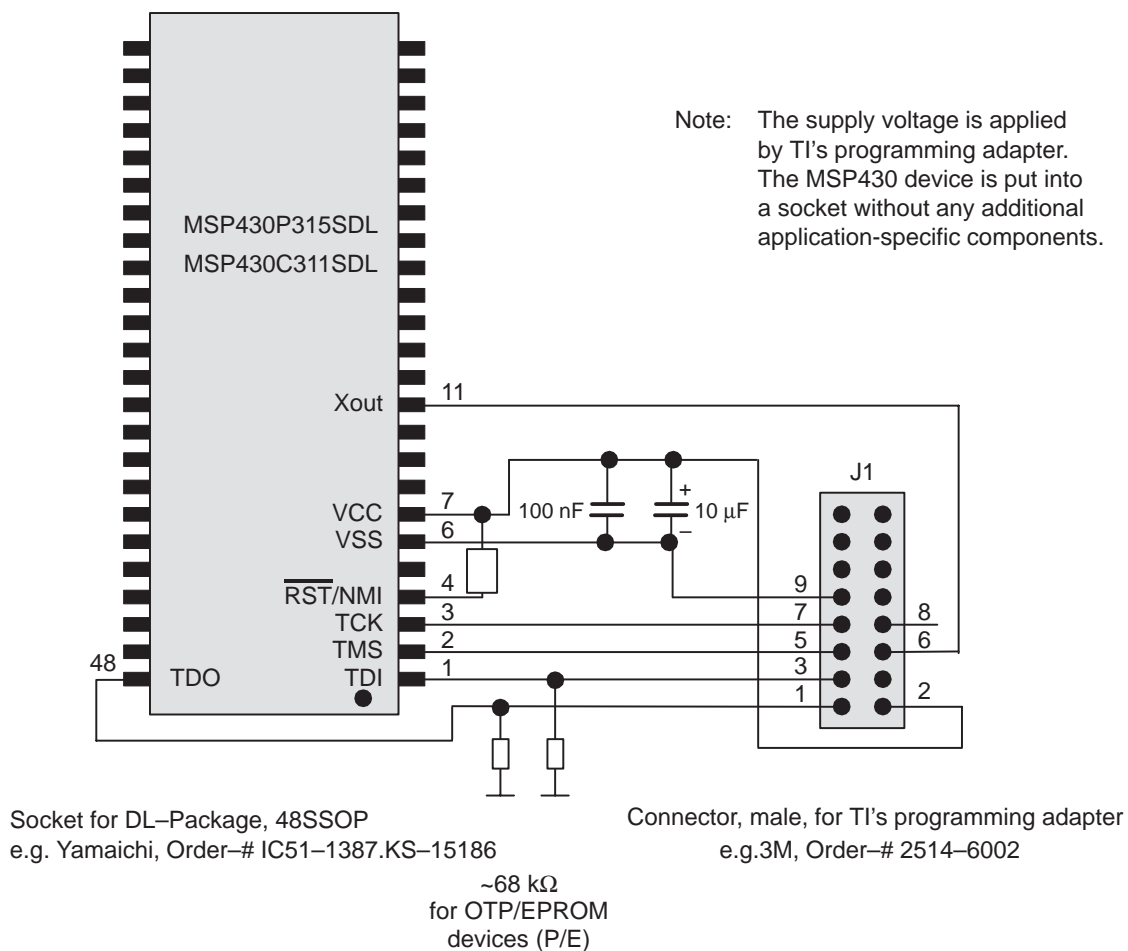


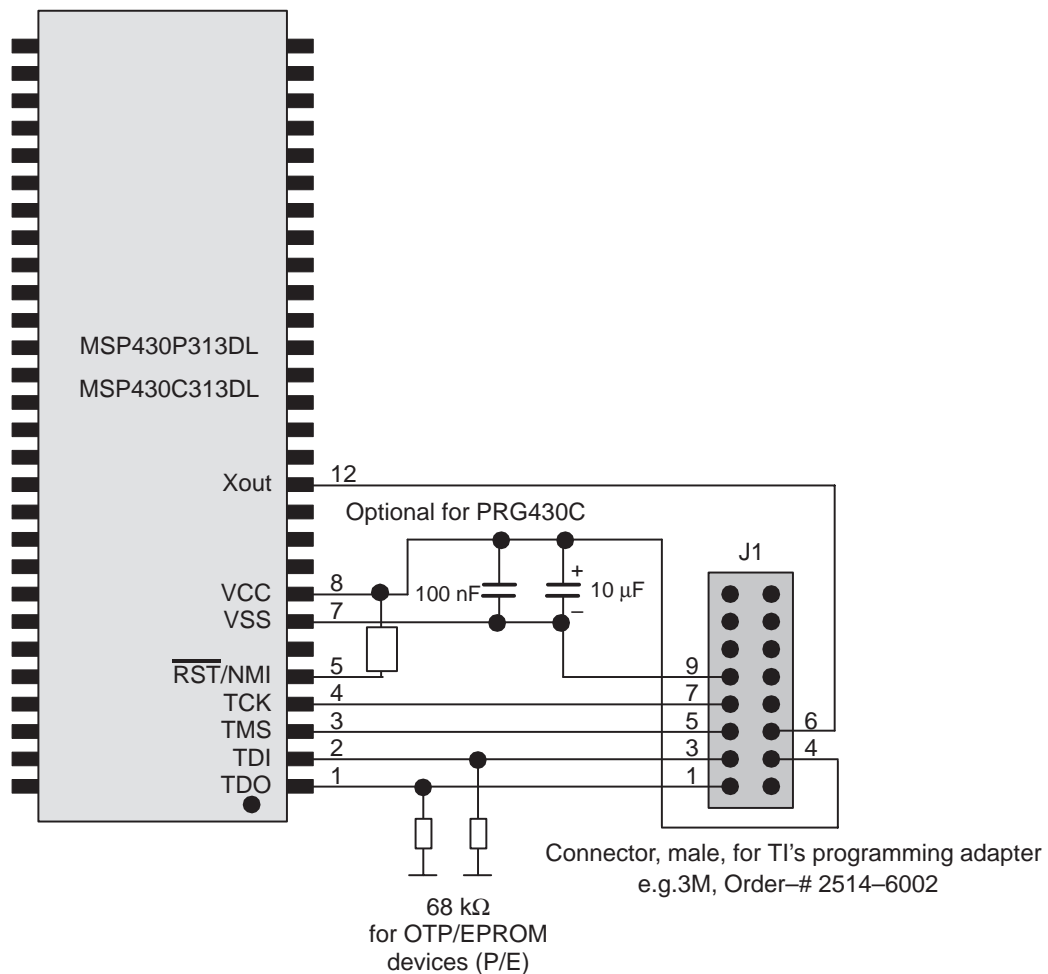
Figure 3–14. MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P315SDL Device



The circuit diagram in Figure 3–15 shows the connections required to program the MSP430P313DL device with programming adapter PRG430B, PRG430C or PRG430D. In this configuration, the device is connected to its own power supply. For example, it is mounted on a PWB with a battery. The signal levels coming from the PRG430B, PRG430C, or PRG430D must be adjusted to the supply voltage applied to the MSP430 device. This level of adjustment is achieved by feedback of the supply voltage of the device into the programming adapter through signal MSP_VCC_IN.

The $\overline{\text{RST}}/\text{NMI}$ pin on the MSP430 device must be held high by the application during access of the device through JTAG with the PRG430B, PRG430C or PRG430D. Any reset event disturbs the proper data sequences and produces unpredictable results.

Figure 3–15. MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P313DL Device



The circuit diagram in Figure 3–16 shows the connections required to program the MSP430E313FZ device with programming adapter PRG430B. The signal levels come from the PRG430B.

The $\overline{\text{RST}}/\text{NMI}$ pin on the MSP430 device is held high by the resistor in the PRG430B or by an external resistor when PRG430C or PRG430D is used. In a noisy environment, using an additional capacitor from RST/NMI to VSS should be considered.

