

# ***FlashPro430***

***Fast USB-MSP430 Flash Programmer***

## ***User's Manual***

*Software version 2.2*

***PM010A04 Rev.8***  
*March-20-2005*

# ***Elprotronic Inc.***

91 Alpine Crescent  
**Richmond Hill,**  
**Ontario, L4S-1V9**  
**CANADA**

Web site:      [www.elprotronic.com](http://www.elprotronic.com)  
E-mail:        [info@elprotronic.com](mailto:info@elprotronic.com)  
Fax:            905-780-2414  
Voice:          905-780-5789

***Copyright © 2004 Elprotronic Inc. All rights reserved.***

## Disclaimer:

No part of this document may be reproduced without the prior written consent of Elprotronic Inc. The information in this document is subject to change without notice and does not represent a commitment on any part of Elprotronic Inc. While the information contained herein is assumed to be accurate, Elprotronic Inc. assumes no responsibility for any errors or omissions.

In no event shall Elprotronic Inc, its employees or authors of this document be liable for special, direct, indirect, or consequential damage, losses, costs, charges, claims, demands, claims for lost profits, fees, or expenses of any nature or kind.

The software described in this document is furnished under a licence and may only be used or copied in accordance with the terms of such a licence.

**Disclaimer of warranties:** You agree that Elprotronic Inc. has made no express warranties to You regarding the software, hardware, firmware and related documentation. The software, hardware, firmware and related documentation being provided to You “AS IS” without warranty or support of any kind. Elprotronic Inc. disclaims all warranties with regard to the software, express or implied, including, without limitation, any implied warranties of fitness for a particular purpose, merchantability, merchantable quality or noninfringement of third-party rights.

**Limit of liability:** In no event will Elprotronic Inc. be liable to you for any loss of use, interruption of business, or any direct, indirect, special incidental or consequential damages of any kind (including lost profits) regardless of the form of action whether in contract, tort (including negligence), strict product liability or otherwise, even if Elprotronic Inc. has been advised of the possibility of such damages.

## **END USER LICENSE AGREEMENT**

PLEASE READ THIS DOCUMENT CAREFULLY BEFORE USING THE SOFTWARE AND ASSOCIATED THE HARDWARE. ELPROTRONIC INC. AND/OR ITS SUBSIDIARIES (“ELPROTRONIC”) IS WILLING TO LICENSE THE SOFTWARE AND ASSOCIATED THE HARDWARE TO YOU AS AN INDIVIDUAL, THE COMPANY, OR LEGAL ENTITY THAT WILL BE USING THE SOFTWARE AND/OR ASSOCIATED THE HARDWARE (REFERENCED BELOW AS “YOU” OR “YOUR”) ONLY ON THE CONDITION THAT YOU AGREE TO ALL TERMS OF THIS LICENSE AGREEMENT. THIS IS A LEGAL AND ENFORCABLE CONTRACT BETWEEN YOU AND ELPROTRONIC. BY OPENING THIS PACKAGE, BREAKING THE SEAL, CLICKING “I AGREE” BUTTON OR OTHERWISE INDICATING ASSENT ELECTRONICALLY, OR LOADING THE SOFTWARE, OR USING THE HARDWARE, YOU AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THESE TERMS AND CONDITIONS, CLICK ON THE “I DO NOT AGREE” BUTTON OR OTHERWISE INDICATE REFUSAL, MAKE NO FURTHER USE OF THE FULL PRODUCT AND RETURN IT WITH THE PROOF OF PURCHASE TO THE DEALER FROM WHOM IT WAS ACQUIRED WITHIN THIRTY (30) DAYS OF PURCHASE AND YOUR MONEY WILL BE REFUNDED.

### **1. License.**

The software, hardware, firmware and related documentation (collectively the “Product”) is the property of Elprotronic or its licensors and is protected by copyright law. While Elprotronic continues to own the Product, You will have certain rights to use the Product after Your acceptance of this license. This license governs any releases, revisions, or enhancements to the Product that Elprotronic may furnish to You. Your rights and obligations with respect to the use of this Product are as follows:

#### **YOU MAY:**

- A. use this Product on a single computer;
- B. make one copy of the software for archival purposes, or copy the software onto the hard disk of Your computer and retain the original for archival purposes;
- C. use the software on the network, provided that You have a licensed copy of the software for each computer that can access the software over that network; and
- D. use the hardware with any software obtained from Elprotronic provided that You have a licensed copy of the software with which the hardware will be used.

**YOU MAY NOT:**

- A. copy the printed documentation that accompanies this Product
- B. sublicense, rent or lease any portion of the Product; reverse engineer, decompile, disassemble, modify, translate, make any attempt to discover the Source Code of the Product; or create derivative works from the Product;
- C. use a previous version or copy the Product after You have received a disk replacement or upgraded version. Upon upgrading the software, all copies of the prior version must be destroyed;
- D. redistribute, in whole or in part, any part of the software component of this Product;
- E. use the hardware with any software that is not obtained from Elprotronic; nor
- F. use the product in any manner not authorized by this license.

**2. Copyright**

All rights, title, and copyrights in and to the Product and any copies of the Product are owned by Elprotronic. The Product is protected by copyright laws and international treaty provisions. Therefore, you must treat the Product like any other copyrighted material.

**3. Limitation of liability.**

In no event shall Elprotronic be liable to you for any loss of use, interruption of business, or any direct, indirect, special, incidental or consequential damages of any kind (including lost profits) regardless of the form of action whether in contract, tort (including negligence), strict product liability or otherwise, even if Elprotronic has been advised of the possibility of such damages.

**4. DISCLAIMER OF WARRANTIES.**

You agree that Elprotronic has made no express warranties to You regarding the software, hardware, firmware and related documentation. The software, hardware, firmware and related documentation being provided to You “AS IS” without warranty or support of any kind. Elprotronic disclaims all warranties with regard to the software and hardware, express or implied, including, without limitation, any implied warranties of fitness for a particular purpose, merchantability, merchantable quality or noninfringement of third-party rights.



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

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

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

**Warning:** *Changes or modifications not expressly approved by Elprotronic Inc. could void the user's authority to operate the equipment.*



*This Class B digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.*

*Cet appareil numérique de la classe B respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.*

# Table of Contents

|   |    |
|---|----|
| 1. Introduction .....                                 | 8  |
| 2. Features .....                                     | 10 |
| 3. Getting Started .....                              | 12 |
| 3.1 Software Installation .....                       | 12 |
| 3.1.1 Driver Installation .....                       | 12 |
| 3.2 Hardware Setup .....                              | 13 |
| 3.3 Starting up "FlashPro430" Flash Programmer .....  | 14 |
| 3.4 X-Pro430 Selector .....                           | 14 |
| 4. Programming Dialogue Screen .....                  | 16 |
| 4.1 Interface Type .....                              | 17 |
| 4.2 Microcontroller Type .....                        | 18 |
| 4.4 Blow Security Fuse and Open Password File .....   | 25 |
| 4.5 Power Device from Adapter .....                   | 29 |
| 4.6 Device Action box .....                           | 30 |
| 4.6.1 Auto Program button .....                       | 32 |
| 4.6.2 Verify Security Fuse / Password button .....    | 32 |
| 4.6.3 Erase Flash button .....                        | 33 |
| 4.6.4 Blank Check button .....                        | 33 |
| 4.6.5 Write Flash button .....                        | 33 |
| 4.6.6 Verify Flash button .....                       | 34 |
| 4.6.7 Read/ Copy Flash button .....                   | 35 |
| 4.7 Next button .....                                 | 37 |
| 5. Data viewers .....                                 | 38 |
| 6. Memory Option Dialogue Screen .....                | 41 |
| 6.1 Memory Erase/Write/Verify Group .....             | 41 |
| 6.2 Read Group .....                                  | 43 |
| 6.3 Verification Group .....                          | 44 |
| 7. Adapter Options .....                              | 45 |
| 7.1 JTAG / BSL Communication Speed Dialogue Box ..... | 45 |
| 7.1.1 BSL Communication Speed .....                   | 45 |

|         |  |    |
|---------|--|----|
| 7.1.2   | JTAG Communication Speed .....                       | 46 |
| 7.2     | Reset Dialogue Box .....                             | 46 |
| 7.2.1   | Reset pulse duration .....                           | 46 |
| 7.2.2   | Final Target Device action .....                     | 47 |
| 7.3     | Options Dialogue Box .....                           | 47 |
| 8.      | <i>Serialization</i> .....                           | 49 |
| 8.1     | Introduction .....                                   | 49 |
| 8.2     | Serialization Dialogue Screen .....                  | 50 |
| 8.2.1   | Serial number File .....                             | 51 |
| 8.2.2   | Serial number formats .....                          | 51 |
| 8.2.2.1 | HEX ( MSW first ) and HEX ( LSW first ) format ..... | 52 |
| 8.2.2.2 | BCD format .....                                     | 55 |
| 8.2.2.3 | ASCII format .....                                   | 58 |
| 8.2.3   | Model, Group, Revision .....                         | 61 |
| 8.2.4   | Device Serialization box .....                       | 61 |
| 8.2.5   | Bar Code Scanner setup .....                         | 62 |
| 8.3     | Serialization Report Dialogue Screen .....           | 62 |
| 9.      | <i>Load/Save Setup</i> .....                         | 64 |
| 10.     | <i>Adapter description</i> .....                     | 65 |
| 11.     | <i>Target connection</i> .....                       | 67 |
|         | Appendix A - <i>specification</i> .....              | 72 |
|         | Appendix B - <i>performance</i> .....                | 73 |
|         | Appendix C - <i>software options</i> .....           | 75 |

# 1. Introduction

---

**FlashPro430** (model USB-MSP430-FPA) programmer is dedicated to program the Texas Instruments MSP430Fxx family microcontrollers. Using **FlashPro430** programmer the target device can be programmed via the JTAG Interface, or via the BSL (Boot Strap Loader) Interface. MSP430Fxx microcontrollers can be program via the JTAG Interface as long, as the Security Fuse is not blown. When the Security Fuse is blown then the MSP430Fxx microcontrollers still can be accessible, but only via the BSL Interface.

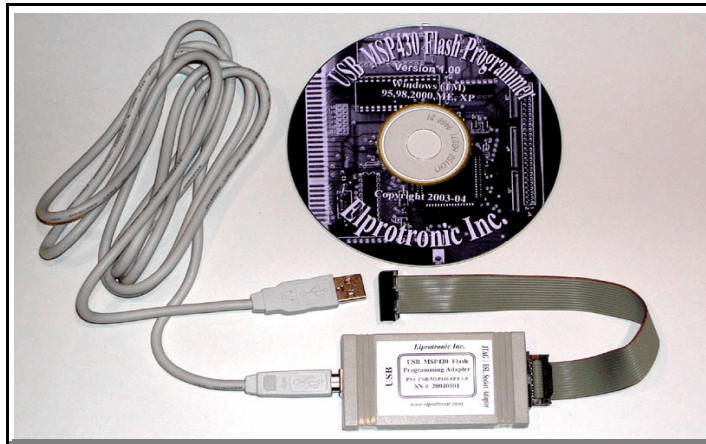


Figure 1-1

Each programmer package (Figure 1-1) consist of a microcontroller based adapter, Windows™ based software and cable to connect the adapter with the computer's USB port. The internal adapter software allows to communicate with the programmed device with the high speed. The programming speed using the JTAG interface is around **29 kbytes/s**. The Fast BSL Interface allows to communicate with the programmed device at speed of up to 350 kbits/s,

which is **over 35 times faster** than with standard BSL (communication speed 9.6 kb/s). Due to this high speed communication, programming time is very short and programmer can be used to program flash devices in the production process. For example, a microcontroller with **60 kB Flash**, such as MSP430F149, can be programmed in **3.5 seconds** via the JTAG Interface, or in **6 seconds** via the BSL Interface. This time includes initialization, erasing memory, blank checking, programming and fast verification.

To facilitate high speed communication via the BSL Interface, a new Fast Bootstrap Loader (Fast BSL) is temporarily downloaded to the RAM of each programmed device. Due to the small size of Fast BSL, almost all TI microcontrollers from MSP430Fxx family can utilize high speed communication. Microcontrollers with small RAM (128 bytes - only 3 family members) can be programmed with the standard communication speed (9.6 kb/s - communication speed is selected automatically). Programming time for these types of microcontrollers is not very long, since the size of the programming FLASH is very small (up to 2 kB).



To simplify production process the programming software package can assign serial number, model type, and revision. Each serial number is unique for each programmed device and is assigned automatically. Several serial number formats are available.

There are a number of erase/write options also available. This allows to erase/write all flash memory, or just the specified fragment of memory. This feature is very useful when only part of programmed data/code should be replaced. For example this feature can be used to download the serial number, calibration data or personality data without losing existing program code.

Currently (March-2005) three types of the USB Flash Programming Adapters commonly named ***X-Pro430 FPA*** are available (from Elprotronic):

1. ***FlashPro430 FPA*** ( model PN: USB-MSP430-FPA-1.x )
2. ***ChainPro430 FPA*** ( model PN: USB-MSP430-FPA-2.x )
3. ***GangPro430 FPA*** ( model PN: USB-MSP430-FPA-3.x )

All three ***X-Pro430 FPA*** models can be used with the ***FlashPro430*** programming software that allows to program only one target device at the time with some different performance - see table below.

| FPA ->                      | <b><i>FlashPro430 FPA</i></b> | <b><i>ChainPro430 FPA</i></b> | <b><i>GangPro430 FPA</i></b> |
|-----------------------------|-------------------------------|-------------------------------|------------------------------|
| 1. Vcc from FPA             | 3.2 V                         | 2.2 to 3.6V step 0.2V         | 2.2 to 3.6V step 0.2V        |
| 2. External Vcc             | 1.8 to 3.6V                   | 1.8 to 3.6V                   | 1.8 to 3.6V                  |
| 3. JTAG interface           | YES                           | YES                           | YES                          |
| speed 4 Mb/s                | YES                           | YES                           | NO                           |
| speed 1 Mb/s                | YES                           | YES                           | YES                          |
| speed 400 kb/s              | YES                           | YES                           | YES                          |
| 4. BSL Interface            | YES                           | YES                           | NO                           |
| 5. MSP430.dll for debugging | YES                           | YES                           | YES                          |

All other performance not listed in the table above are the same when used with the ***FlashPro430*** programming software. Note, that the ***ChainPro430 FPA*** can be used with the ***ChainPro430*** software and the ***GangPro430 FPA*** can be used with the ***GangPro430*** software to access few target device from one FPA. See the ***ChainPro430*** and ***GangPro430*** specifications for details.

## 2. Features

---

**FlashPro430** programmer is dedicated to program the Texas Instruments MSP430Fxx microcontroller family via the JTAG port or via the BSL port.

When the JTAG Interface is used, then the **FlashPro430** programmer is using the standard JTAG communication port, available on all MSP430 microcontroller. Detailed information describing features of the JTAG communication port can be found in the Texas Instruments ( TI ) documentation - SLAA149 -“Programming a Flash-Based MSP430 Using the JTAG Interface”.