Figure 3–16. MSP-PRG430B Used to Program the MSP430E313FZ Device

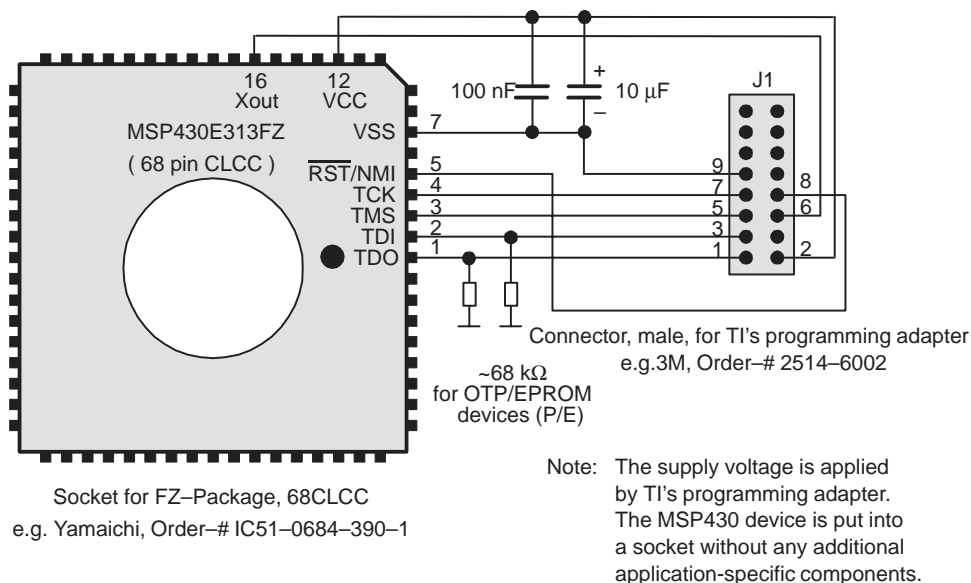
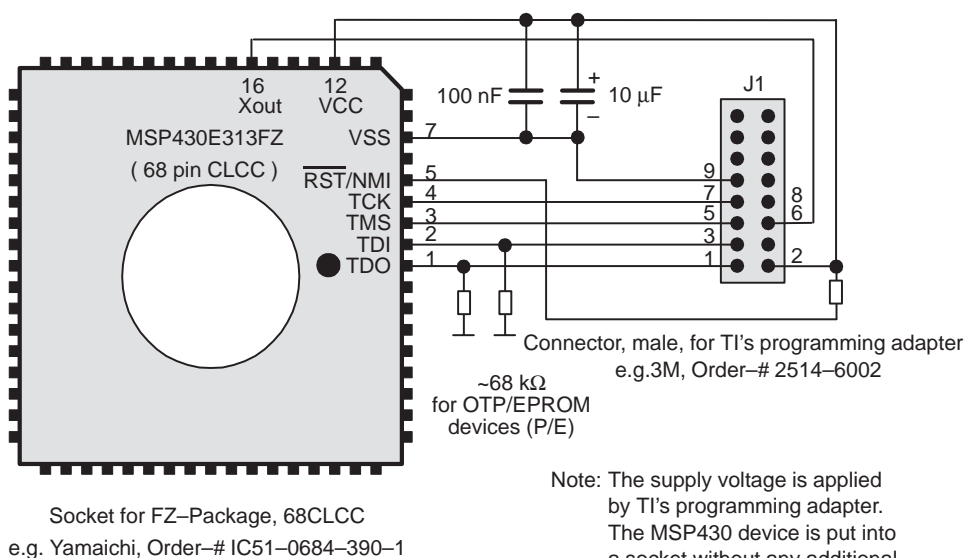


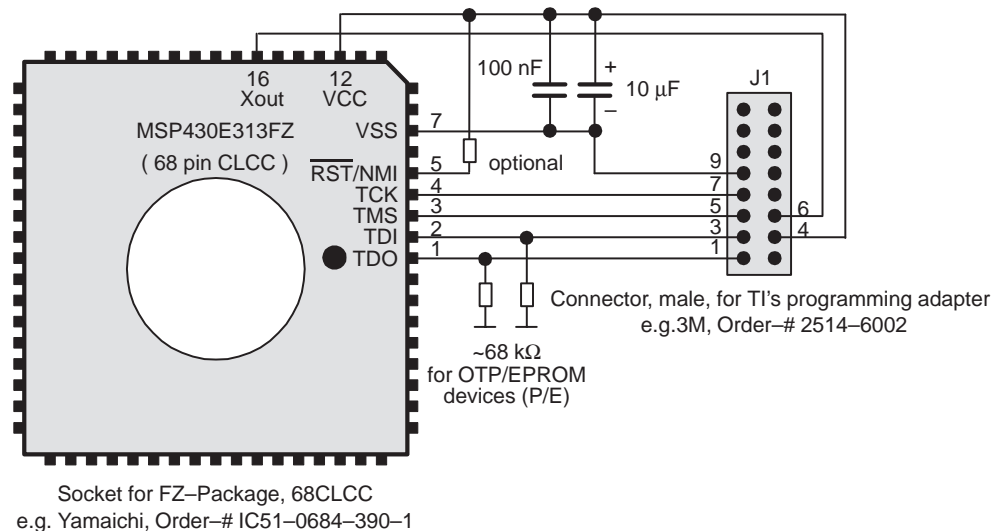
Figure 3–17. MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430E313FZ Device



The circuit diagram in Figure 3–18 shows the connections required to program the MSP430E313FZ device with programming adapter PRG430B, PRG430C, or PRG430D. In this configuration, the device is connected to its own power supply. For example, it is mounted on a PWB with a battery. The signal levels coming from the PRG430B, PRG430C, or PRG430D must be adjusted to the supply voltage applied to the MSP430 device. This level of adjustment is achieved by feedback of the supply voltage of the device into the programming adapter through signal MSP_VCC_IN.

The $\overline{\text{RST/NMI}}$ pin on the MSP430 device must be held high by the application during access of the device through JTAG with the PRG430B, PRG430C or PRG430D. Any reset event disturbs the proper data sequences and produces unpredictable results.

Figure 3–18. MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430E313FZ Device



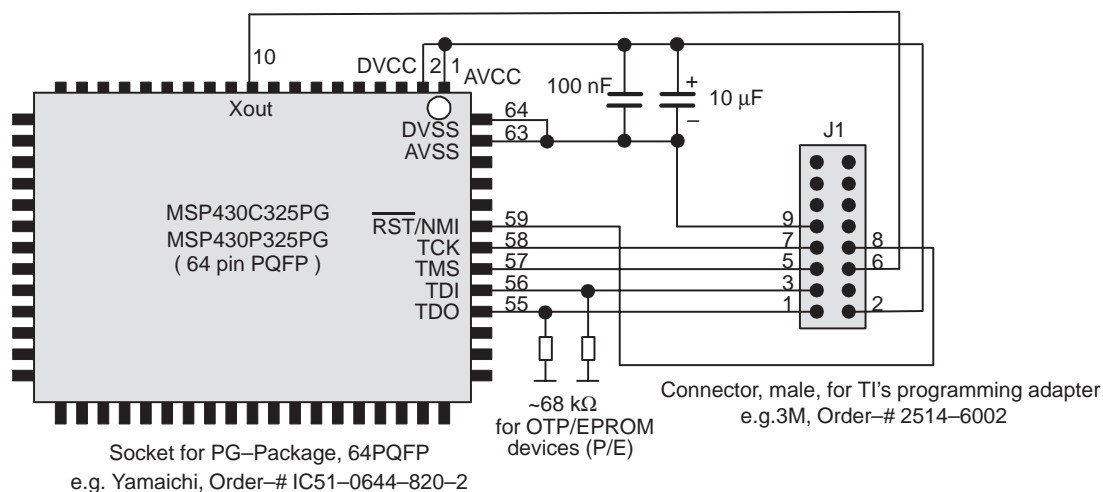
3.11 Interconnection of MSP-PRG430B, or '430C to MSP430C325PG, C325PM MSP430P325PG or 'P325PM

The circuit diagrams in Figure 3–19 show the connections required to program the MSP430C325PG, MSP430P325PG, or MSP430P325APG device with programming adapter PRG430B, PRG430C or PRG430D in a separate socket. Since the device is not connected to a power supply in this configuration, the necessary supply comes from the PRG430B, PRG430C, or PRG430D.

The $\overline{\text{RST/NMI}}$ pin on the MSP430 device is held high by the resistor in the PRG430B or by an external resistor when PRG430C or PRG430D is used. In a noisy environment, consider using an additional capacitor from RST/NMI to VSS.

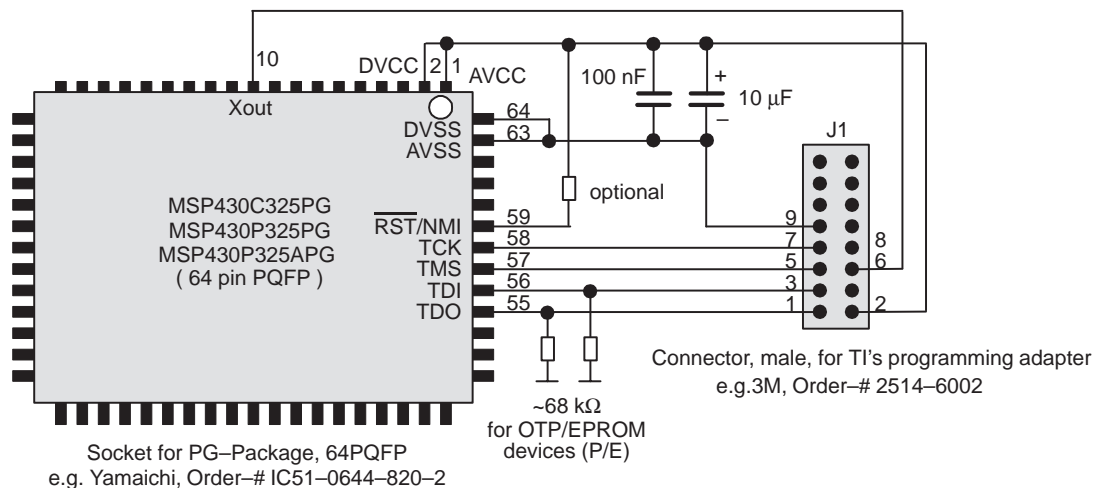
Ensure that both positive terminals AVCC and DVCC are connected. In addition, ensure that both negative terminals AVSS and DVSS are connected.

Figure 3–19. MSP-PRG430B Used to Program theMSP430P325PG Device



Note: The supply voltage is applied by TI's programming adapter. The MSP430 device is put into a socket without any additional application-specific components.

Figure 3–20. MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P325PG or MSP430P325APG Device



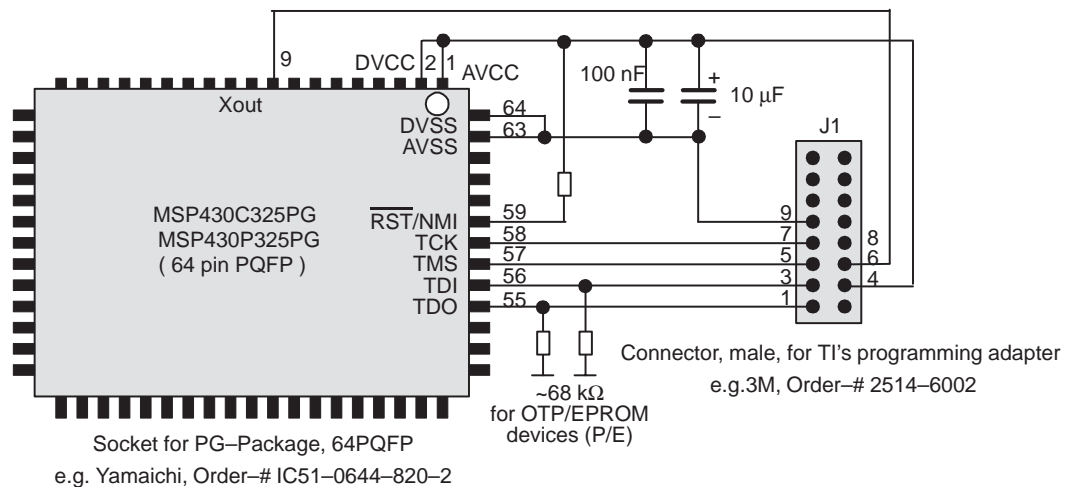
Note: The supply voltage is applied by TI's programming adapter. The MSP430 device is put into a socket without any additional application-specific components.

The circuit diagrams in Figure 3–21 show the connections required to program the MSP430C325PG, MSP430P325PG, or MSP430P325APG device with programming adapter PRG430B, PRG430C, or PRG430D in a separate socket. Since the device is connected to a power supply in this configuration, the signal levels coming from the PRG430D need to be adjusted to the supply voltage applied to the MSP430 device. This level adjustment is achieved by feedback of the supply voltage of the device into the programming adapter through signal MSP_VCC_IN.

The $\overline{\text{RST}}/\text{NMI}$ pin on the MSP430 device must be held high by the application during access of the device through JTAG with the PRG430B, PRG430C, or PRG430D. Any reset event disturbs the proper data sequences and produces unpredictable results.

Ensure that both positive terminals AVCC and DVCC are connected. In addition, ensure that both negative terminals AVSS and DVSS are connected.

Figure 3–21. MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P325PG or MSP430P325APG Device

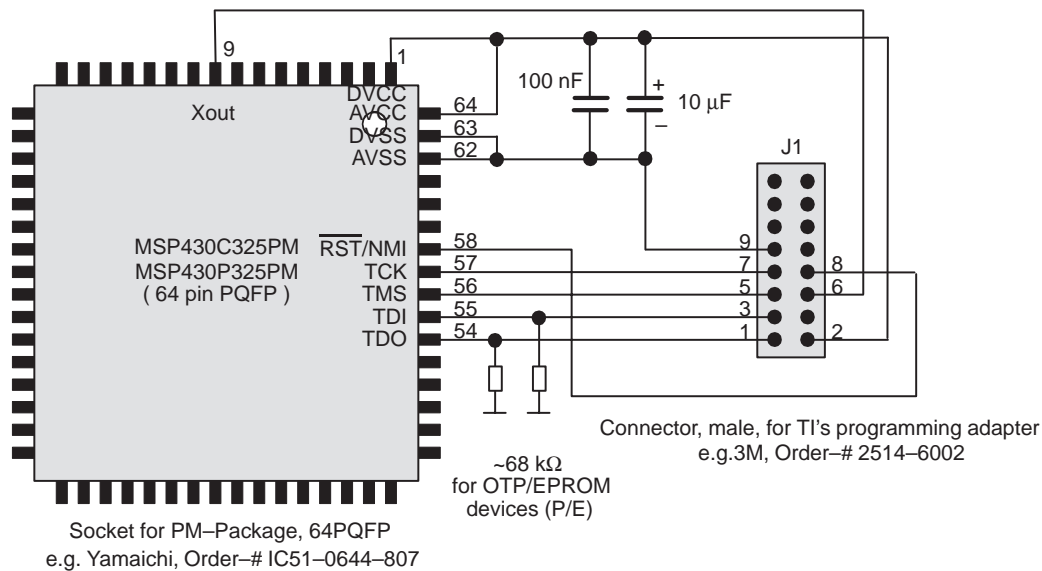


The circuit diagrams in Figure 3–22 show the connections required to program the MSP430C325PM, MSP430P325PM, or MSP430P325APM device with programming adapter PRG430B, PRG430C, or PRG430D in a separate socket. Since the device is not connected to a power supply in this configuration, the necessary supply comes from the PRG430B, PRG430C, or PRG430D.

The $\overline{\text{RST/NMI}}$ pin on the MSP430 device is held high by the resistor in the PRG430B or by an external resistor when PRG430C or PRG430D is used. In a noisy environment, consider using an additional capacitor from RST/NMI to VSS.

Ensure that both positive terminals AVCC and DVCC are connected. In addition, ensure that both negative terminals AVSS and DVSS are connected.

Figure 3–22. MSP-PRG430B Used to Program the MSP430C325PM or MSP430P325PM Device

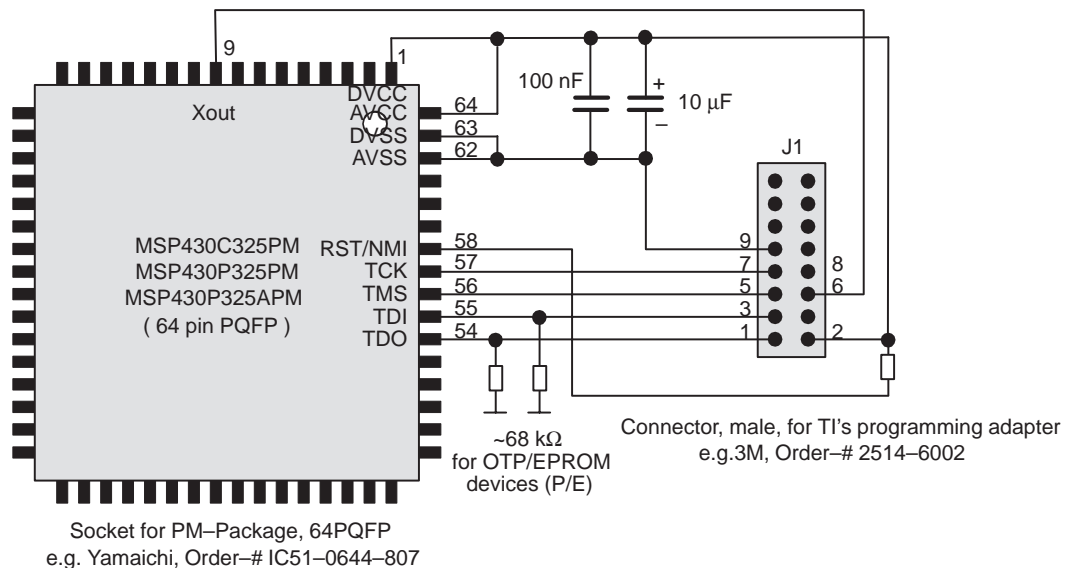


The circuit diagrams in Figure 3–23 show the connections required to program the MSP430C325PM, MSP430P325PM, or MSP430P325APM device with programming adapter PRG430B, PRG430C, or PRG430D in a separate socket. Since the device is not connected to a power supply in this configuration, the necessary supply comes from the PRG430B, PRG430C, or PRG430D.

The $\overline{\text{RST/NMI}}$ pin on the MSP430 device is held high by the resistor in the PRG430B or by an external resistor when PRG430C or PRG430D is used. In a noisy environment, consider using an additional capacitor from RST/NMI to VSS.

Ensure that both positive terminals AVCC and DVCC are connected. In addition, ensure that both negative terminals AVSS and DVSS are connected.

Figure 3–23. MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430P325PM or MSP430P325APM Device



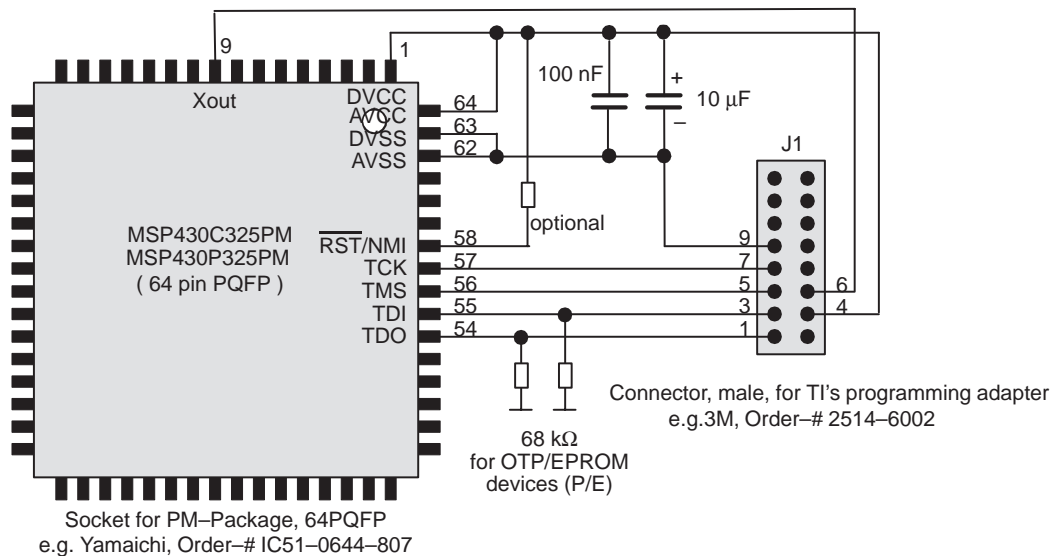
Note: The supply voltage is applied by TI's programming adapter. The MSP430 device is put into a socket without any additional application-specific components.