To facilitate high speed communication via the JTAG port, an application software for the programming adapter has been optimized for the maximum speed. Also a few new procedures have been implemented, decreasing the flash programming time. Key features of the **FlashPro430** programmer using JTAG communication port are as follows:

- support all MSP430 flash-based devices,
- programming speed approximately 29 kB/s,
- mass erase, erase main memory only or segment erase,
- fast verify and blank check,
- check sum calculation,
- supports programming of the JTAG access security fuse (permanently disables device memory access via JTAG),

When the BSL Interface is used, then the **FlashPro430** programmer is using the standard BSL communication port, available on all MSP430 microcontrollers. Detailed information describing features of the standard BSL can be found in the Texas Instruments ( TI ) documentation - SLAA089A -“Features of the MSP430 Bootstrap Loader”.

To facilitate high speed communication a new Fast Bootstrap Loader (Fast BSL) (proprietary of Elprotronic Inc.) is temporary downloaded to the RAM of each programmed device. The Fast BSL provides the following functions:

- mass erase, erase main memory only or segment erase,
- blank memory check,
- check sum calculation,
- write and verify word (2 bytes) into flash memory,
- read word (2 bytes) from memory.

## Major features of the *FlashPro430* programmer are:

- \* Support all MSP430Fxx microcontrollers from TI.
- \* Our programmers are professionally made and are **recommended by Texas Instruments** as the Third Party Tools source.
- \* Our programmers are currently **the fastest programmers** on the market.
- \* Programmer has a **unique feature** - two interfaces in one package allowing to program the target device via **JTAG** or **BSL** Interface.
- \* To speed up production process optimized programming algorithms are used to significantly reduce programming time. Our proprietary Fast BSL algorithms allow to communicate with the target devices over **35 times faster** than standard TI BSL (9.6kb/s) reducing programming time from few minutes to few seconds.
- \* Programming speed via JTAG Interface is around **29 kbytes/s**.
- \* Blow the JTAG security fuse capability.
- \* Full memory or sector memory erase capability.
- \* Write Check Sum verification.
- \* No code size limitations.
- \* Target device can be powered from the programming adapter or from external source.
- \* Easy to use Windows<sup>TM</sup> based software.
- \* Programmer accept TI (\*.txt), Motorola (\*.s19) and Intel (\*.hex) data files for programming.
- \* Combine code files capability.
- \* Lock setup capability, useful in production.
- \* Software package can assign and automatically increment **serial number**, model type and revision. Serial Number with or without an automatically inserted current date can be stored in the FLASH memory in HEX, BCD or ASCII format. Log file capability allowing to review information about the flashed target devices.
- \* **DLL** software package can control programmer from other programs.
- \* Programmer has been fully tested to comply with the **FCC** and **CE** requirements.
- \* Our programmers are inexpensive - for users interested in basic features we provide limited, or lite software version.
- \* Using USB-1.1 (12Mbits/s) Port to communicate with the Programming Adapter.
- \* Communication with the target device via JTAG/BSL Interface using TI-standard 14-pins header connector.

## 3. Getting Started

---

The *X-Pro430* programmer package contains:

1. One READ ME FIRST document.
2. One *X-Pro430* Flash Programmer CD ROM ( Software + Manual ).
3. One *FlashPro430*, *ChainPro430* or *GangPro430* (USB MSP430 FPA) Flash Programming Adapter.
4. One 6 feet length USB-A to USB-B cable extender.

### 3.1 Software Installation

The *X-Pro430* USB MSP430 Flash Programming Software runs on PC under Windows™ ME, WinNT, 2000 or XP. Follow instructions below to install the software:

1. Insert *X-Pro430* (the USB MSP430 Flash Programming) Software CD into your CD-ROM drive.
2. The *X-Pro430* Setup wizard appears automatically. Click *Install X-Pro430 Programmer* to begin the installation process.
3. If the Setup wizard does not start automatically, click the Start button and choose the Run dialogue box. Type “D:\SETUP.EXE”, where D represents the drive letter of your CD-ROM drive. Then click the OK button.
4. Once the installation program starts, on-screen instructions will guide you through the remainder of the installation. You **must** accept licence agreement before using software.

#### 3.1.1 Driver Installation

1. Plug in USB-MSP430-FPA to the PC USB Port, using provided cable extender (USB-A to USB-B).  
==== Windows XP ====
2. The “*New hardware has been found*” should be displayed. Follow the wizard instruction to install the drivers. Drivers are located in the CD ROM directory “D:\drivers\W2000,ME,XP”, where “D” represents the drive letter of your CD-ROM drive or in the application software directory  
**C:\Program Files\Elprotronic\XPro USB Drivers\W2000,ME,XP**

==== Windows - 2000, 98-SE ====

2. The “*New hardware has been found*” should be displayed. Follow the wizard instruction to install the drivers.
3. Press ‘**Next**’ when the Device Wizard Driver screen appear.
4. Select the following option on the wizard screen:  
    \* **select for a suitable driver for my device (recommended)**  
and press ‘**Next**’.
5. Select the third option - “**Specify a location**” for a location of the Driver Files.
6. From the browser select the “**D:\drivers\W2000,ME,XP**” for Win-2000 or  
    “**D:\drivers\W98**” for Win-98SE directory, where D:CD-ROM drive location or in the application software directory  
    **C:\Program Files\Elprotronic\XPro USB Drivers\W98** and press ‘**Next**’.
7. Driver installation process will start.

Driver installation procedures should be done twice. Software will install two USB drivers - the Boot driver and the Application driver. Reboot computer on the end.

### **3.2 Hardware Setup**

1. Connect the Fast USB-MSP430 Flash Programming Adapter to the PC USB Port, using provided cable extender (USB-A to USB-B).
2. Plug in socket connector from the Fast USBMSP430 Flash Programming Adapter to the header connector on your device board. Make sure that pin 1 on your device board’s header is connected to pin 1 of the socket connector. Pin 1 is marked as a red cable on the ribbon cable.

### 3.3 Starting up “FlashPro430” Flash Programmer

To start the **FlashPro430** Flash Programmer click on the Elprotronic FlashPro430 icon.



Figure 3.3-1

Once started the software will attempt to access the programming adapter. If no error messages appear then the software has initialized without a problem and you may begin using it. However, if the programming adapter is not detected an error message will appear. To correct the problem, make sure that the connection cable is properly attached and the USB driver is installed.

### 3.4 X-Pro430 Selector

The **X-Pro430** ( *FlashPro430* (Single), *ChainPro430* (Chain) or *GangPro430* (Gang)) **Flash Programming Adapter (FPA)** has Multi-USB feature. Up to 8 Flash Programming Adapters can be connected to one PC. Each adapter can be controlled by one opened software application. Up to eight application software can be opened at the same time. Each application software can have independent setup from the other application software setup (code file, controlled microcontroller type etc.)

When more than one **X-Pro430** adapter is connected to PC then following **X-Pro430** selector dialogue screen will be displayed on the PC screen (see Figure 3.4-1). Using available buttons the one desired Flash Programming Adapter should be selected. Make a sure, that selected **X-Pro430** programming adapter is not used by other opened application. Selected **X-Pro430**'s serial number will be displayed on the left bottom side of the programming

dialogue screen. The **FlashPro430** programming software supports all **X-Pro430** programming adapters. **FlashPro430** programming software can be used to access a single target device, regardless type of the used FPA. Using the **FlashPro430 FPA** (model USB-MSP430-FPA-1.x) or **ChainPro430-FPA** (model USB-MSP430-FPA-2.x) the target device can be connected to FPA via JTAG or BSL interface, while using the **GangPro430 FPA** (model USB-MSP430-FPA-3.x) the target device can be accessible via JTAG interface only.

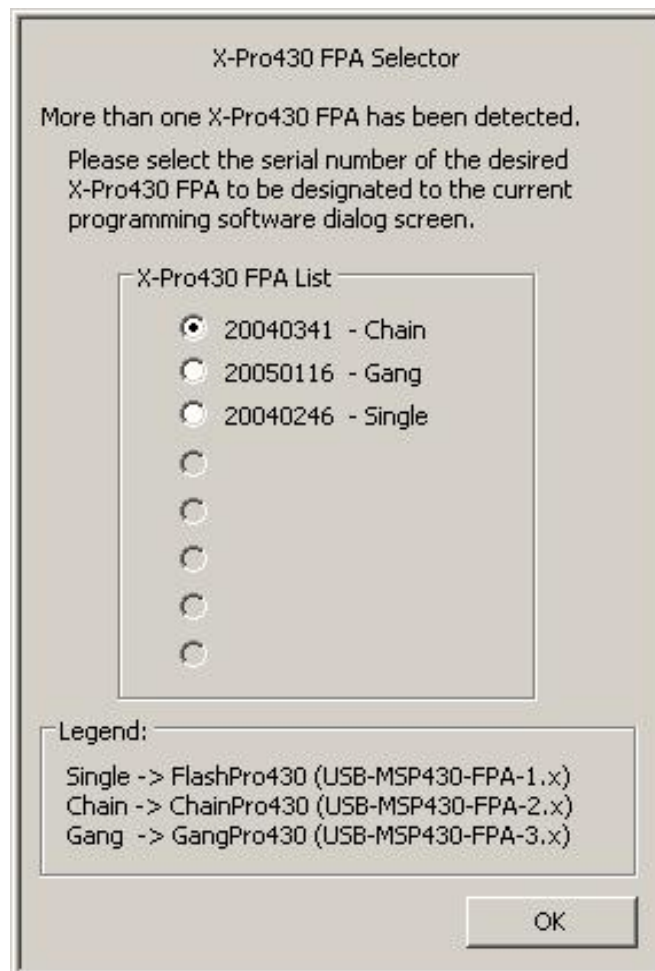


Figure 3.4-1

## 4. Programming Dialogue Screen

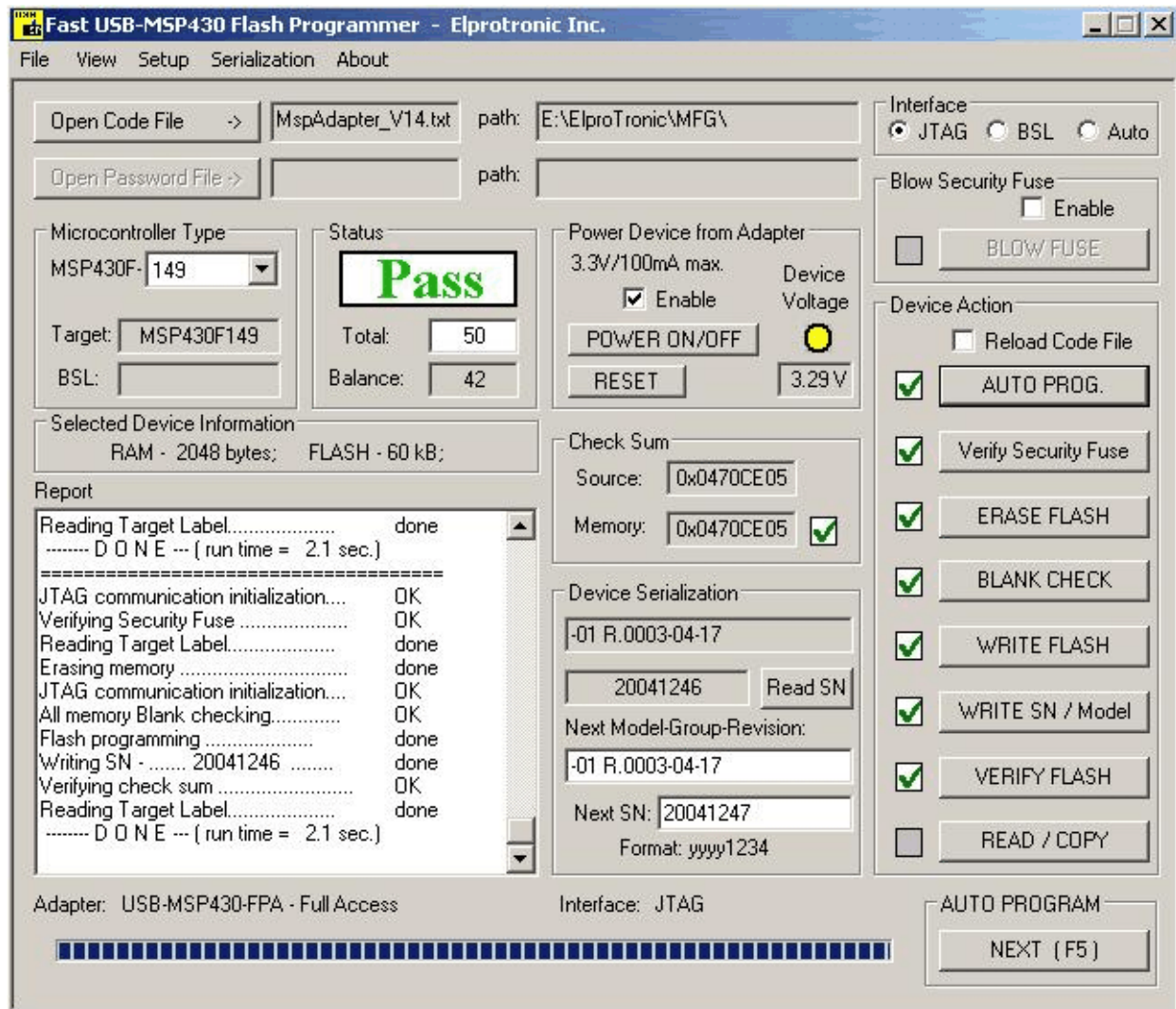









Figure 4-1. Programming dialog box screen.

The programming dialogue box (see Fig. 4-1.) contains a pull down menu, interface selection box, blow fuse box, device action buttons, report (status) window, open file buttons, processor information box, serial number box, power DC status and check sum result boxes.



All device action buttons, power ON/OFF button and the check sum result box have their own status indicators. Each indicator can assume any of the following conditions:

-  - blank - idle status.
-  - yellow - Test in progress. For power on/off - DC voltage is correct.
-  - green - access enabled.
-  - red sign - access denied. For power on/off - DC voltage is too low (below 2.6V)
-  - device action has been finished successfully.
-  - device action has been finished, but result failed.
-  - applies to blank check only - Memory is not clean, but the specified memory segment is.

## 4.1 Interface Type

The communication interface type - **JTAG**, **BSL** or **Auto** can be selected in the Interface group. When the programming adapter has implemented two types of communication interface, then all buttons in the interface group are active, otherwise only one selecting button is active - JTAG or BSL. Proper communication interface should be selected, otherwise communication with the target device can fail.



Figure 4.1

It is recommended to prepare in the target device board one 14-pin JTAG connector with JTAG and BSL connections (see chapter 11 for details). When two communication interfaces are connected between the target device and programming adapter, then selection of any interface type can activate communication between the programming adapter and a target device.

It is recommended to use by default communication through the JTAG Interface, because this communication is around two times faster than the communication through the BSL Interface. Also '**blow the security fuse**' procedure is accessible only from the JTAG communication port. When the security fuse is blown, then JTAG interface is unusable, and only the BSL communication port can be used. When option **Auto** is selected, then communication always starting using the JTAG interface. If **blow of the security fuse** has been detected, then

communication is continuing using the BSL interfaces. Next communication is starting using the JTAG interface as the first again.