The circuit diagram in Figure 3–24 shows the connections required to program the MSP430P325PM or MSP430P325APM device with programming adapter PRG430B or PRG430C or PRG430D. In this configuration, the device is connected to its own power supply. For example, it is mounted on a PWB with a battery. The signal levels coming from the PRG430B or PRG430C or PRG430D must be adjusted to the supply voltage applied to the MSP430 device. This level adjustment is achieved by feedback of the supply voltage of the device into the programming adapter through signal MSP_VCC_IN.

The $\overline{\text{RST/NMI}}$ terminal on the MSP430 device must be held high by the application during access of the device through JTAG with the PRG430B or PRG430C. Any reset event disturbs the proper data sequences and produces unpredictable results.

Ensure that both positive terminals AVCC and DVCC are connected. In addition, ensure that both negative terminals AVSS and DVSS are connected.

Figure 3–24. MSP-PRG430B or MSP-PRG430C or PRG430D Used to Program the MSP430P325PM or MSP430P325APM Device

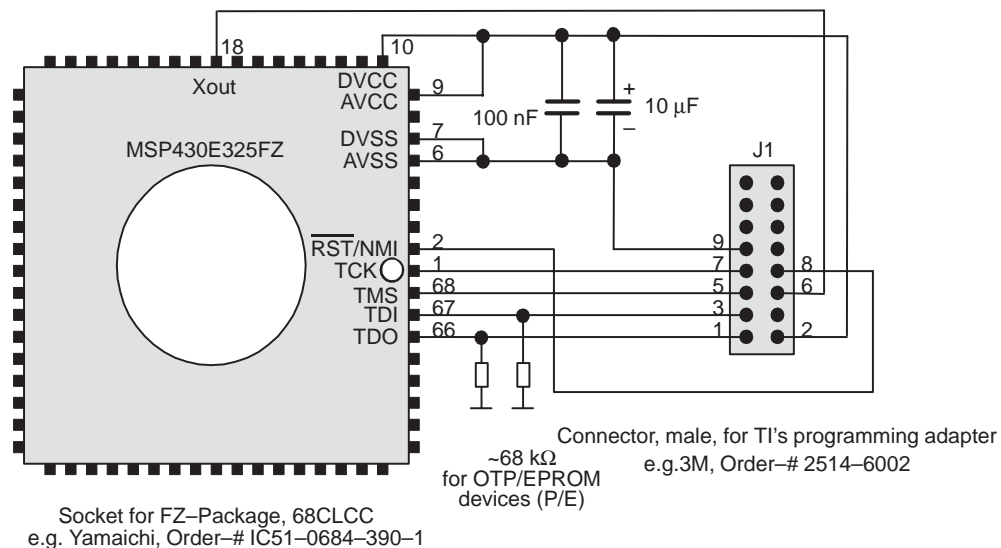


The circuit diagrams in Figure 3–24 show the connections required to program the MSP430E325FZ device with programming adapter PRG430B, PRG430C, or PRG430D in a separate socket. Since the device is not connected to a power supply in this configuration, the necessary supply comes from the PRG430B, PRG430C, or PRG430D.

The $\overline{\text{RST/NMI}}$ terminal on the MSP430 device is held high by the resistor in the PRG430B or by an external resistor when PRG430C or PRG430D is used. In noisy environments, consider using an additional capacitor from RST/NMI to VSS.

Ensure that both positive terminals AVCC and DVCC are connected. In addition, ensure that both negative terminals AVSS and DVSS are connected.

Figure 3–25. MSP-PRG430B Used to Program the MSP430E325FZ Device



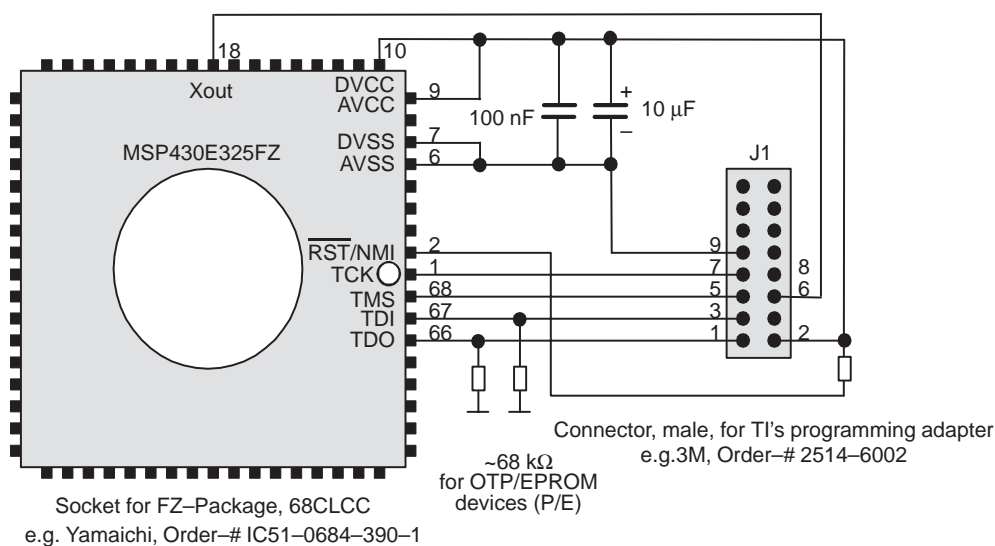
Note: The supply voltage is applied by TI's programming adapter. The MSP430 device is put into a socket without any additional application-specific components.

The circuit diagrams in Figure 3–26 show the connections required to program the MSP430E325FZ device with programming adapter PRG430B, PRG430C, or PRG430D in a separate socket. Since the device is not connected to a power supply in this configuration, the necessary supply comes from the PRG430B, PRG430C, or PRG430D.

The $\overline{\text{RST/NMI}}$ terminal on the MSP430 device is held high by the resistor in the PRG430B or by an external resistor when PRG430C or PRG430D is used. In noisy environments, consider using an additional capacitor from RST/NMI to VSS.

Ensure that both positive terminals AVCC and DVCC are connected. In addition, ensure that both negative terminals AVSS and DVSS are connected.

Figure 3–26. MSP-PRG430B or MSP-PRG430C Used to Program the MSP430E325FZ Device



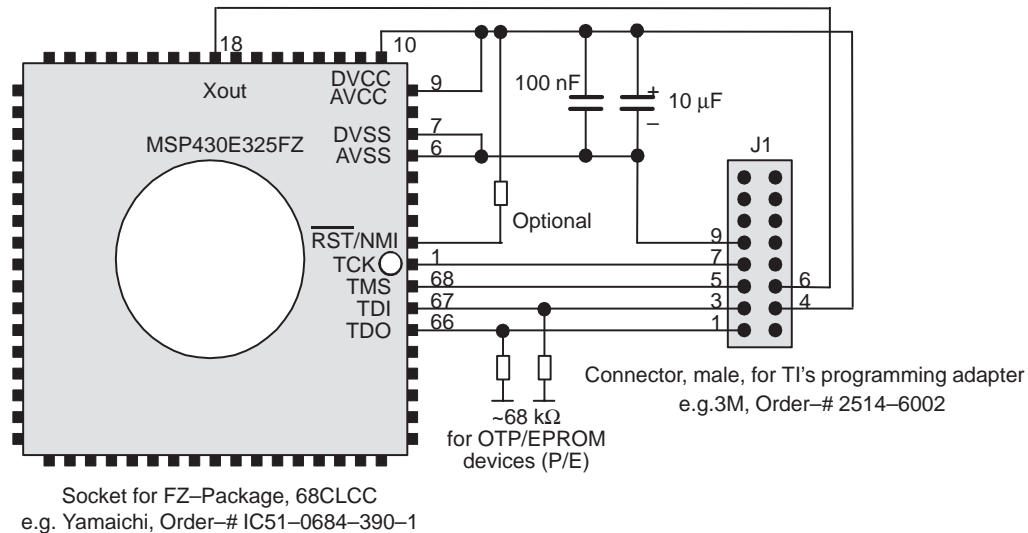
Note: The supply voltage is applied by TI's programming adapter. The MSP430 device is put into a socket without any additional application-specific components.

The circuit diagram in Figure 3–27 shows the connections required to program the MSP430E325FZ device with programming adapter PRG430B, PRG430C or PRG430D. In this configuration, the device is connected to its own power supply. For example, it is mounted on a PWB with a battery. The signal levels coming from the PRG430B, PRG430C, or PRG430D must be adjusted to the supply voltage applied to the MSP430 device. This level adjustment is achieved by feedback of the supply voltage of the device into the programming adapter through signal MSP_VCC_IN.

The $\overline{\text{RST/NMI}}$ terminal on the MSP430 device must be held high by the application during access of the device through JTAG with the PRG430B, PRG430C, or PRG430D. Any reset event disturbs the proper data sequences and produces unpredictable results.

Ensure that both positive terminals AVCC and DVCC are connected. In addition, ensure that both negative terminals AVSS and DVSS are connected.