## 4.2 Microcontroller Type

The microcontroller type can be selected from the pull down field of the processor type group. The pull down field contains a list of all microcontrollers in MSP430Fxx family currently available. One thing to note, the microcontroller type can be selected automatically if the option '**Any**' is selected.

When communication between microcontroller and programming adapter is initialized, the software will detect the target microcontroller's automatically. The type of detected microcontroller is displayed in the field '**Target:**'. This allows the software to warn you if the connected microcontroller does not match the one specified by the user.

*Note: No warning message will appear when '**Any**' microcontroller type is selected.*

When communication between microcontroller and programming adapter is using BSL Interface, then Bootstrap Loader (BSL) version downloaded from the target device is displayed in the field '**BSL:**'

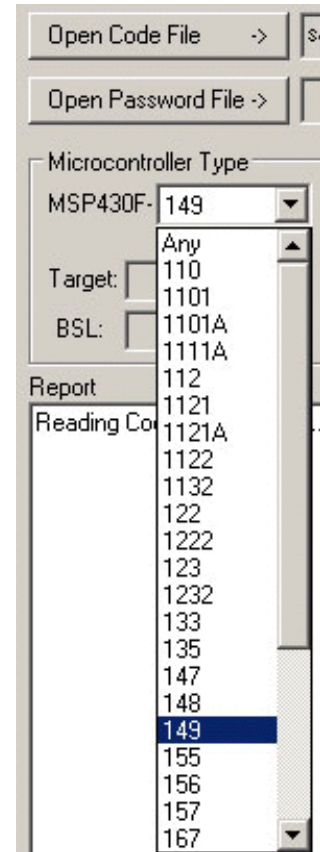


Figure 4.2-1

Texas Instruments has been created number of microcontroller's groups and numbers of the microcontroller's type. Microcontrollers with the same group has the same ID number saved in the ROM at the location 0x0FF0. Microcontrollers with the same group ID has a similar features with a different size of RAM and FLASH.

Contents of the ROM at location 0x0FF0 containing ID number can be read using JTAG or BSL communication. Particular type of the microcontroller in the same group ID can be detected when communication via JTAG is available, but this feature is not available via BSL

interface communication. For programming flash feature knowledge of the type of the microcontrollers is not required as long as size of FLASH is available.

Tables of all currently available group and types of microcontrollers are shown below. Tables contains following information:

- in [F112] - ID (in HEX format) taken from the ROM at location 0x0FF0,
- F11x(1) - information displayed in the microcontroller type window in programmer dialogue box,
- list of available microcontrollers in particular group with RAM and Flash size specification.

[ F112 ] F11x(1)

| Name                | RAM size [bytes] | ROM size [kbytes] |
|---------------------|------------------|-------------------|
| MSP430F110          | 128              | 1 k               |
| MSP430F1101         | 128              | 1 k               |
| MSP430F1101A        | 128              | 1 k               |
| MSP430F1111A        | 128              | 2 k               |
| MSP430F112          | 256              | 4 k               |
| MSP430F1121         | 256              | 4 k               |
| <b>MSP430F1121A</b> | <b>256</b>       | <b>4 k</b>        |

[ 1132 ] F11x2

| Name               | RAM size [bytes] | ROM size [kbytes] |
|--------------------|------------------|-------------------|
| MSP430F1122        | 256              | 4 k               |
| <b>MSP430F1132</b> | <b>256</b>       | <b>8 k</b>        |

[ F123 ] F122..F123

| Name              | RAM size [bytes] | ROM size [kbytes] |
|-------------------|------------------|-------------------|
| MSP430F122        | 256              | 4 k               |
| <b>MSP430F123</b> | <b>256</b>       | <b>8 k</b>        |

[ 1232 ] F12x2

| Name               | RAM size [bytes] | ROM size [kbytes] |
|--------------------|------------------|-------------------|
| MSP430F1222        | 256              | 4 k               |
| <b>MSP430F1232</b> | <b>256</b>       | <b>8 k</b>        |

[ F149 ] F13x..F14x

| Name              | RAM size [bytes] | ROM size [kbytes] |
|-------------------|------------------|-------------------|
| MSP430F133        | 256              | 8 k               |
| MSP430F135        | 512              | 16 k              |
| MSP430F147        | 1024             | 32 k              |
| MSP430F1471       | 1024             | 32 k              |
| MSP430F148        | 2048             | 48 k              |
| MSP430F1481       | 2048             | 48 k              |
| <b>MSP430F149</b> | <b>2048</b>      | <b>60 k</b>       |
| MSP430F1491       | 2048             | 60 k              |

[ F169 ] F15x..F16x

| Name              | RAM size [bytes] | ROM size [kbytes] |
|-------------------|------------------|-------------------|
| MSP430F155        | 512              | 16 k              |
| MSP430F156        | 1024             | 24 k              |
| MSP430F157        | 1024             | 32 k              |
| MSP430F167        | 1024             | 32 k              |
| MSP430F168        | 2048             | 48 k              |
| <b>MSP430F169</b> | <b>2048</b>      | <b>60 k</b>       |

[ F16C ] F16xx

| Name               | RAM size [bytes] | ROM size [kbytes] |
|--------------------|------------------|-------------------|
| MSP430F1610        | 5120             | 32 k              |
| MSP430F1611        | 10240            | 48 k              |
| <b>MSP430F1612</b> | <b>5120</b>      | <b>55 k</b>       |

[ F413 ] F412..F417

| Name              | RAM size [bytes] | ROM size [kbytes] |
|-------------------|------------------|-------------------|
| MSP430F412        | 256              | 4 k               |
| MSP430F413        | 256              | 8 k               |
| MSP430F415        | 512              | 16 k              |
| <b>MSP430F417</b> | <b>1024</b>      | <b>32 k</b>       |

[ F437 ] F435..F437

| Name                     | RAM size [bytes] | ROM size [kbytes] |
|--------------------------|------------------|-------------------|
| MSP430F435-80pins        | 512              | 16 k              |
| MSP430F436-80pins        | 1024             | 24 k              |
| <b>MSP430F437-80pins</b> | <b>1024</b>      | <b>32 k</b>       |

[ F449 ] F43x..F44x

| Name                      | RAM size [bytes] | ROM size [kbytes] |
|---------------------------|------------------|-------------------|
| MSP430F435-100pins        | 512              | 16 k              |
| MSP430F436-100pins        | 1024             | 24 k              |
| <b>MSP430F437-100pins</b> | <b>1024</b>      | <b>32 k</b>       |
| MSP430F447                | 1024             | 32 k              |
| MSP430F448                | 2048             | 48 k              |
| <b>MSP430F449</b>         | <b>1024</b>      | <b>60 k</b>       |

[ F427 ] F423..F427

| Name              | RAM size [bytes] | ROM size [kbytes] |
|-------------------|------------------|-------------------|
| MSP430F423        | 256              | 8 k               |
| MSP430FE423       | 256              | 8 k               |
| MSP430FW423       | 256              | 8 k               |
| MSP430F425        | 512              | 16 k              |
| MSP430FE425       | 512              | 16 k              |
| MSP430FW425       | 512              | 16 k              |
| <b>MSP430F427</b> | <b>1024</b>      | <b>32 k</b>       |
| MSP430FE427       | 1024             | 32 k              |
| MSP430FW427       | 1024             | 32 k              |

[ F439 ] FG43x

| Name               | RAM size [bytes] | ROM size [kbytes] |
|--------------------|------------------|-------------------|
| MSP430FG438        | 1024             | 48 k              |
| <b>MSP430FG439</b> | <b>2048</b>      | <b>60 k</b>       |

| Name               | RAM size [bytes] | ROM size [kbytes] |
|--------------------|------------------|-------------------|
| MSP430F2101        | 128              | 1 k               |
| MSP430F2111        | 128              | 2 k               |
| MSP430F2121        | 256              | 4 k               |
| <b>MSP430F2131</b> | <b>256</b>       | <b>8 k</b>        |

When '**Any**' microcontroller type is selected then only name of the microcontroller's group like **FG43x** is displayed in the **Target microcontroller type**. Because type of microcontroller can not be fully detected (especially via BSL Interface) then the max. FLASH from the particular group is assigned, eg. 60kB for the group **FG43x** (see table above). If correct size of the FLASH is required then the desired microcontroller type should be selected. When communication with the target device is established and when the selected and the target microcontrollers are from the same group, then a size of the target device are taken from the selected microcontroller type data table. In this case the full name of the microcontroller's is displayed in the **Target microcontroller type** field like **MSP430FG438** instead the group name **FG43x** only. Otherwise the microcontroller with the maximum size of FLASH from the detected group is selected (shown in **bold** in the tables above) and group name is displayed like **FG43x**.

### 4.3 Code File Management

**FlashPro430** flash programmer provides three options to manage code files. These options allow the user to open a code file, combine several code files into a single file, and save the programming data into a code file.

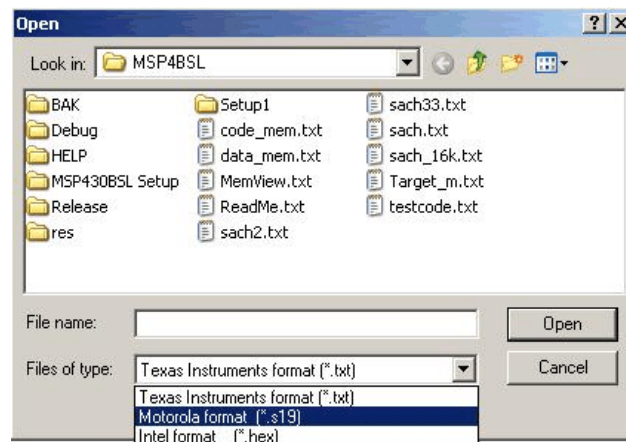


Figure 4.3-1

The **Open Code File** button, or the **Open Code File** from the **FILE** pull down menu, prompts for opening the object file that contains the code data, as shown in Figure 4.3-1. When the file is selected the contents of the object file are downloaded into the PC memory. If the selected microcontroller does not have enough memory to fit the data contained in the code file, the warning message in Figure 4.3-2 will be displayed.



Figure 4.3-2

When code file is open and read successfully the code file name and full path will be displayed on the right side of the **Open Code File** button (see Fig.4-1 Programming dialogue box screen). Contents of the selected file can be viewed by the selecting of '**Code File Data**' from the '**View**' menu (see chapter 5).

The **Combine Code Files** option allows up to 40 code files to be loaded into the PC memory. When this option is selected the programmer will create a new data block, which will contain the combined data of the user selected files. In order to add a code file to the newly created data block, the user needs to press the **ADD Code File** button. The programmer will then prompt the user to specify the code file to be appended to the newly created memory block, using the window in Figure 4.3-1. Every appended file will be verified, so that the total code size does not exceed the target microcontroller's memory space and that there is no overlap with previously selected code segments. After the addition of each file the window in Figure 4.3-3 will be shown. The window shows the status of previous append operations.

The Programmer is also able to append files of any type to the new data block. In order to do this the user must specify the memory location into which the programmer is to load the file and then press the **Add file contents** button. The window in Figure 4.3-1 will appear prompting the user to specify the file to be added. Once the file is added to the new memory block, the programmer will display the memory space occupied by the selected file. An example of this is shown in Figure 4.3-3

for the file number 4.

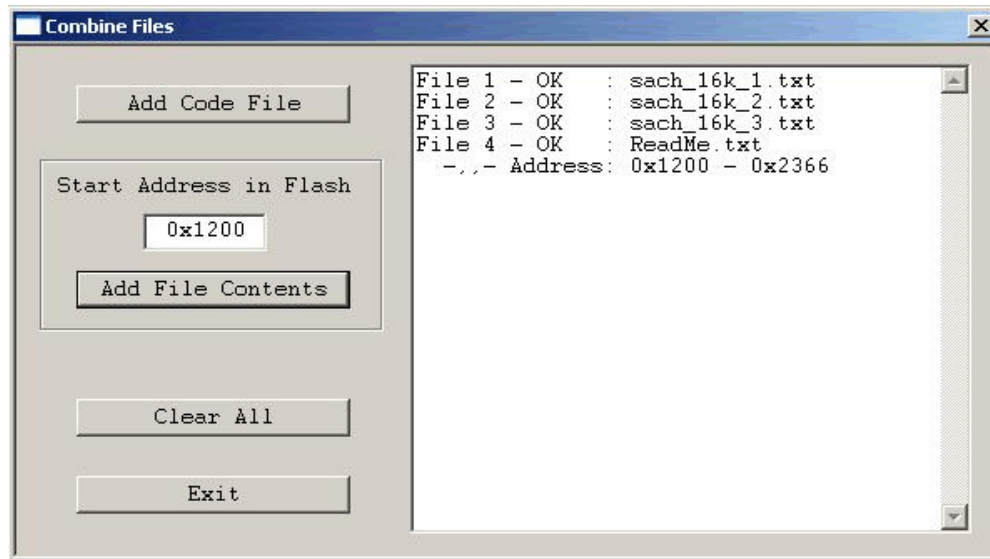


Figure 4.3-3

The **Save Code File** option saves the data currently contained within the PC code data block into a code file. When the user selects this option from the File menu, the window in Figure 4.3-4 will appear, prompting for the name of the file to be created.

All of the aforementioned Code File options work with three most popular code file formats. These formats are the Texas Instruments, the Motorola and the Intel file formats. **FlashPro430** will work with any of these formats and will easily convert one file format to another by using the Open Code File and Save Code File options.

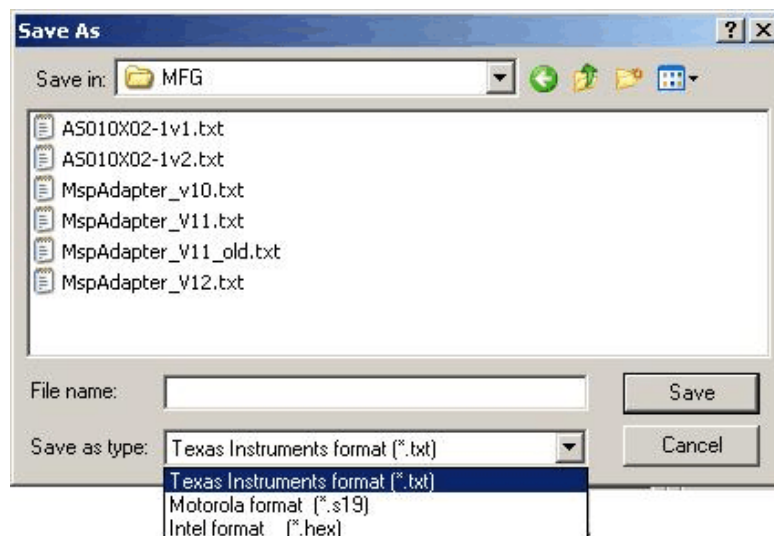


Figure 4.3-4



## 4.4 Blow Security Fuse and Open Password File

The microcontroller's memory is protected against unauthorized access. When the microcontroller is accessed via the JTAG interface, then the Security Fuse if blown is protecting access to the microcontroller. Blowing the Security Fuse is not reversible and when done, then the JTAG interface becomes unusable.