Figure 3–27. MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430E325FZ Device



3.12 Interconnection of MSP-PRG430B/430C/430D to MSP430C336PJM/337PJM or MSP430E337CQFP

The circuit diagrams in Figure 3–28 and Figure 3–29 show the connections required to program the MSP430C336PJM, MSP430P337PJM, or MSP430E337CQFP device with programming adapter PRG430B, PRG430C or PRG430D in a separate socket. Since the device is not connected to a power supply in this configuration, the necessary supply comes from the PRG430B, PRG430C, or PRG430D.

The $\overline{\text{RST/NMI}}$ terminal on the MSP430 device is held high by the resistor in the PRG430B or by an external resistor when PRG430C or PRG430D is used. In a noisy environment, consider using an additional capacitor from RST/NMI to VSS.

Ensure that the two positive terminals, VCC1 and VCC2, as well as the three negative terminals, VSS1, VSS2, and VSS3, are connected.

Figure 3–28. MSP-PRG430B Used to Program the MSP430P325PM or MSP430P325APM Device

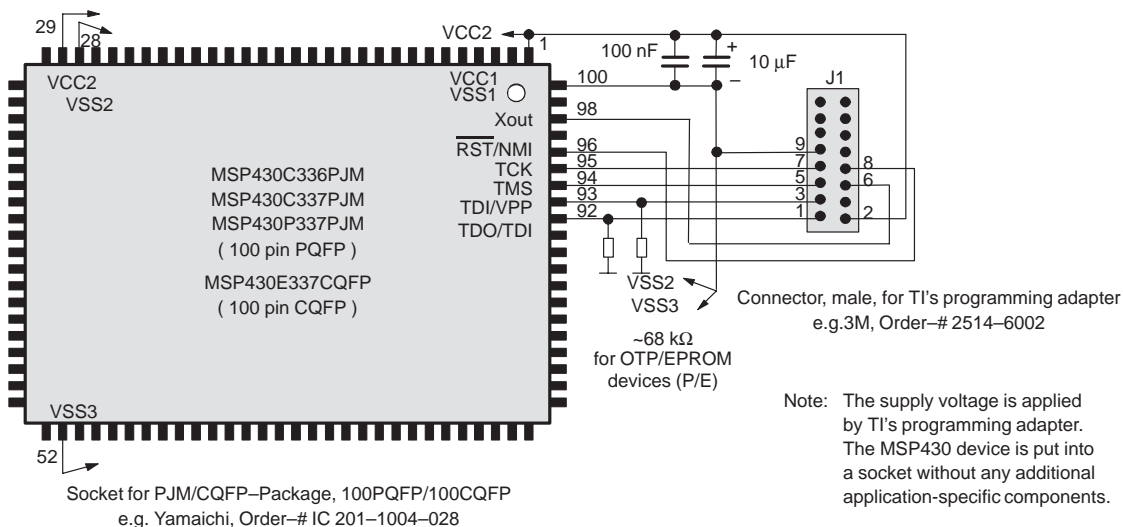
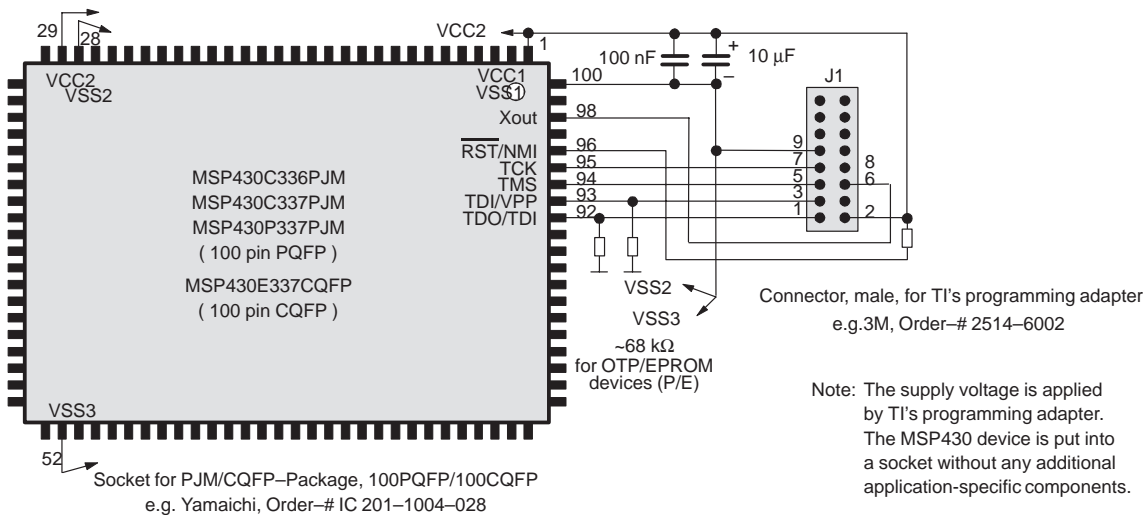


Figure 3–29. MSP-PRG430B or MSP-PRG430C or MSP-PRG430D Used to Program the MSP430C336PJM, MSP430P337PJM, or the MSP430E337 Device

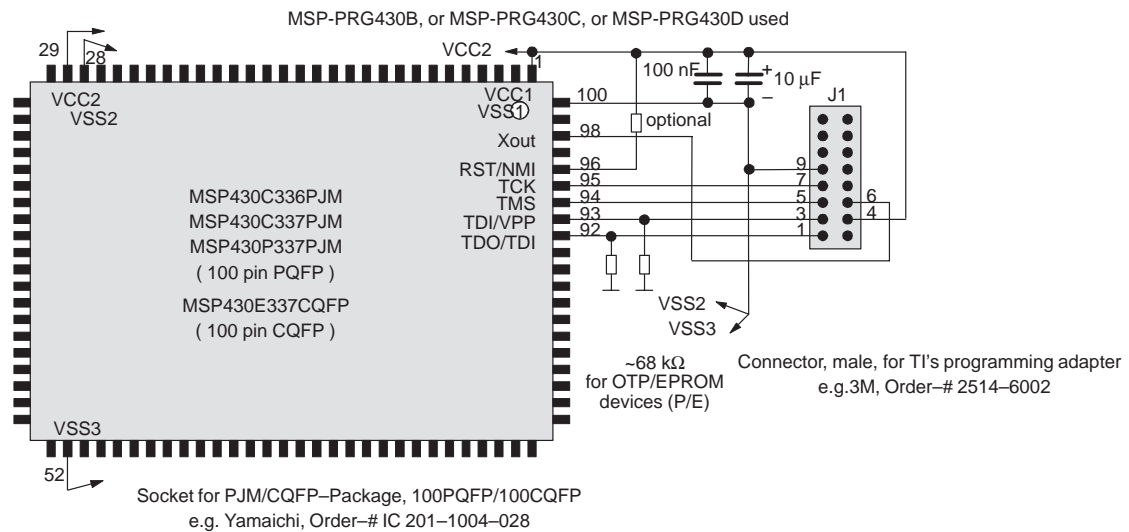


The circuit diagram in Figure 3–30 shows the connections required to program the MSP430C336PJM, MSP430P337PJM, or MSP430E337 device with programming adapter PRG430B, PRG430C, or PRG430D. In this configuration, the device is connected to its own power supply. For example, it is mounted on a PWB with a battery. The signal levels coming from the PRG430B, PRG430C, or PRG430D must be adjusted to the supply voltage applied to the MSP430 device. This level adjustment is achieved by feedback of the supply voltage of the device into the programming adapter through signal MSP_VCC_IN.

The $\overline{\text{RST}}/\text{NMI}$ terminal on the MSP430 device must be held high by the application during access of the device through JTAG with the PRG430B, PRG430C, or PRG430D. Any reset event disturbs the proper data sequences and produces unpredictable results.

Ensure that the two positive terminals, VCC1 and VCC2, as well as the three negative terminals, VSS1, VSS2, and VSS3, are connected.

Figure 3–30. MSP-PRG430B or MSP-PRG430C Used to Program the MSP430C336PJM, MSP430P337PJM, or the MSP430E337 Device



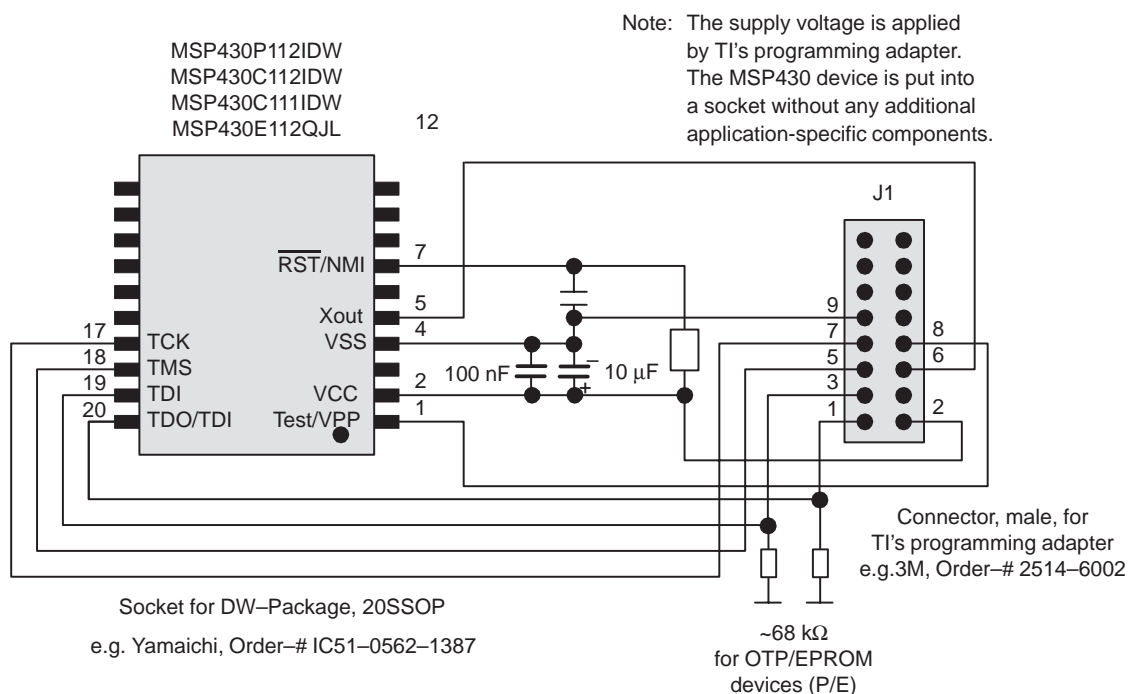
3.13 Interconnection of MSP-PRG430C to MSP430C111DW, MSP430C112DW, MSP430P112DW, or MSP430E112JL