When JTAG interface is selected, then '**Verify Security Fuse**' button allows to verification, if the fuse is blown or not. Fuse is verified also at the beginning of any device action command.

To blow the Security Fuse the check mark '**Enable**' must be selected first (see Figure 4.4-1).



Figure 4.4-1

Because blowing of the Security Fuse is not reversible, the following warning message is displayed when check mark is selected to be enabled.

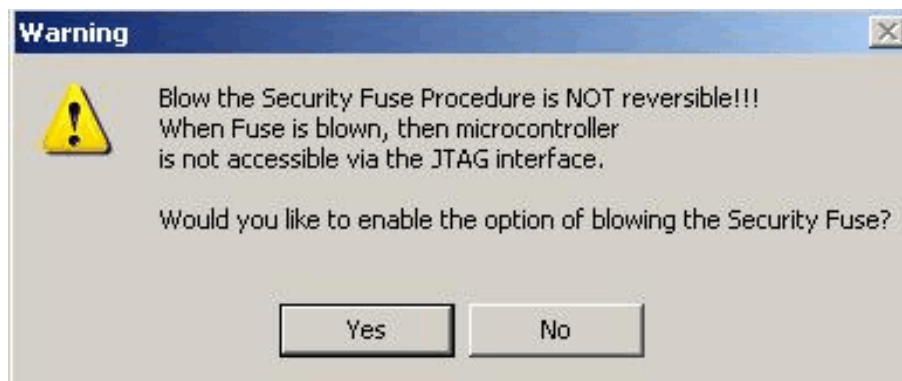


Figure 4.4-2

*Note: If the option of blowing the Security Fuse is enabled, then if AUTO PROGRAM device action is selected, the fuse will be blown without warning.*

When '**BLOW FUSE**' button is pressed, then two following warnings are displayed, before fuse will be blown.

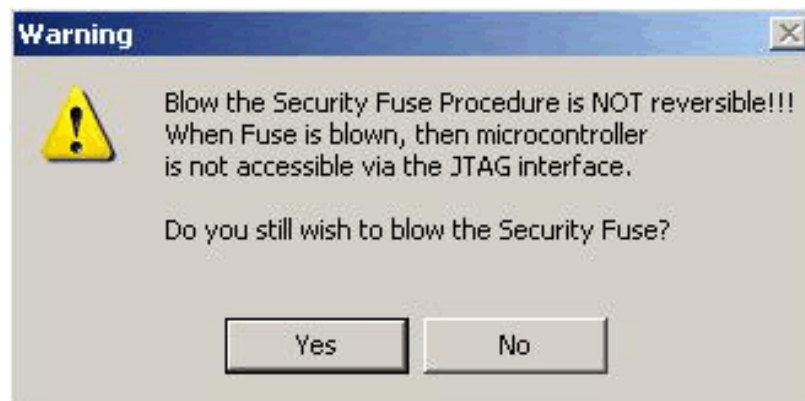


Figure 4.4-3



Figure 4.4-4

When the button 'YES' is pressed twice, the procedure of blowing the security fuse will be initiated. When Security Fuse is blown, the JTAG interface becomes inoperable.

TI prepared the second access to the MSP430 microcontroller via the BSL interface. This access is available any time, even if the Security Fuse is blown. To access the microcontroller via the BSL interface a valid password must be provided. The password consists of 32 bytes of code at location 0xFFE0. If the correct password is not provided, access to the microcontroller via BSL is limited to two commands - erase all memory and verify password. All other commands are unavailable. When password is not known, then the all memory can be erased and a new password (all 0xFF) can unlock access to all BSL commands.

The software supports three types of passwords. The first password, or *default*, is used when the flash memory is blank, ie. all bytes contain value 0xFF. Consequently, the password contains 32 bytes of 0xFF.

The second password is provided by using the ***Open Password File*** button, or ***Open Password File*** from the **FILE** menu. This command opens the code file and retrieves 32 bytes of data that contain the password. Using this password allows the code in the flash memory to be modified.

The third password comes from the code file itself. Whenever ***Open Code File*** command is selected, data from the code file at the memory location 0xFFE0...0xFFFF is extracted for this

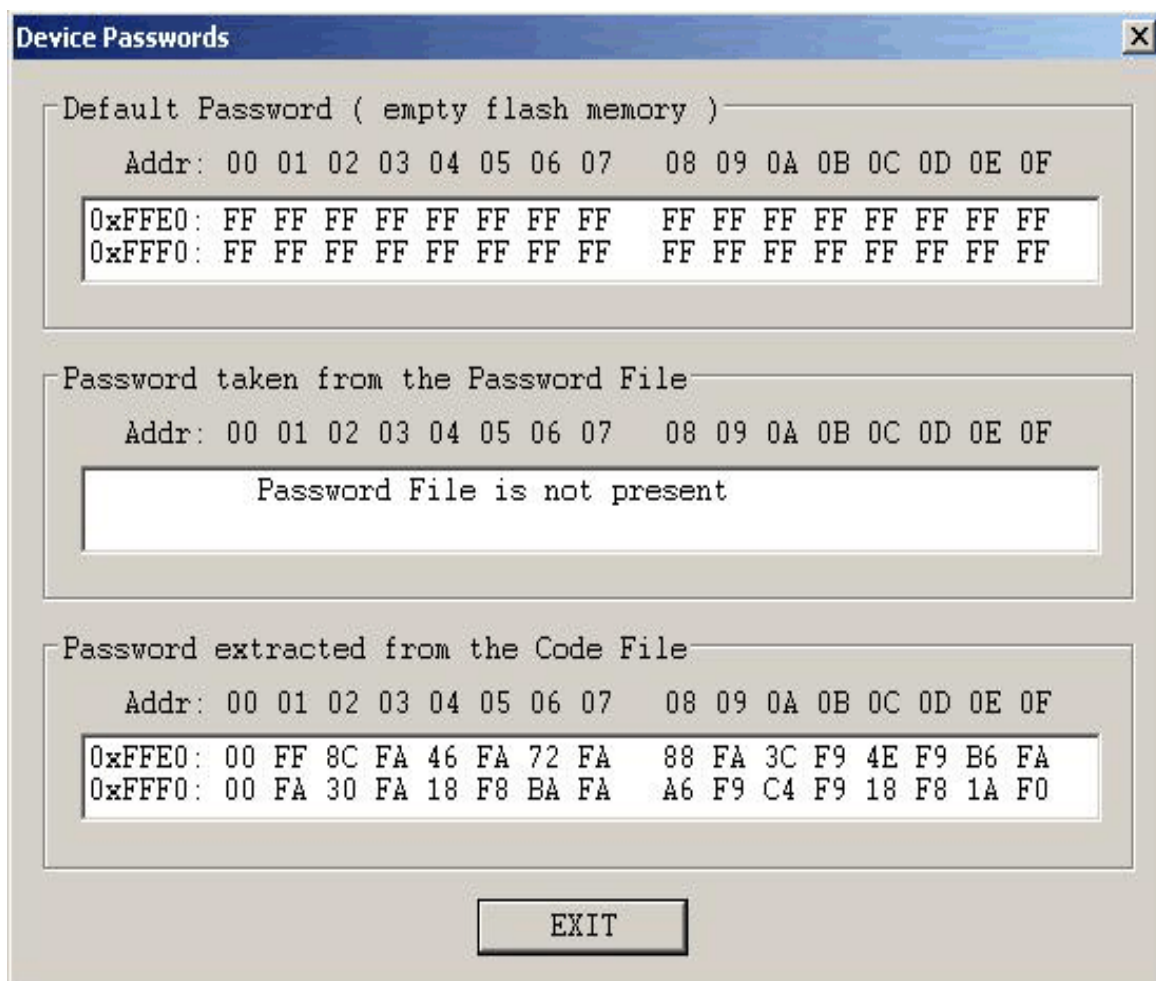


Figure 4.4-5

purpose. The password obtained this way enables verification of memory contents.

The use of a password file is optional. If a password file is not used, then the remaining two passwords will be used. All three passwords can be viewed by selecting **View\Device Passwords** selected from main menu (see Fig.4.4-5).

The three passwords are checked in sequence: default, password file and code file. The first valid password will be accepted. If all passwords fail the following error message will appear:



Figure 4.4-6

## 4.5 Power Device from Adapter

The programming adapter is powered from the USB Port interface. Target device will be powered from the adapter, if check box '**Enable**' in the '**Power Device from Adapter**' group (figure 4.5-1) is selected. When the '**Enable**' checkbox is selected a warning message shown in figure 4.5-2 will be displayed. If you confirm this selection by clicking **YES**, then POWER ON/OFF button is enabled. By clicking POWER ON/OFF button you can turn the power on or off on the target device. Current DC voltage on the target device

is permanently monitored and displayed in the '**Device Voltage**' field in the '**Power Device from Adapter**' group, even if the target device is powered from the external DC sources. If DC voltage is higher then 2.7 V, then yellow box will be displayed, indicating that DC voltage is OK and target device is fully functional under this DC voltage. If DC level is below 2.7V, but higher then 1V, then access denied sign box will be displayed (red sign with white line). If DC level is below 1V, then blank sign box will be displayed.

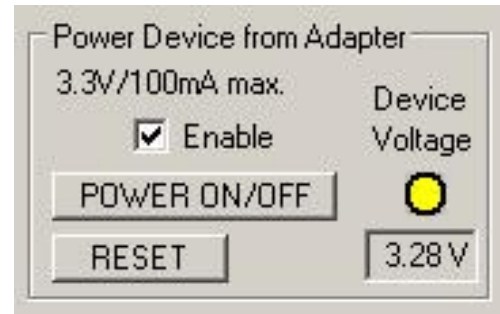


Figure 4.5-1

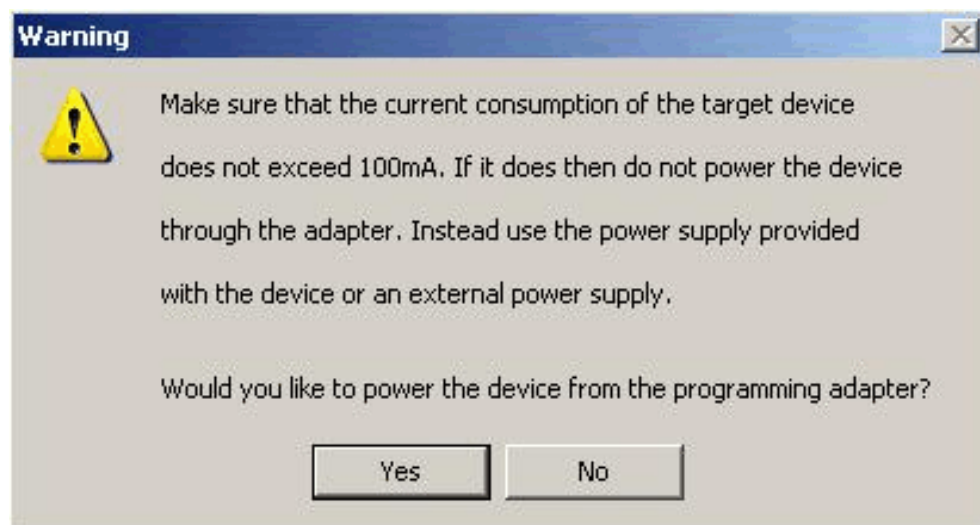


Figure 4.5-2

*Note: Programming adapter prevents any program from running on the target device, when programming adapter is connected to target device via BSL connector.*

When the target device is powered from its own power supply or battery then the check box '**Enable**' should not be selected.

RESET button located under POWER ON/OFF button can generate reset pulse to the target device. Pressing this button the target devices can be reset manually at any time, starting the target's device application program from the beginning.

## 4.6 Device Action box

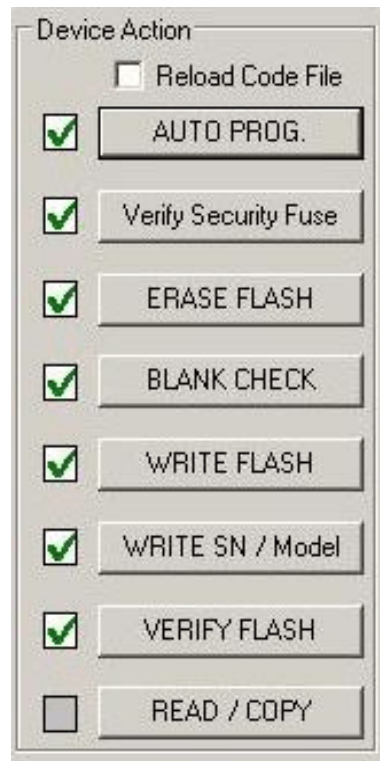


Figure 4.6-1  
JTAG Interface selected

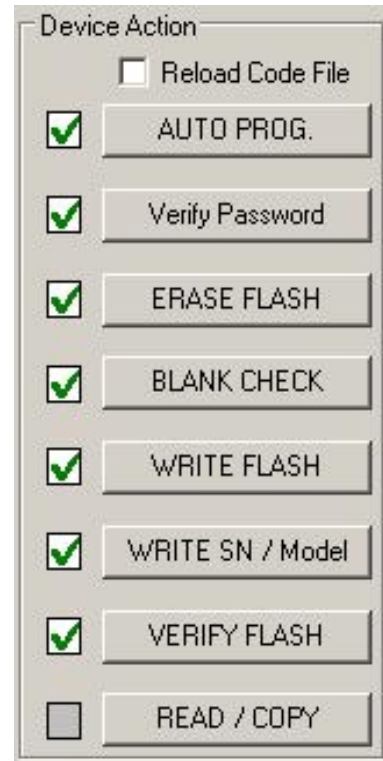


Figure 4.6-2  
BSL Interface selected.

Device Action box contains 8 buttons (see Figure 4.6-1 and 4.6-2) and 8 status boxes. Each button allows a specific action to be executed. Software procedures related to each action allow you to fully execute the desired task, without the need to follow a specific sequence of actions. Every action starts by powering up the target device, if **Power Device from the Adapter**

is enabled. When the DC voltage level becomes higher than 2.7V, the communication with the target device is initiated via JTAG or BSL Interface. When the JTAG Interface is selected then the security fuse is verified, if access to the microcontroller is available. When the BSL Interface is selected, then the password is verified to unlock access to the microcontroller and the Fast BSL is downloaded to the target device. Once the specified action is completed successfully the green check mark will appear. Also, the device will return to the state it was in before the action was executed.

Progress of all actions is displayed in the report window. If the particular action has been finished successfully, then message 'done' or 'OK' will appear on the right side of processed procedure (Fig.4.6.3). If not, a message 'failed' will be displayed and selected action will be terminated. Final status is also displayed in the *Status* window (see Fig.4.6-4) as Active (blue), Pass (green), or Fail (red). On the bottom of the programmer dialogue screen the progress bar is displayed and the total run time is shown in the report window. Run time does not include the time when user interaction is required.

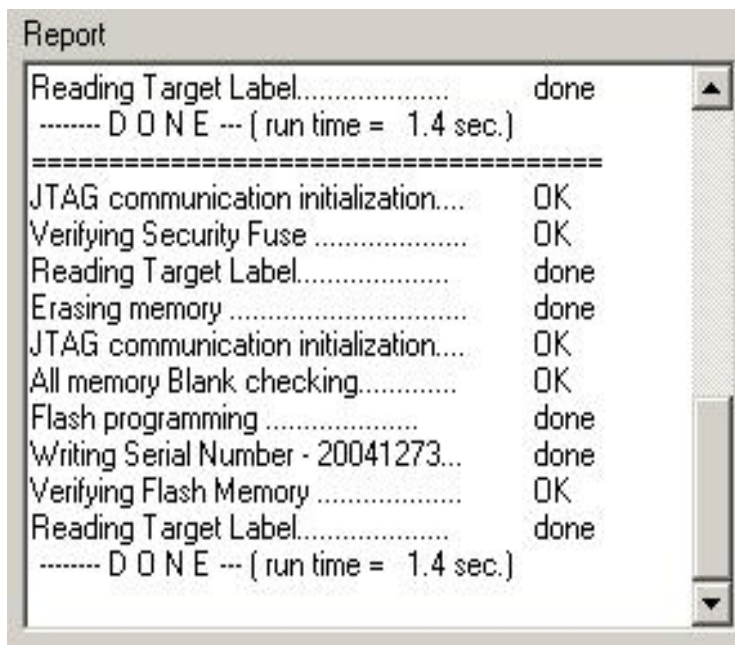


Figure 4.6-3

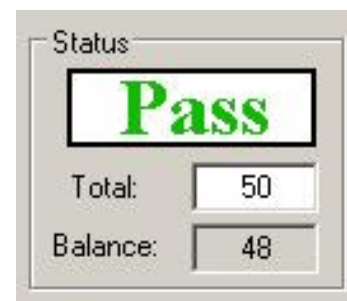


Figure 4.6-4



### 4.6.1 Auto Program button

Auto Program button is the most frequently used button when programming microcontrollers in the production process. Auto Program button activates all required procedures to fully program and verify the flash memory contents. Typically, when flash memory needs to be erased, *Auto Program* executes the following procedures:

- reload code file when “**Reload Code File**” is selected  
(useful for debugging when the code file is frequently modified)
- initialization
- read labelling information (Serial Number, Model, Group, Revision) (optional)
- erase flash memory,
- confirm if memory has been erase,
- flash programming and verification,
- labelling information generation,
- flash memory check sum verification,
- retrieve labelling information,
- blowing the security fuse (if enabled).

In the report window you can see a typical report message during the Auto Program procedure (see Fig. 4.6-3 ).

**Status** window (see fig. 4.6-4) has a counter that is useful in production process. The total number of programmed microcontrollers can be entered in the **Total** edit line. The **Balance** line shows the number of microcontrollers that have not been programmed yet. The Balance counter is initialized to the value entered in the **Total** edit line and is decremented every time *Auto Program* is completed successfully.

**Note:** *Balance counter works only with Auto Program procedure.*

### 4.6.2 Verify Security Fuse / Password button

This button allows the security fuse or the password to be verified. This is useful, if you try to find the correct password from a few available password files, or to check if the security fuse is blown. This procedure is used for test purposes only.



### 4.6.3 Erase Flash button

This button enables the flash memory segments, or mass (all) memory to be erased. If any option other than '**Erase All Memory**' is selected in the Memory Options Setup (see chapter 6.1 **Memory Erase/Write/Verify Group** for details), then the following question message box will be displayed:

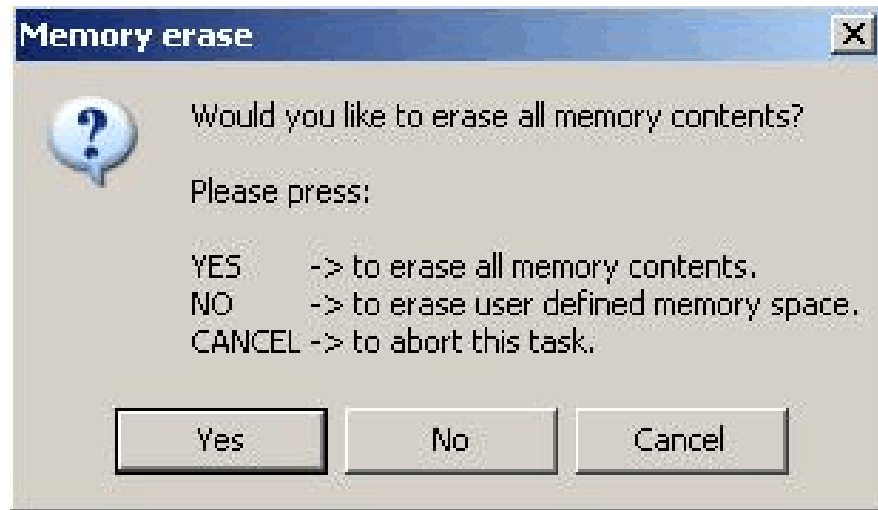





Figure 4.6.3-1

### 4.6.4 Blank Check button

When **Blank Check** button is clicked, the program checks if flash memory of the target microcontroller is blank (all bytes contain the value 0xFF). This test checks if either all memory is clean, or just the specified memory segment. The first test checks all memory contents. If it fails, then just the specified memory segment is checked (see setup in **Memory Erase/Write Group**). The following conditions can appear at the completion of this operation:

-  - all memory is blank
-  - all memory is not blank, but selected part of it is.
-  - memory is not blank.

### 4.6.5 Write Flash button

When write flash button is clicked, then contents from the code file will be written to the

flash memory. If the BSL interface is selected and communication speed is 9.6 kb/s then no additional action will be taken. Otherwise all bytes are verified during the writing process. If verification fails, then writing process is terminated.

*Note: See chapter 5.1 **Memory Erase/Write Group** for details on how to specify memory segment for writing.*

#### 4.6.6 Verify Flash button

The Verify Flash function compares the contents of the flash memory with data from the code file. Verify flash function initiated this way will always use the standard memory verification method, even if the fast verification method is selected from the memory write verification options (see chapter 5. **Memory Option Dialogue Screen**).

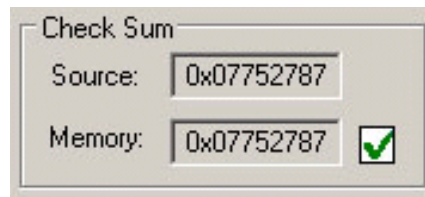


Figure 4.6.6-1

If the JTAG Interface is selected, or BSL Interface with communication speed of 75 or 300 kb/s is selected, then the check sum is verified first. Check sum calculated from the code file data is displayed in the **Source** line of the **Check Sum** group (see Fig.4.5.6-1), and check sum calculated from the target microcontroller flash memory data is displayed in the **Memory** line of this group. If the JTAG Interface is selected, then verification is performed also using a pseudo signature analysis (PSA) algorithm. When check sums (and PSA) match, then OK sign is displayed on the right side of memory check sum result, and the second phase of verification begins. In the second phase all contents of the memory are verified.

*Note: During the verification process either all memory or just the selected part of the memory is verified, depending on settings specified in the Memory Erase/Write Address Range in the Memory Options setup. See chapter 5.1 **Memory Erase/Write Group** for details.*

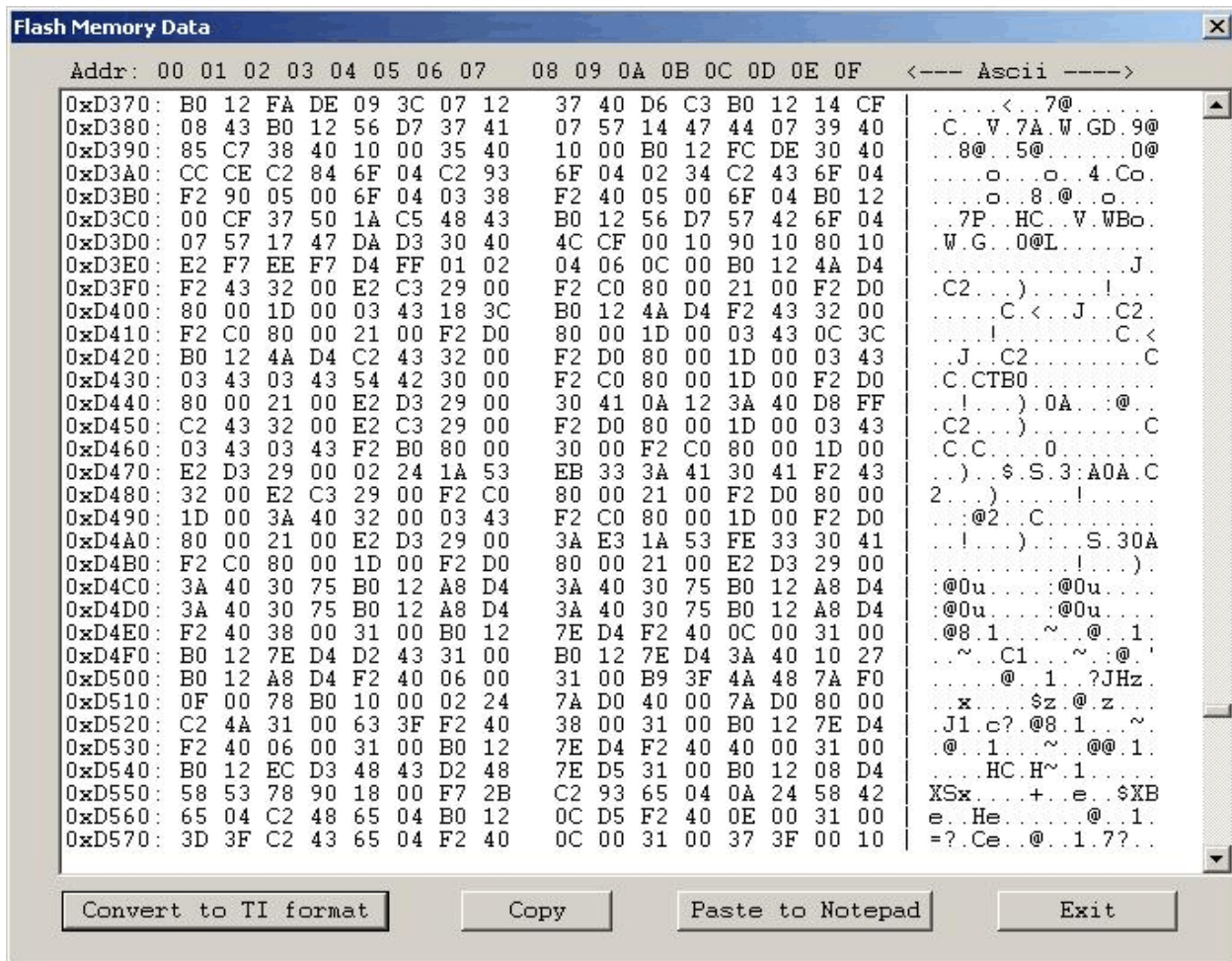


Figure 4.6.7-1

#### 4.6.7 Read/ Copy Flash button

When 'Read/Copy' button is clicked, then data can be read from the target microcontroller and displayed in the Flash Memory Data window (see Fig.4.6.7-1). This window can also be selected from 'Flash Memory Data' from the 'View' menu. Flash memory data viewer, shown in figure 4.6.7-1, displays the code address on the left side, data in hex format in the central column, the same data in Ascii format in the right column. The contents of the code viewer can be converted to Texas Instruments \*.txt file format by clicking on the 'Convert to TI format' button. Data will be viewed in the Notepad Editor.

Read address range can be specified in the Memory Option screen. See chapter 5.2 Read group for details.

When the '**Copy**' button is clicked, then the contents of the read target device memory will be saved in the specified by user file name and opened as a current Code File. Also programmer setup will be modified for the copy procedure. Especially the serialization will be disabled and the '**All Memory**' option will be selected in the '**Write/Erase/Verify Address Range**'. Following message will be displayed.

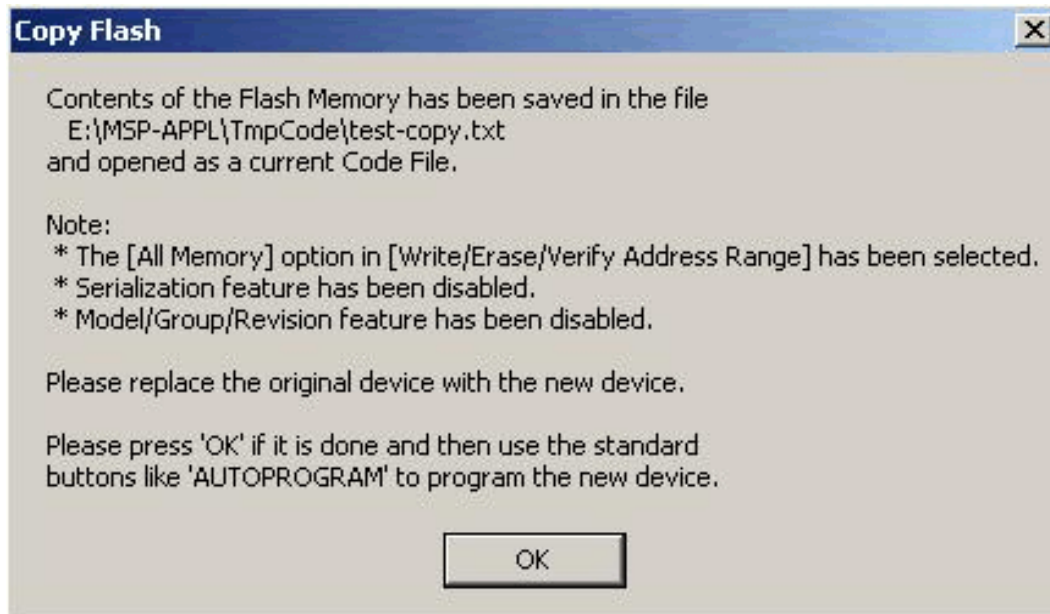


Figure 4.6.7-2

When the button '**OK**' is pressed then programmer is ready to program the destination microcontrollers.

## 4.7 *Next button*

The '*Next*' button is the dynamically programmable device action button, which is very useful in production process. After opening the program, '*NEXT*' button is disabled (see Fig.4.7-1). When any button from the *Device Action* group is pressed, then button '*NEXT*' takes the name and feature of that button. For example, if *Auto Program* button has been used, then it's name will be displayed on top of the '*NEXT*' button (see Fig.4.7-2). From now the button '*NEXT*' will perform the same function as the *Auto Program* button. The '*NEXT*' button has a shortcut to function key *F5*. Button '*NEXT*' will retain its functionality until some other device key is clicked. For example, if key '*READ FLASH*' is clicked, then from this moment button '*NEXT*' will take a name and feature of the '*READ FLASH*' button (see Fig.4.7-3). The read flash procedure will be called, if button '*NEXT*' or function key *F5* is pressed.

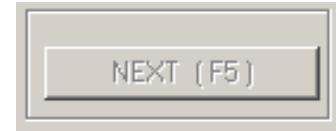


Figure 4.7-1

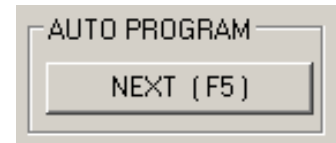


Figure 4.7-2

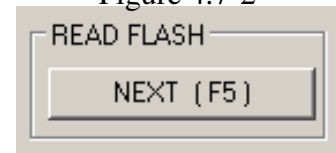


Figure 4.7-3



## 5. Data viewers

Contents data from the Code file and from the Flash memory can be viewed in data viewers.