The circuit diagram in Figure 3–31 shows the connections required to program with the programming adapter PRG430B, or PRG430C, or PRG430D in a separate socket. Since the device is not connected to a power supply in this configuration, the necessary supply comes from the PRG430C or PRG430D.

The $\overline{\text{RST/NMI}}$ terminal on the MSP430 device is held high by an external resistor. In a noisy environment, consider using an additional capacitor from RST/NMI to VSS.

Figure 3–31. MSP-PRG430D or Modified MSP-PRG430C or Modified MSP-PRG430B

MSP-PRG43A can not be used



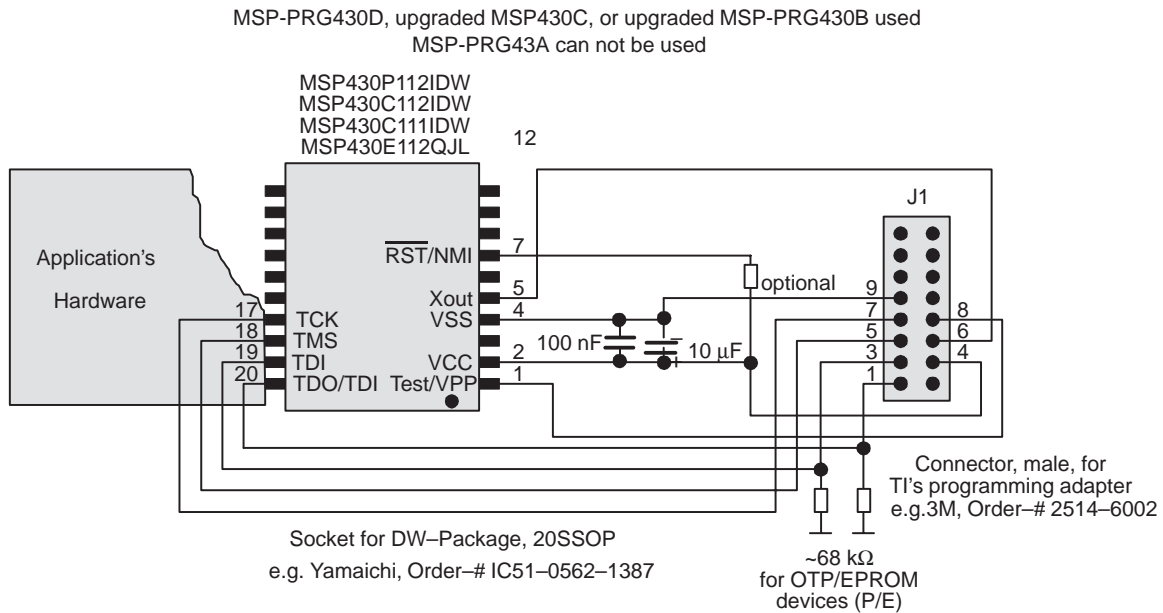
Note: The upgraded programming adapter MSP-PRG430B, upgraded MSP-PRG430C, and the MSP-PRG430D are mechanically identical. The VPP of the MSP-PRG430D is typically one volt higher than that of MSP-PRG430B or MSP-PRG430C.

The circuit diagram in Figure 3–32 shows the connections required to program and to burn the fuse of MSP430x11x family devices with programming adapter upgraded MSP-PRG430B, upgraded PRG430C, or PRG430D. In this configuration, the device is connected to its own power supply. For example, it is mounted on a PWB with a battery. The signal levels coming from the programming adapter must be adjusted to the supply voltage applied to the MSP430 device. This level adjustment is achieved by feedback of the supply voltage of the device into the programming adapter through signal MSP_VCC_IN.

The $\overline{\text{RST/NMI}}$ pin on the MSP430 device must be held high by the application during access of the device through JTAG with the upgraded PRG430B, upgraded PRG430C or PRG430D. Any reset event disturbs the proper data sequences and produces unpredictable results.

Special attention must be given to the design for the four JTAG pins, TDO/TDI, TDI, TMS, and TCK, since they are shared between the application's hardware and the programming adapter. The programming adapter should be able to drive the device correctly, but the application should continue working properly.

Figure 3–32. MSP-PRG430C or Upgraded MSP-PRG430





EPROM Programming

This chapter describes the MSP430 EPROM module. The EPROM module is erasable with ultraviolet light and electrically programmable. Devices with an EPROM module are offered in a windowed package for multiple programming and in an OTP package for one-time programmable devices.

Note:

Small improvements to the chapter printed in the *Architecture Guide*, issue 1996, were implemented.

Topic	Page
4.1 EPROM Operation	4-2
4.2 FAST Programming Algorithm	4-4
4.3 Programming an EPROM Module Through a Serial Data Link Using the JTAG Feature	4-5
4.4 Programming an EPROM Module With Controller's Software	4-6
4.5 Code	4-8

4.1 EPROM Operation

The CPU acquires data and instructions from the EPROM. When the programming voltage is applied to the TDI/VPP terminal, the CPU can also write to the EPROM module. The process of reading the EPROM is identical to the process of reading from other internal peripheral modules. Both programming and reading can occur on byte or word boundaries.

4.1.1 Erasure

The entire EPROM may be erased before programming begins. Erase the EPROM module by exposing the transparent window to ultraviolet light.

Note: EPROM exposed to ambient light (1)

Since normal ambient light contains the correct wavelength for erasure, cover the transparent window with an opaque label when programming a device. Do not remove the label until it has to be erased. Any useful data in the EPROM module must be reprogrammed after exposure to ultraviolet light.

The data in the EPROM module can be programmed serially through the integrated JTAG feature, or through software included as a part of the application software. The JTAG implementation features an internal mechanism for security purposes provided by the implemented fuse. Once the security fuse is activated, the device cannot be accessed through the JTAG functions. The JTAG is permanently operating in the by-pass mode.

Refer to the appropriate data sheet for more information on the fuse implementation.

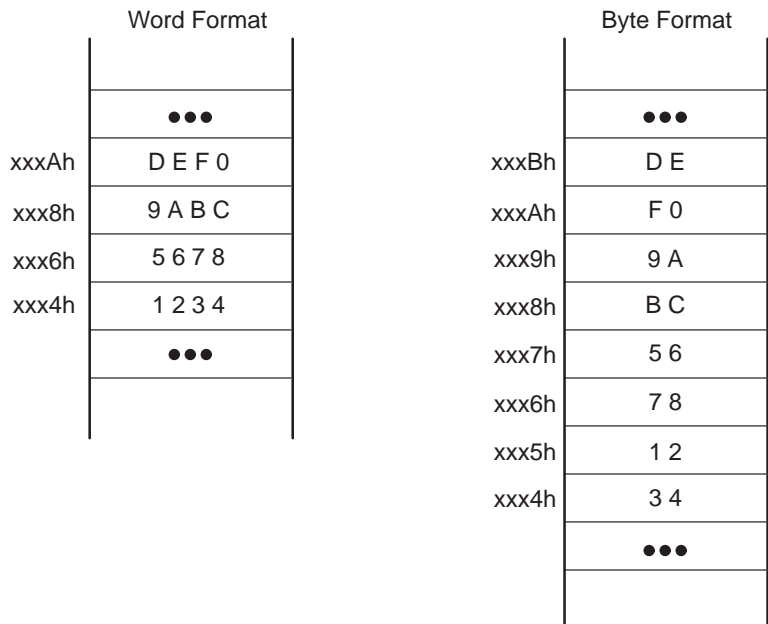
4.1.2 Programming Methods

The application must provide an external voltage supply to the TDI/VPP terminal ('31x, '32x, and '33x configurations) or to the Test/VPP terminal ('11x configuration) to provide the necessary voltage and current for programming. The minimum programming time is noted in the electrical characteristics of the device data sheets.

The EPROM control register EPCTL controls the EPROM programming, once the external voltage is supplied. The erase state is a 1. When EPROM bits are programmed, they are read as 0.