Also code data and flash memory data can be compared and differences between them can be displayed.

Contents of the selected file can be viewed by selecting of the '**Code File Data**' from the '**View**' menu. Code data viewer, shown in figure 5-1, displays the code address on the left side, data in hex format in the central column, the same data in Ascii format in the right column. Data in hex format is displayed from 00 to FF when contents of data exist in the code file, otherwise it

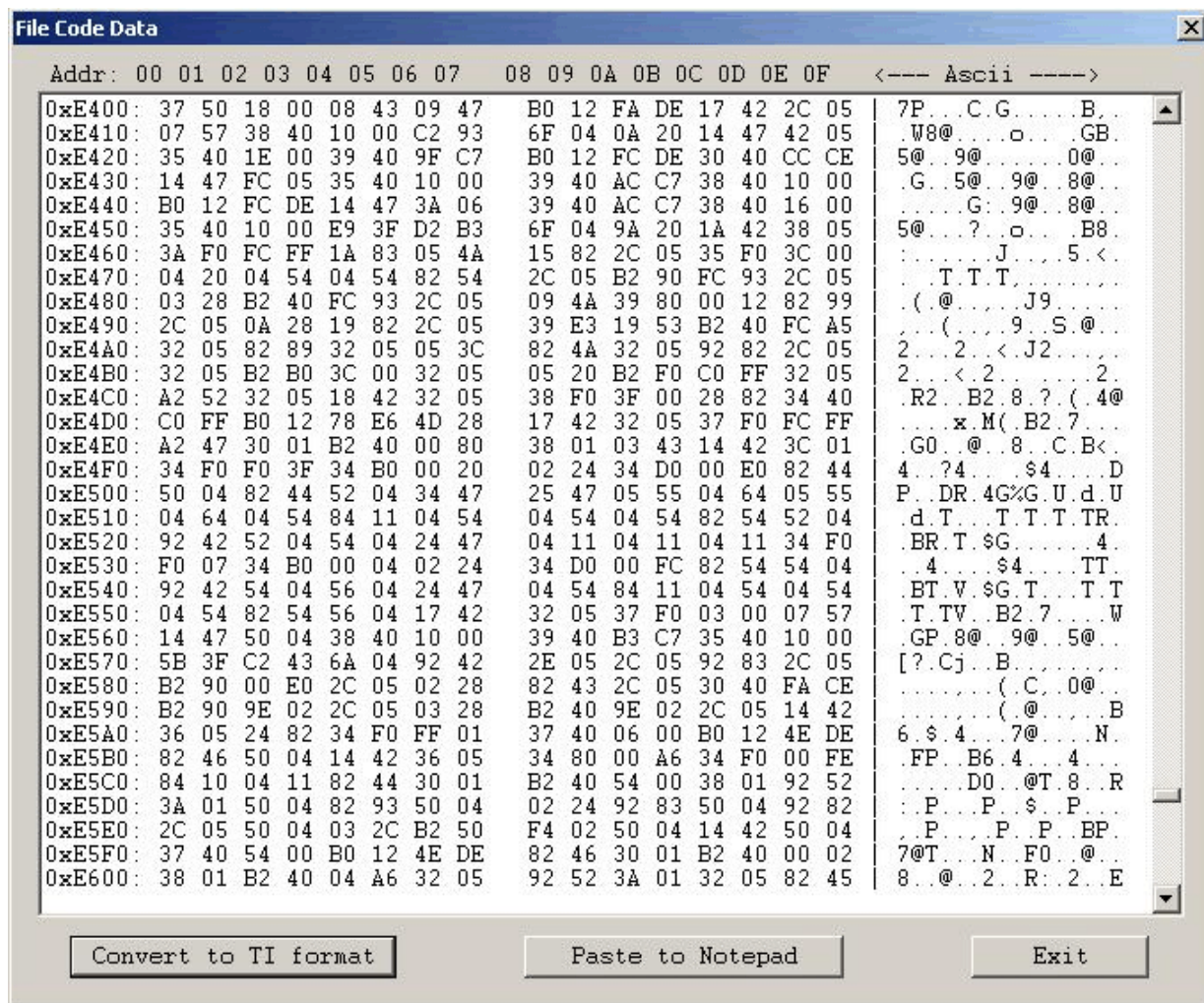


Figure 5-1

is displayed as double dots '..'(if data does not exist in the code file) . When code size exceeds Flash memory space of the selected microcontroller, then warning message

```
'::= Data out of the Flash Memory Space of the selected MSP430. =='
```

is displayed first.

The contents of the code viewer can be converted to Texas Instruments \*.txt file format by clicking on the **'Convert to TI format'** button. Data will be viewed in the Notepad Editor.

Contents of the Flash Memory data can be viewed by selecting of the **'Flash Memory Data'** from the **'View'** menu. Flash Memory data viewer displays the memory address, data in hex and Ascii format in the same way as the code data viewer (Figure 5-1 and 4.6.7-1). To be able to see Flash Memory contents, **'Read Flash'** option must be selected first.

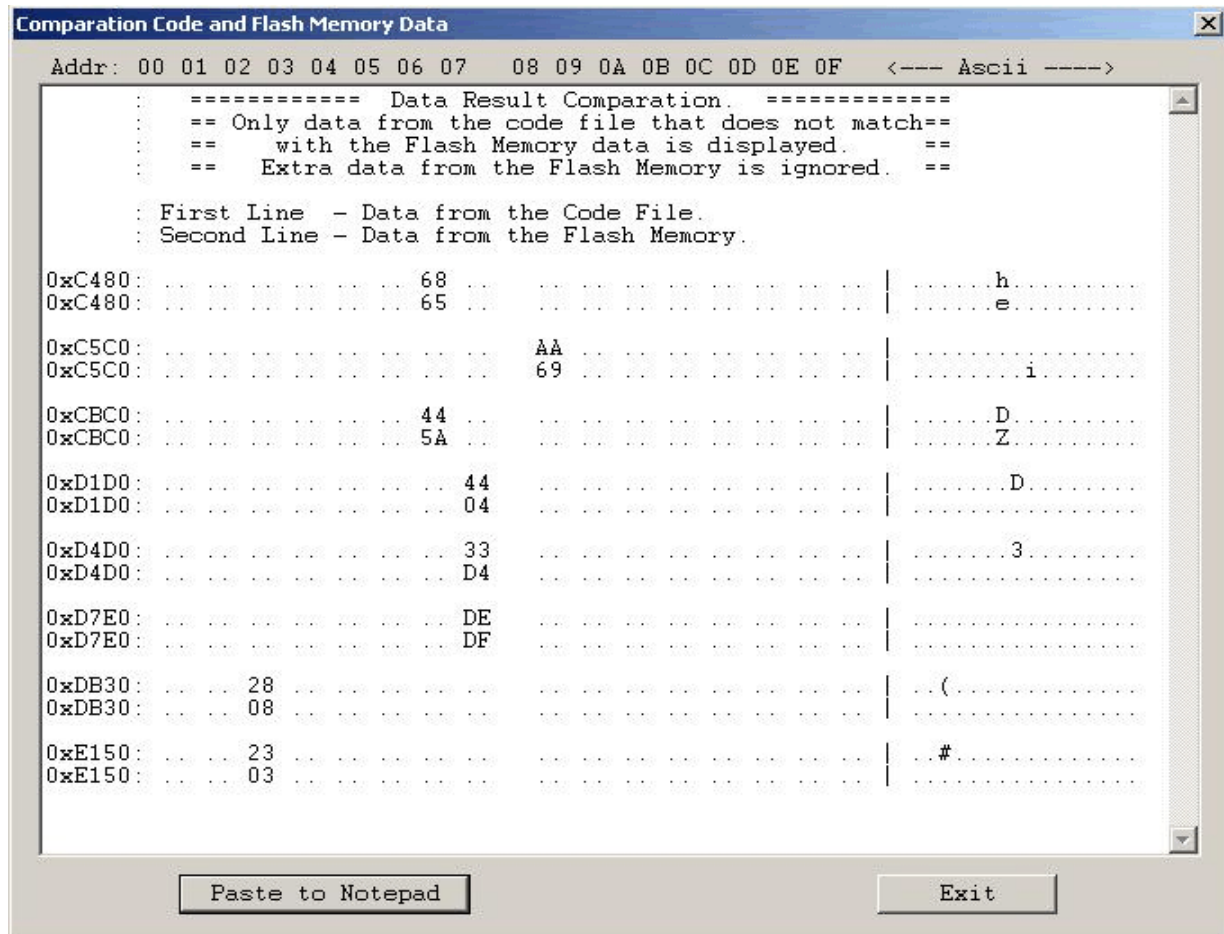


Figure 5-2

Contents of the Code File data and Flash Memory Data can be compared and differences displayed in a the viewer by selecting '**Compare Code & Flash Data**' from the '**View**' menu. Only data that are not the same in the code file data and the Flash memory will be displayed. In the first line code file data will be displayed, and in the second line - Flash memory data (Figure 5-2).

*Note: Only data at the addresses specified in the code file can be displayed. Any data not specified in code file will not be displayed, even if the Flash Memory data contains any not empty (FF) data.*



## 6. Memory Option Dialogue Screen

The Memory Options Dialogue Screen (Fig.6-1) has three settings groups and one information group. Two of the settings groups allow the flash memory addresses range for erase, write and read operation to be specified. The third settings group, write verification, allows the user to select the verification method for *Auto Program* procedure. The information group contains the start and stop address of the user specified main memory segment that can be erased, written and verified independently.

The dialog box is titled "Memory Options" and contains the following sections:

- Memory Erase/Write/Verify Address Range:**
  - ☐ Update Only
  - ☒ All Memory [ including protected INFO segments in F2xx microcontrollers ]
  - ☐ Main Memory Only
  - ☐ Used by Code File
  - ☐ User Defined
- Information Memory Segments:**

| (F1xx,F4xx)                            | [F2xx]      |
|--|-------------|
| <input type="checkbox"/> D (-----)     | [0x1000-3F] |
| <input type="checkbox"/> C (-----)     | [0x1040-7F] |
| <input type="checkbox"/> B (0x1000-7F) | [0x1080-BF] |
| <input type="checkbox"/> A (0x1080-FF) | [0x10C0-FF] |
- Main Memory:**
  - ☐ Enable
  - Start Address: 0x1100
  - Stop Address: 0xFFFF
- Read Address Range:**
  - ☒ All Memory
  - ☐ Main Memory Only
  - ☐ Info Memory Only
  - ☐ User Defined
- Information Memory Segments:**

| (F1xx,F4xx)                            | [F2xx]      |
|--|-------------|
| <input type="checkbox"/> D (-----)     | [0x1000-3F] |
| <input type="checkbox"/> C (-----)     | [0x1040-7F] |
| <input type="checkbox"/> B (0x1000-7F) | [0x1080-BF] |
| <input type="checkbox"/> A (0x1080-FF) | [0x10C0-FF] |
- Main Memory:**
  - ☐ Enable
  - Start Address: 0x1100
  - Stop Address: 0xFFFF
- Write Verification:**
  - ☒ Fast (Write,Verify + Check Sum) [ Recommended ]
  - ☐ Standard (Write,Verify + Check Sum + Read,Verify)
  - ☐ None
- About Microcontroller:**

|                  |                         |        |
|------------------|-------------------------|--------|
| Selected         | Main Memory Start Addr: | 0x4000 |
| Microcontroller: | Main Memory Stop Addr:  | 0xFFFF |
| MSP430F148       | Flash Segment Size:     | 0x0200 |

Buttons: OK, Cancel

Figure 6-1

### 6.1 Memory Erase/Write/Verify Group

The Memory Erase/Write/Verify Address Range group block (see Fig.6-1) specifies common addresses range for erase, write and verify operations. Memory setup has five available options:

1. **Update only:**

When this option is selected the *Auto Program* procedure will not erase memory contents. Instead Contents of the code data taken from the Code File will be downloaded to the flash memory. This option is useful when a relatively small amount of data, such as calibration data, needs to be added to the flash memory. Flash memory space defined by Code File

should be blank. Code file should contain ONLY data, which will be downloaded to flash memory. For example, if code file contains only data as shown in figure 6.1-1 (in Texas Instruments format) then 8 bytes of data will be written starting at location 0x1008 and 6 bytes of data starting at location 0x2200. Before writing operation, all data in the flash memory at the specified location should be blank (contain value 0xFF). The software will verify automatically if this part of memory is blank and will only proceed to program the device if verification is successful.

|   |
|---|
| @1008<br>25 CA 80 40 39 E3 F8 02<br>@2200<br>48 35 59 72 AC B8<br>9 |
|---|

Figure 6.1-1

*Note: Addresses in the Code File should contain only EVEN addresses. Number of bytes in all data blocks **must** be even. The software uses word (two bytes) operation for writing and reading data. In case that the code file contains an odd number of bytes to write the data segment will be appended by a single byte containing the value 0xFF. This value will not overwrite the current memory contents, but verification process will return an error if the target device does not contain the value 0xFF at that location.*

2. **All Memory**

This is the most frequently used option during flash memory programming process. All memory is erased before programming. All contents from the code file are downloaded to the target microcontroller's flash memory.

3. **Main memory only**

This option allows to erase and program the main memory only. Flash information memory (segments A and B) will not be modified. Contents of the information memory from the code file will be ignored, if code file contains such data.

4. ***Used by code file:***

This option allows main memory segments or/and information memory segments, used by data specified in code file, to be erased. Flash memory segments, which do not contain any data to be written to the memory from the code file, will not be erased. This option is useful, if some data, like calibration data, should be replaced in memory. If code file contains some new calibration data, such as described in figure 6.1-1, then the ENTIRE information memory segment at addresses 0x1000 to 0x107F and main memory segment at addresses 0x2200 to 0x23FF will be erased and new data at locations 0x1008 and 0x2200 will be written.

5. ***User Defined:***

This option is functionally similar to options described before, but addresses range of the erased/write/verify main memory and sectors of the information memory can be defined by the user. When the ***User Defined*** option is selected, then on the right side of the ***Memory Erase/Write/Verify Group*** two check boxes and two addresses edit lines will be enabled. The check boxes allow the user to select the information memory sectors A, or/and B to be used (erased, write, verified). Edit lines in the ***Main Memory*** group allow the user to specify the main memory address range (start and stop addresses). Start address should specify the first byte in the segment, and the stop address should specify the last byte in the segment. Since the main memory segment size is 0x200, then the start address should be a multiple of 0x200, eg. 0x2200. The stop address should specify the last byte of the segment to be written. Therefore, it should be greater than the start address and point to a byte that immediately precedes a memory segment boundary, eg. 0x23FF or 0x55FF.

## 6.2 ***Read Group***

The ***Read Address Range*** group block (see Fig.6-1) specifies the address range used in reading process. Memory read setup has four available options:

1. ***All Memory***
2. ***Main memory only***
3. ***Info memory only***
4. ***User Defined***

The meaning of each option is the same as for the erase/write/verify procedure. The ***Info Memory***

*only* option works the same way as *Main memory only* option described above, except that only information memory is modified.

### **6.3 Verification Group**

Verification group setup allows the user to select one of the three write verification methods:

1. ***Fast Verification,***
2. ***Standard Verification,***
3. ***None.***

#### ***Fast Verification:***

Fast verification method can only be used if the JTAG Interface or the Fast BSL is used (communication speed of 350 kb/s). If fast verification is selected and BSL is used (communication speed of 9.6 kb/s), then standard verification procedure is used. During the fast verification, each byte is verified after being written and at the end of the process the check sum is read from the flash memory and compared to calculated check sum taken from the code file. If JTAG Interface is used then verification is performed also using a pseudo signature analysis (PSA) algorithm.

*Note: Fast verification is permanently enabled and can not be switched off, if the JTAG Interface or Fast BSL is used.*

#### ***Standard verification:***

Standard verification is performed after memory write process is completed. Contents of the flash memory are read and compared with the contents of the code file. If both data are the same, then verification process is finished successfully. Typically, the standard verification procedure requires the same amount of time as read/write procedure. Total programming time with standard verification is around two times longer than read/write procedure time.

*Note: If BSL Interface is selected and communication speed is set to 9.6kb/s then standard verification method is the only method available to verify contents of written memory. Otherwise, fast verification is used first and if fast verification is successful, then standard verification procedure is initiated.*

## 7. Adapter Options

---

### 7.1 JTAG / BSL Communication Speed Dialogue Box

The JTAG/BSL Communication Speed Dialogue screen enables the user to select the BSL communication speed between programming adapter and target microcontroller, and select the reset pulse duration coming from the programming adapter to target microcontroller.

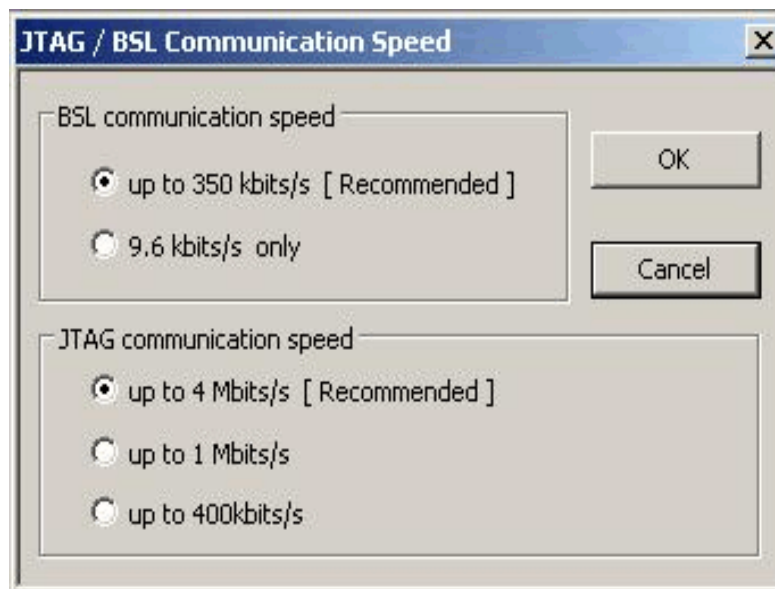


Figure 7-1

#### 7.1.1 BSL Communication Speed

After resetting the target microcontroller the standard Bootstrap Loader, installed in microcontroller's ROM memory, is used for communication between target microcontroller and programming adapter. If internal RAM memory size is 256 or more bytes then Fast BSL is downloaded and used for future communication. The Fast BSL provides communication speed 350 kb/s.

If microcontroller's RAM memory size is below 256 bytes and Fast BSL cannot be downloaded, then standard communication speed (9.6kb/s) will be used. This applies only to three microcontroller types the MSP430Fxx family.

### 7.1.2 JTAG Communication Speed

Default JTAG communication speed between programming adapter and target device is 4 Mb/s. In some condition, when the cable between FPA and target device is long or some protection components are installed in the JTAG interface then the fast JTAG communication can not be used. In this case lower speed 1Mb/s or 400kb/s can be used to establish communication between FPA and target device (see Figure 7-1 - JTAG communication speed selector ).

## 7.2 Reset Dialogue Box

The Target's Reset Dialogue screen enables the user to select the Reset pulse duration and reset line state at the end of programming process.

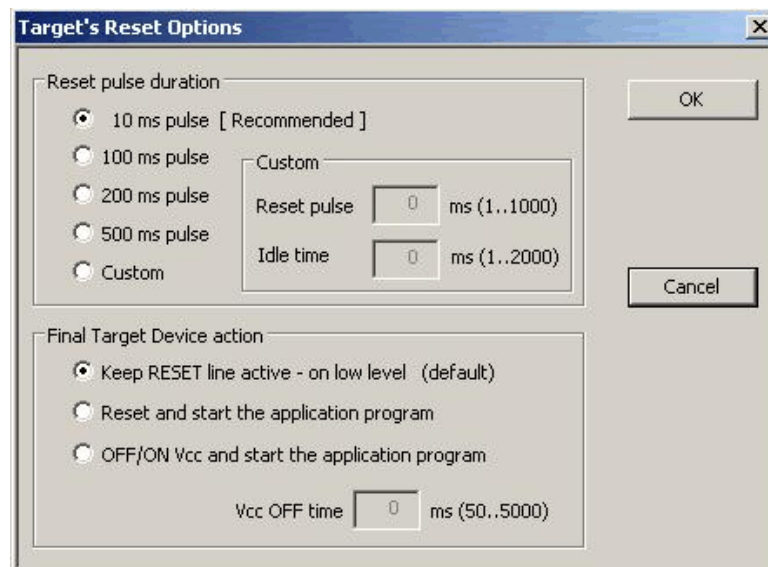


Figure 7.2

### 7.2.1 Reset pulse duration

The reset pulse allows the adapter to initiate communication with a microcontroller using the

JTAG or BSL Interface. In most cases the pulse width of 10ms is sufficient to initiate communication process. However, this may be affected by additional load on the reset line. Therefore, four additional settings, 100, 200, 500 ms and custom , are available. When the RESET IC circuit is used then the custom defined reset pulse duration can be used. Two parameters of the custom define reset pulse are defined - initialization reset pulse time (typically very short - 1 ms) and an idle reset time. Idle reset time must be set at least to duration of the reset time generated by the RESET circuit.

### 7.2.2 Final Target Device action

Every device action, like AUTO Program, Read etc. starts with the activation of the RESET line (active low). When the device programming action begins the RESET line is raised high. When device action is finished, then RESET line is again asserted, protecting the target device from running the application program. This method is commonly used to protect the programming adapter from the DC overload. However, when target device is supplied from its own power supply, or a battery, the overload protection of the programming adapter is no longer necessary.

The target device can be set to run an application immediately after the target device programmed. In order to do this check the ***'Reset and start the application program'*** option in the Reset Options window, shown in Figure 7-2.

## 7.3 Options Dialogue Box

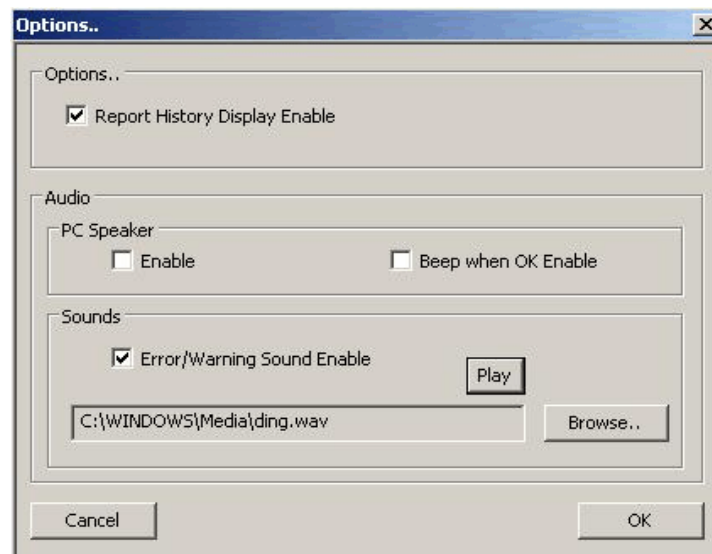


Figure 7.3

The Options Dialogue screen allows to enable or disable the report history in the report window (see figure 4.1). When enabled then the report history is displayed up to 8 kB characters (approximately 20 last communication messages). When disabled, then the only last programming report is displayed.

Programming software can generate audio tones when error programming occurred or tone Ok at the end of programming. Tone can be generated using PC speaker or audio wave generator. Option dialogue box allows to select desired audio option (see Figure 7.3).



## 8. Serialization

---

### 8.1 Introduction

**FlashPro430** programming software has ability to automatically create the target device's serial number and save it in the flash memory. The serial number (SN) that have already been used are stored in the data file. The new SN is created by incrementing a counter that for the SN and the highest SN is stored in a data file. Furthermore, model name, group, revision can be downloaded to target device.

*Note: The SN format and location in the device's flash memory must be specify by the user.*

Serial number is created, when *Auto Program* or *Write SN* button is pressed and the Serial Number feature is enabled. When *Auto Program* function is activated the SN is programmed to the target's device memory at the same time along with code data. If *Auto Program* fails for any reason then new SN is not created.

The software also allows the microcontroller to retain its SN if one has already been assigned to it. Every time a device is programmed and serialization is enabled the contents of the target's memory are scanned for existing serial number. If the serial number is found the message in figure 8.1-1 will appear and allow you to decide if you wish to keep the old serial number. This warning message can be disabled in the serialization dialogue screen.

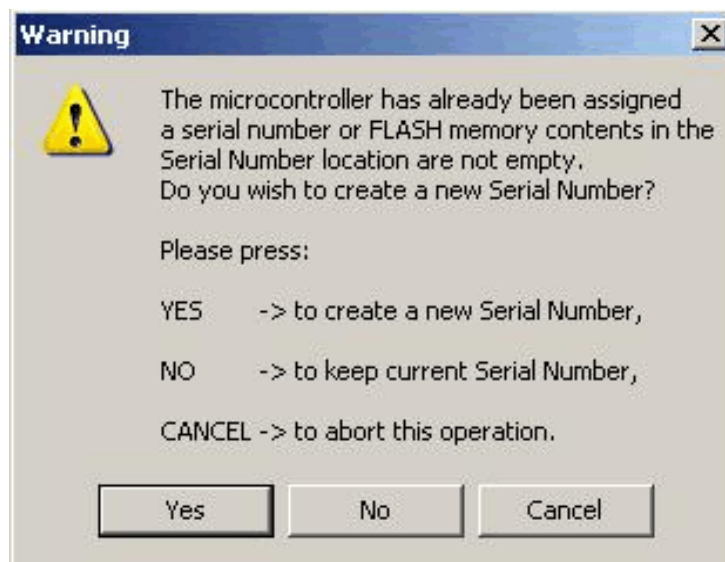


Figure 8.1-1

## 8.2 *Serialization Dialogue Screen*

Serialization dialogue box, shown in figure 8-2, allows configuration for serialization process to be set. Serialization can be enabled, or disabled, by selecting the check mark in the ENABLE Serialization box. When serialization is disabled all edit lines and check boxes are disabled. When serialization is enabled all fields must be set.

**Serialization**

Serialization Setup

☒ ENABLE Serialization

Serial Numbers' File Path and Name:

BarCode Scanner

☐ ENABLE Terminator Character

☐ Start AUTOPROGRAM following BarCode scan

Serial Number Format

Display Format

☒ YYYY-1234(5)

☐ YYMM-1234(5)

☐ YYMMDD-1234

☐ YYDDD-1234(5)

☐ 12345678

☐ 1234(5)

☐ Custom  characters (4..16)

In Memory Format

☐ HEX (MSW First)

☐ HEX (LSW First)

☒ BCD

☐ Ascii

Memory Location

SN Start Address in Memory:  (must be even address)

Used size:  bytes

☒ Warn if Device's Flash Memory is not empty on the SN location

Serial Number (date excluded) starting from:  (decimal)

Model / Group / Revision

☒ ENABLE Text size in Bytes:  (2..32) (even number)

Start Address in Memory:  (must be even address)

Model / Group / Revision text:  16 char.

Figure 8-2

### 8.2.1 Serial number File

The 'Serial Number File Path and Name' specifies the full path and file name, where data base contents will be saved. Serial Number file contains following data, separated by tabulation:

1. Serial Number Format (F0,F1,F2,F3,F4,F5,F6),
2. Serial Number,
3. SN action type (New SN, unmodified SN, overwritten SN, manual SN)
4. Time and date, when SN has been created,
5. Code File Name
6. Model text.

Below is an example of data file, containing data from the three consecutively created serial numbers.

```
F0    200300011  m  ( Sat, Mar 29,2003, 10:09 ) AS010X02-1v2.txt  -01 R.0003-04-17
F0    200300012  .  ( Sat, Mar 29,2003, 10:43 ) AS010X02-1v2.txt  -01 R.0003-04-17
F0    200300013  u  ( Sat, Mar 29,2003, 10:43 ) AS010X02-1v2.txt  -01 R.0003-04-17
```

Serial number can be created as a unique SN per target's device type, or as a unique SN in any devices type. When unique SN per target device type is created, then serial number file name and path should be used for each device type separately. If a unique SN for any devices type is created, then only one serial number file name should be used.

### 8.2.2 Serial number formats

Programming software has seven methods for creating the serial number, referred to as *Display format*, and four methods of storing the SN in the memory, referred to as *In Memory Format* in the serialization dialogue screen. When a serial number is created, current date (if required) is taken from the PC timer. Make a sure, that your computer has correct date and time.

Display Format:

1. YYYY-1234(5)    -( SN Format - **F0**) Serial number has 8 or 9 characters. First four characters contain current year, and remaining 4 or 5 characters contain the serial number, eg. SN 20030123 or 200300123 has a

- number 0123 (or 00123) created in the 2003 year.
2. YYMM-1234(5) - ( SN Format - **F1**) Serial number has 8 or 9 characters. First two characters contain last two digits of current year, next two characters contains current month, and remaining 4 or 5 characters contain a number, eg. SN 03030123.
  3. YYMMDD-1234 - ( SN Format - **F5**) Serial number has 10. First six characters contain date ( year, month, day of month) and remaining 4 characters contain a number, eg. 0405120123.
  4. YYDDD-1234(5) - ( SN Format - **F4**) Serial number has 9 or 10. First five characters contain date ( year, day of year from 1 to 366) and remaining 4 or 5 characters contain a number, eg. 041230123.
  5. 123456768 - ( SN Format - **F2**) 8 digits serial number without date stamp.
  6. 1234(5) - ( SN Format - **F3**) 4 or 5 digits serial number without date stamp.
  9. Custom - ( SN Format - **F6**) 4 to 16 Ascii characters or hexadecimal numbers entered manually or from the Bar-Code Reader.

Serials numbers format 1 to 6 can be stored in memory in HEX, BCD or Ascii format. These formats accept only numeric characters from **0** to **9**. All numbers are displayed in the decimal format, regardless of the format HEX, BCD, Ascii used in the target memory.