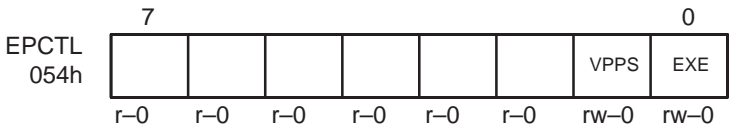
The programming of the EPROM module can be done for single bytes, words, blocks of individual length, or the entire module. All bits that have a final level of 0 must be erased before the EPROM module is programmed. The programming can be done on single devices or even in-system. The supply voltage should be in the range required by the device data sheet but at least the maximum supply voltage of the target application. The levels on the JTAG terminals are defined in the device data sheet, and are usually CMOS levels.

Example 4–1. MSP430 On-Chip Program Memory Format



4.1.3 EPROM Control Register EPCTL

Figure 4–1. EPROM Control Register EPCTL



For bit 0, the executable bit EXE initiates and ends the programming to the EPROM module. The external voltage must be supplied to the TDI/VPP or Test/VPP before the EXE bit is set. The timing conditions are noted in the data sheets.

Note:

The programming voltage is applied to TDI/VPP terminal in '31x, '32x, and '33x configurations. The programming voltage is applied to Test/VPP terminal in '110 configurations.

For bit 1, when the VPPS bit is set, the external programming voltage is connected to the EPROM module. The VPPS bit must be set before the EXE bit is set. It can be reset together with the EXE bit. The VPPS bit must not be cleared between programming operations.

Note:

Ensure that no VPP is applied to the programming voltage pin (TDI/VPP or Test/VPP) when the software in the device is executed or when the JTAG is not fully controlled. Otherwise, an undesired write operation may occur.

4.1.4 EPROM Protect

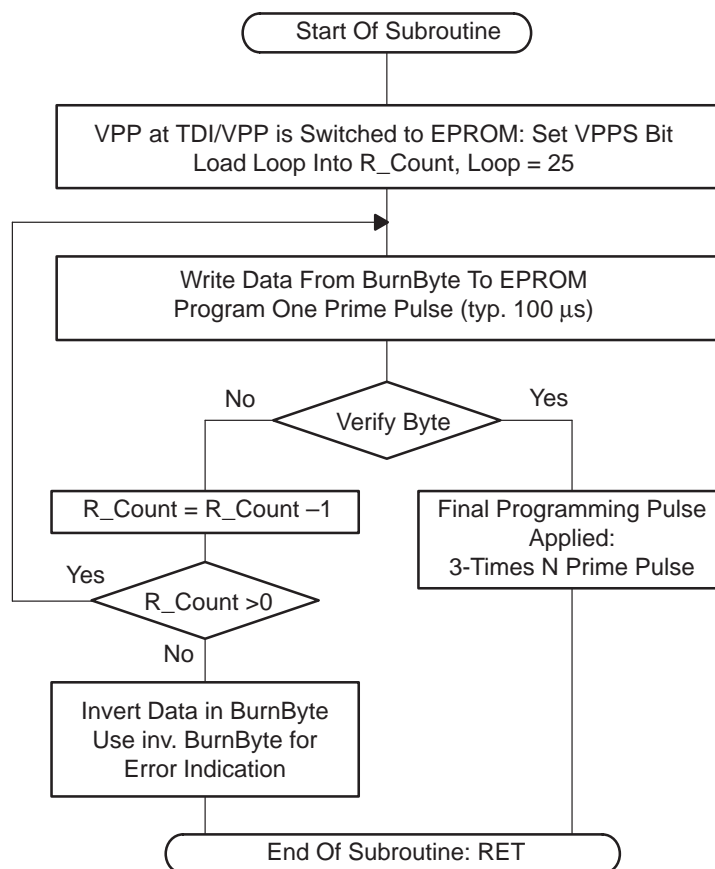
The EPROM access through the serial test and programming interface JTAG can be inhibited when the security fuse is activated. The security fuse is activated by serial instructions shifted into the JTAG. Activating the fuse is not reversible and any access to the internal system is disrupted. The by-pass function described in the standard IEEE 1149.1 is active.

4.2 FAST Programming Algorithm

The FAST programming cycle is normally used to program the data into the EPROM. A programmed logical 0 can be erased only by ultraviolet light.

Fast programming uses two types of pulses: prime and final. The length of the prime pulse is typically 100 μ s (see the latest datasheet). After each prime pulse, the programmed data are verified. If the verification fails 25 times, the programming operation was false. If correct data are read, the final programming pulse is applied. The final programming pulse is 3 times the number of prime pulses applied.

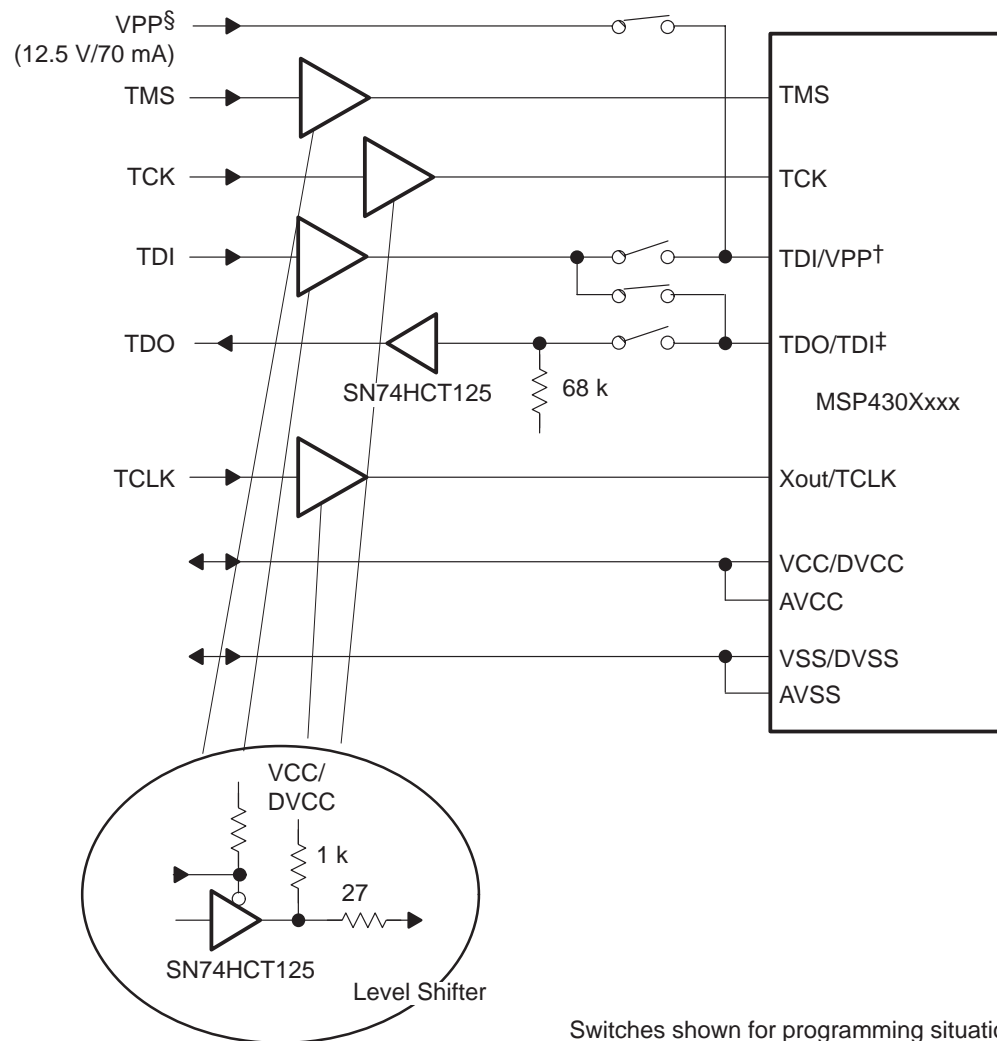
Example 4–2. Fast Programming Subroutine



4.3 Programming an EPROM Module Through a Serial Data Link Using the JTAG Feature

The hardware interconnection of the JTAG terminals is established through four separate terminals, plus the ground or VSS reference level. The JTAG terminals are TMS, TCK, TDI/VPP, and TDO/TDI.

Figure 4–2. EPROM Programming With Serial Data Link



[†] TDI in standard mode, VPP input during programming

[‡] TDO in standard mode, data input TDI during programming

[§] See electrical characteristics in the latest data sheet

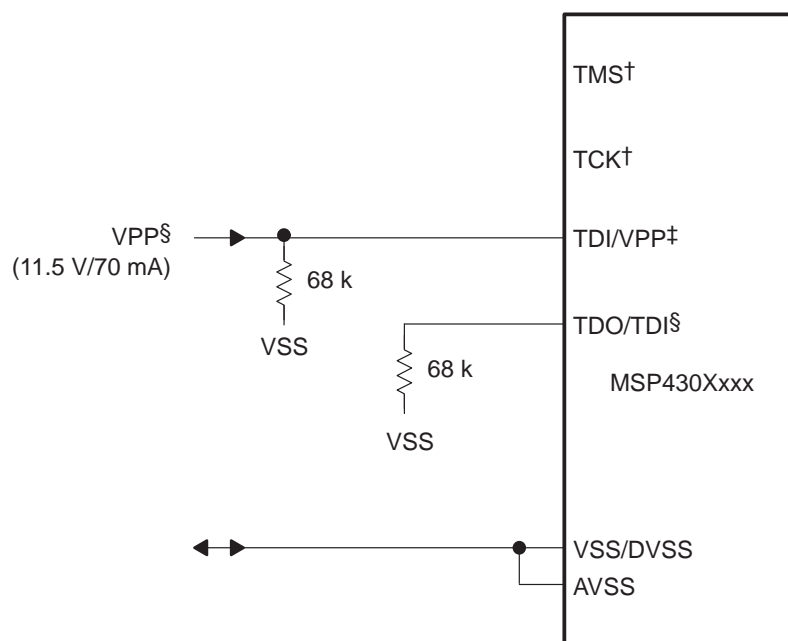
4.4 Programming an EPROM Module With Controller's Software

The procedure for programming an EPROM module is as follows:

- 1) Connect the required supply to the TDI/VPP terminal.
- 2) Run the proper software algorithm.

The software algorithm that controls the EPROM programming cycle cannot run in the same EPROM module to which the data are being written. It is impossible to read instructions from the EPROM and write data to it at the same time. The software needs to run from another memory such as a ROM module, a RAM module, or another EPROM module.

Figure 4–3. EPROM Programming With Controller's Software



† Internally a pullup resistor is connected to TMS and TCK

‡ ROM devices of MSP430 have an internal pullup resistor at pin TDI/VPP.

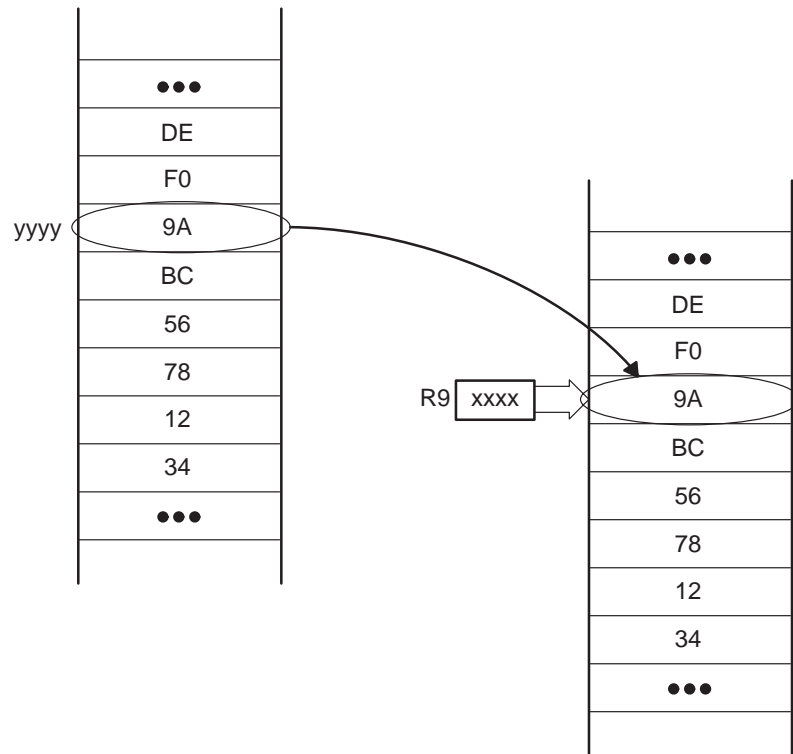
MSP430Pxxx or MSP430Exxx have no internal pullup resistor. They should have an ext. pulldown resistor preventing floating input node.

§ The TDO/TDI pin should have an ext. pulldown resistor preventing floating input node for secondary TDI function.

4.4.1 Example

The software example writes one byte into the EPROM with the fast programming algorithm. The code is written position-independent, and will have been loaded to the RAM before it is used. The programming algorithm runs during the programming sequence in the RAM, thus avoiding conflict when the EPROM is written. The data (byte) that should be written is located in the RAM address BurnByte. The target address of the EPROM module is held in the register pointer defined with the set directive. The timing is adjusted to a cycle time of 1 μs. When another cycle time/processor frequency is selected, the software should be adjusted according to the operating conditions.

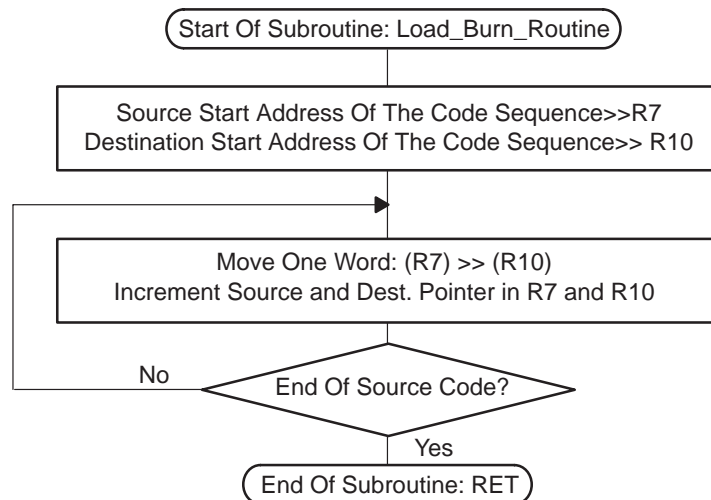
Example 4–3. Programming EPROM Module With Controller's Software



Example: Write data in yyyy into location xxxx
 BumByte = (yyyy) = (9Ah)
 R9 = xxxx

The target EPROM module cannot execute the programming code sequence while the data are being written into it. In the example, a subroutine moves the programming code sequence into another memory, for example, into the RAM.

Example 4–4. Subroutine



4.5 Code

```

;-----
; Definitions used in Subroutine :
; Move programming code sequence into RAM (load_burn_routine)
; Burn a byte into the EPROM area          (Burn_EEPROM)
;-----

EPCTL    .set    054h    ; EPROM Control Register
VPPS     .set    2       ; Program Voltage bit
EXE      .set    1       ; Execution bit
BurnByte .set    0220h   ; address of data to be written
Burn_orig.set    0222h   ; Start address of burn
                        ; program in the RAM

loops    .set    25
r_timer  .set    r8      ; lus = 1 cycle
pointer  .set    r9      ; pointer to the EPROM address
                        ; r9 is saved in the main routine
                        ; before subroutine call is executed

r_count  .set    r10
lp       .set    3       ; dec r_timer : 1 cycle : loop_t100
                        ; jnz          : 2 cycles : loop_t100
ov       .set    2       ; mov #(100-ov)/lp,r_timer : 2 cycles

; Load EPROM programming sequence to another location e.g. RAM, Subroutine

;--- The address of Burn_EEPROM (start of burn EPROM code) and
;--- the address of Burn_end (end of burn EPROM code) and
;--- the start address of the location of the destination
;--- code area (RAM_Burn_EEPROM) are known at assembly/linking time

RAM_Burn_EEPROM .set    Burn_orig
load_burn_routine
    push    r9
    push    r10
    mov     #Burn_EEPROM,R9      ; load pointer source
    mov     #RAM_Burn_EEPROM,R10 ; load pointer dest.
load_burn1
    mov     @R9,0(R10)           ; move a word
    incd    R10                  ; dest. pointer + 2
    incd    R9                   ; source pointer + 2
    cmp     #Burn_end,R9         ; compare to end_of_table
    jne     load_burn1
    pop     r9
    pop     r10
    ret

; Program one byte into EPROM, Subroutine

;--- Burn subroutine: position independent code is needed
;   since in this examples it is shifted to RAM >> only
;--- relative addressing, relative jump instructions, is used!
;--- The timing is correct due to lus per cycle

Burn_EEPROM
    dint                                ; ensure correct burn timing
    mov.b   #VPPS,&EPCTL              ; VPPS on
    push    r_timer                   ; save registers
    push    r_count                   ; programming subroutine
    mov     #loops,r_count            ; 2 cycles = 2 us
Repeat_Burn
    mov.b   &BurnByte,0(pointer)      ; write to data to EPROM

```

```

        bis.b #EXE,&EPCTL          ; 6 cycles = 6 us
                                   ; EXE on
                                   ; 4 cycles = 4 us
                                   ; total cycles VPPon to EXE
                                   ; 12 cycles = 12 us (min.)
    mov     #(100-ov)/lp,r_timer   ;:programming pulse of 100us
wait_100    ;:starts, actual time 102us
    dec     r_timer                ;:
    jnz     wait_100              ;:
    bic.b #EXE,&EPCTL              ;:EXE / prog. pulse off

    mov     #4,r_timer             ;:wait min. 10 us
wait_10     ;:before verifying
    dec     r_timer                ;:programmed EPROM
    jnz     wait_10               ;:location, actual 13+ us

    cmp.b &BurnByte,0(pointer)    ; verify data = burned data
    jne     Burn_EEPROM_bad        ; data ≠ burned data > jump

; Continue here when data correctly burned into EPROM location
    mov.b &BurnByte,0(pointer)    ; write to EPROM again
    bis.b #EXE,&EPCTL              ; EXE on
    add     #(0ffffh-loops+1),r_count
                                   ; Number of loops for
                                   ; successful programming

final_puls
    mov     #(300-ov)/lp,r_timer   ;:programming pulse of
wait_300    ;:3*100us*N starts
    dec     r_timer                ;:
    jnz     wait_300              ;:
    inc     r_count                ;:
    jn      final_puls             ;:
    clr.b &EPCTL                   ;:EXE off / VPPS off
    jmp     Burn_EEPROM_end

Burn_EEPROM_bad
    dec     r_count                ; not ok : decrement
                                   ; loop counter
    jnz     Repeat_Burn            ; loop not ended : do
                                   ; another trial
    inv.b &BurnByte                ; return the inverted data
                                   ; to flag
                                   ; failing the programming
                                   ; attempt the EPROM address
                                   ; is unchanged
                                   ;

Burn_EEPROM_end
    pop     r_count
    pop     r_timer
    eint
    ret

Burn_end

```