Custom serial number can be stored in Ascii format only and is accepting any alphanumeric characters. All characters are converted to the lower letters characters. Any white characters like space, tab are ignored and eliminated from the final SN string.

### **8.2.2.1      *HEX ( MSW first ) and HEX ( LSW first ) format***

When hex format is selected, then all SN display formats described above can be stored as a one or two integer (16-bits - 2 bytes) numbers. First four display characters will be saved as one hex integer number and remaining five characters will be saved as a second hex integer number. When format **HEX(MSW first)** is selected then the first hex integer number is saved as a first word and the second number - as a next word in the Flash memory location. When format **HEX(LSW first)** is selected then the first hex integer number is saved as a second word and the second number - as a first word in the Flash memory location.

**Display Format: YYYY-1234(5)** - size in FLASH - 4 bytes

SN 200300123 will be saved as

|               |        |    |        |       |
|---------------|--------|----|--------|-------|
| YYYY - 2003   | (Decy) | -> | 0x07D3 | (hex) |
| 12345 - 00123 |        | -> | 0x007B | (hex) |

In flash memory this number can be seen as

|      |      |    |                       |
|------|------|----|-----------------------|
| 07D3 | 007B | -> | <i>HEX(MSW first)</i> |
| 007B | 07D3 | -> | <i>HEX(LSW first)</i> |

when integer numbers are viewed, or as

|  |    |    |    |                          |
|--|----|----|----|--------------------------|
| <--- Hex format bytes---> (Size - 4 bytes) |    |    |    |                          |
| D3   | 07 | 7B | 00 | -> <i>HEX(MSW first)</i> |
| 7B   | 00 | D3 | 07 | -> <i>HEX(LSW first)</i> |

when bytes are viewed (first byte is the LSW byte from the integer number)

Displayed consecutive serial number (16-bits integer number) can have a value from 0 to  $(2^{16}-1)$  equal 65535 and is displayed as the 5 digits serial number.

**Display Format: YYMM-1234(5)** - size in FLASH - 4 bytes

SN 030300123 will be saved as

|               |        |    |        |       |
|---------------|--------|----|--------|-------|
| YYMM - 0303   | (Decy) | -> | 0x012F | (hex) |
| 12345 - 00123 |        | -> | 0x007B | (hex) |

In flash memory this number can be seen as

|      |      |    |                       |
|------|------|----|-----------------------|
| 012F | 007B | -> | <i>HEX(MSW first)</i> |
| 007B | 012F | -> | <i>HEX(LSW first)</i> |

or

|  |    |    |    |                          |
|--|----|----|----|--------------------------|
| <--- Hex format bytes---> (Size - 4 bytes) |    |    |    |                          |
| 2F   | 01 | 7B | 00 | -> <i>HEX(MSW first)</i> |
| 7B   | 00 | 2F | 01 | -> <i>HEX(LSW first)</i> |

**Display Format: YYMMDD-1234** - size in FLASH - 4 bytes

The format date is compressed to be able to fit data in only in two bytes as follows:

Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

<---(year-2000)----> < month><— day -->

SN 0405110123 will be saved as

YYMMDD - 040511 (Decy) -> 0x08AB (hex)

1234 - 0123 -> 0x007B (hex)

In flash memory this number can be seen as

08AB 007B -> *HEX(MSW first)*

007B 08AB -> *HEX(LSW first)*

or

<--- Hex format bytes---> (Size - 4 bytes)

AB 08 7B 00 -> *HEX(MSW first)*

7B 00 AB 08 -> *HEX(LSW first)*

**Display Format: YYDDD-1234** - size in FLASH - 4 bytes

The format date is compressed to be able to fit data only in two bytes as follows:

Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

<---(year-2000)----> < --- day of year --->

SN 041110123 will be saved as

YYDDD - 04111 (Decy) -> 0x086F (hex)

1234 - 0123 -> 0x007B (hex)

In flash memory this number can be seen as

086F 007B -> *HEX(MSW first)*

007B 086F -> *HEX(LSW first)*

or

<--- Hex format bytes---> (Size - 4 bytes)

6F 08 7B 00 -> *HEX(MSW first)*

7B 00 6F 08 -> *HEX(LSW first)*

**Display Format: 12345678** - size in FLASH - 4 bytes

SN 12345678 will be saved as

12345678 (Decy) -> 0x00BC614E (hex)

In flash memory this number can be seen as

00BC 614E -> **HEX(MSW first)**

614E 00BC -> **HEX(MSW first)**

or

<--- Hex format bytes---> (Size - 4 bytes)

00 BC 4E 61 -> **HEX(MSW first)**

4E 61 00 BC -> **HEX(LSW first)**

**Display Format:** 1234(5) - size in FLASH - 2 bytes

SN 12345 will be saved as

12345 (Decy) ---> 0x3039 (hex)

In flash memory this number can be seen as

3039 ( integer numbers ) -> **HEX(MSW first)** or **HEX(LSW first)**

or

<--- Hex format bytes---> (Size - 2 bytes)

39 30 (bytes) -> **HEX(MSW first)** or **HEX(LSW first)**

**Display Format:** Custom - size in FLASH - defined size divided by 2

Entered manually or read via Bar Code Scanner hexadecimal number is converted to HEX format and saved in flash memory in order related to MSW or LSW first selection.

E.g. entered hexadecimal number

02A569C1

will be seen as

02 A5 69 C1 -> **HEX (MSW first)**

or

C1 69 A5 02 -> **HEX (LSW first)**

### 8.2.2.2 **BCD format**

When BCD format is selected, then all SN display formats described above can be stored as a two or four separate bytes converted to BCD format, where first and last four bits of 8 bit byte contains a value from 0 to 9. All consecutive serial number characters are converted to half byte each. Finally two consecutive serial number characters will be converted to a single byte.

**Display Format: YYYY-1234**

- size in FLASH - 4 bytes

SN 20030123 will be saved as

|             |              |         |
|-------------|--------------|---------|
| YYYY - 2003 | -> 0x20 0x03 | (bytes) |
| 1234 - 0123 | -> 0x01 0x23 | (bytes) |

When flash memory bytes are viewed, then this number can be seen as

<--- Hex format bytes--->  
20 03 01 23 (Size - 4 bytes)

The consecutive serial number ( 4 bytes BCD ) can have a value from 0 to 9999 and is displayed as the 4 digit serial number.

**Display Format: YYMM-1234**

- size in FLASH - 4 bytes

SN 03030123 will be saved as

|             |              |         |
|-------------|--------------|---------|
| YYMM - 0303 | -> 0x03 0x03 | (bytes) |
| 1234 - 0123 | -> 0x01 0x23 | (bytes) |

In flash memory this number can be seen as

<--- Hex format bytes--->  
03 03 01 23 (Size - 4 bytes)

**Display Format: YYMMDD-1234**

- size in FLASH - 5 bytes

SN 0405110123 will be saved as

|                 |                   |
|-----------------|-------------------|
| YYMMDD - 040511 | -> 0x04 0x05 0x11 |
| 1234 - 0123     | -> 0x01 0x23      |

In flash memory this number can be seen as

<--- Hex format bytes--->  
04 05 11 01 23 (Size - 5 bytes)



**Display Format: YYDDD-1234** - size in FLASH - 4 bytes

The format date is compressed to be able to fit data only in two bytes as follows:

|             |   |
|-------------|---|
| Bit 15...12 | - Year number - multiple of ones (9,8,...1,0) |
| 11,10       | - Year number - multiple of tens (3,2,1,0)    |
| 9, 8        | - Day number - multiple of hundreds (3,2,1,0) |
| 7...4       | - Day number - multiple of tens (9,8,...1,0)  |
| 3...0       | - Day number - multiple of ones (9,8,...1,0)  |

SN 041110123 will be saved as

|                      |                    |
|----------------------|--------------------|
| YYDDD - 04111 (Decy) | -> 0x41 0x11 (hex) |
| 1234 - 0123          | -> 0x01 0x23 (hex) |

**Display Format: 12345678** - size in FLASH - 4 bytes

SN 12345678 will be saved as

|          |                                |
|----------|--------------------------------|
| 12345678 | -> 0x12 0x34 0x56 0x78 (bytes) |
|----------|--------------------------------|

In flash memory this number can be seen as

<--- Hex format bytes--->  
12 34 56 78 (Size - 4 bytes)

**Display Format: 1234** - size in FLASH - 2 bytes

SN 1234 will be saved as

|      |                      |
|------|----------------------|
| 1234 | -> 0x12 0x34 (bytes) |
|------|----------------------|

In flash memory this number can be seen as

<--- Hex format bytes--->  
12 34 (Size - 2 bytes)

### 8.2.2.3 ASCII format

When Ascii format is selected, then all SN display formats described above can be stored as a four or eight separate bytes converted to Ascii characters. All consecutive serial number characters are converted to Ascii characters.

#### Display Format: YYYY-1234

- size in FLASH - 8 bytes

SN 20030123 will be saved as

YYYY - 2003

-> 0x32 0x30 0x30 0x33 (bytes)

or '2' '0' '0' '3'

1234 - 0123

-> 0x30 0x31 0x32 0x33 (bytes)

or '0' '1' '2' '3'

When flash memory bytes are viewed, then this number can be seen as

<----- Hex format ----->

32 30 30 33 30 31 32 33

<- Ascii format ->

20030123

(Size - 8 bytes)

#### Display Format: YYMM-1234

- size in FLASH - 8 bytes

SN 03030123 will be saved as

YYMM - 0303

-> 0x30 0x33 0x30 0x33 (bytes)

or '0' '3' '0' '3'

1234 - 0123

-> 0x30 0x31 0x32 0x33 (bytes)

or '0' '1' '2' '3'

In flash memory this number can be seen as

<----- Hex format ----->

30 33 30 33 30 31 32 33

<- Ascii format ->

03030123

(Size - 8 bytes)

#### Display Format: YYMMDD-1234

- size in FLASH - 10 bytes

SN 0405110123 will be saved as

YYMMDD - 040511

-> 0x30 0x34 0x30 0x35 0x31 0x31 (bytes)

or '0' '4' '0' '5' '1' '1'

1234 - 0123 -> 0x30 0x31 0x32 0x33 (bytes)  
or '0' '1' '2' '3'

In flash memory this number can be seen as

|                               |                              |
|-------------------------------|------------------------------|
| <----- Hex format ----->      | <- Ascii format ->           |
| 30 34 30 35 31 31 30 31 32 33 | 0405110123 (Size - 10 bytes) |

**Display Format: YYDDD-1234** - size in FLASH - 9 bytes

SN 042140123 will be saved as

YYDDD - 04214 -> 0x30 0x34 0x32 0x31 0x34 (bytes)  
or '0' '4' '2' '1' '4'

1234 - 0123 -> 0x30 0x31 0x32 0x33 (bytes)  
or '0' '1' '2' '3'

In flash memory this number can be seen as

|                            |                            |
|----------------------------|----------------------------|
| <----- Hex format ----->   | <- Ascii format ->         |
| 30 34 32 31 34 30 31 32 33 | 042140123 (Size - 9 bytes) |

**Display Format: 12345678** - size in FLASH - 8 bytes

SN 12345678 will be saved as

12345678 -> 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 (bytes)

In flash memory this number can be seen as

|                          |                           |
|--------------------------|---------------------------|
| <----- Hex format -----> | <- Ascii format ->        |
| 31 32 33 34 35 36 37 38  | 12345678 (Size - 8 bytes) |

**Display Format: 1234** - size in FLASH - 4 bytes

SN 1234 will be saved as

1234 -> 0x31 0x32 0x33 0x34 (bytes)

In flash memory this number can be seen as

|                          |                    |                  |
|--------------------------|--------------------|------------------|
| <----- Hex format -----> | <- Ascii format -> |                  |
| 31 32 33 34              | 1234               | (Size - 4 bytes) |

**Display Format:** Custom - size in FLASH - defined size in bytes  
Entered manually or read via Bar Code Scanner ascii string will be saved in flash memory  
“as is”. E.g. entered hexadecimal number  
02WX24S234  
will be seen as  
30 32 57 58 32 34 53 32 33 34 -> “02WX24S234”

Location in the target device’s flash memory, where described above bytes are saved, is specify in the ‘**Memory Location - SN Start Address in Memory**’ field of the serialization dialogue screen (see figure 8.2-1). Specified address must be even and should be specified in the empty memory space, not used by program code or data block

When software detects that any serial number character is using memory location used by code file, then the following error message will be displayed:

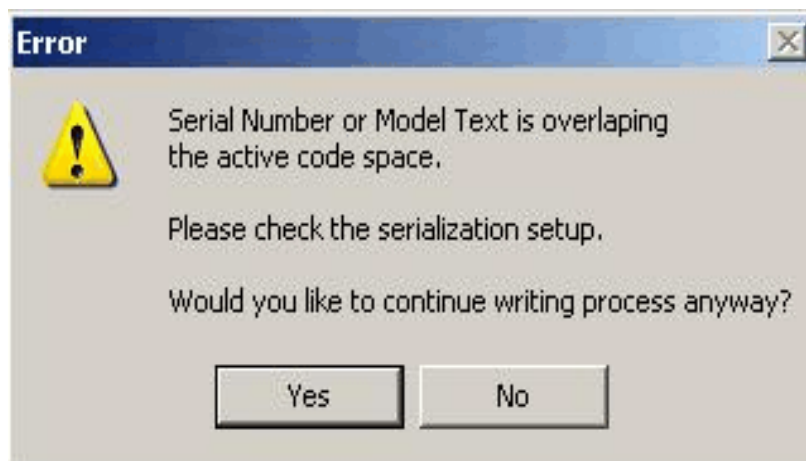


Figure 8.2.1-1

### 8.2.3 *Model, Group, Revision*

Custom text, saved in target device's flash memory is a string, up to 32 characters long, in Ascii format. It can contain any text, but this feature is intentionally created to allow the hardware model, revision and group to be saved. Typically the object code does not contains this kind of information, but it may be useful in some applications.

This feature is enabled when the check box **ENABLE** in the **Model/Group/Revision** field is marked (see figure 8.2-1). When enabled, the size of desired text must be specified in the field '**Text size in bytes**'. Size value can be any *even* number between 2 and 32. The location of the text in the flash memory can be specified in the field '**Start Address in Memory**'. Similarly to the location of the serial number, the specified address must be even and must be specified in the empty memory space, unused by program code or data block. Otherwise, the error message shown in figure 8.2.1-1 will be displayed.

The text to be saved in the flash memory can be entered in the '**Model/Group/Revision text**' edit line. If the size of entered text exceeds the size specified in the '**Text size in bytes**' field, then all character that do not fit in the allocate space will be truncated.

### 8.2.4 *Device Serialization box*

Device Serialization box, located on the main programming dialogue screen (see figures 10-2 and 4-1), contains serial number and model information. The first two read only lines contain information taken from the target device. The next two lines contain model text and serial number that are to be saved. Whenever a communication with the target device is performed the model text and serial number is read and displayed in the Device Serialization group.

The '**Next Model-Group\_Revision**' and '**Next SN**' edit lines can contain any SN and text. When the device is programmed the next model text is taken from the '**Model/Group/Revision Text**' of the **Serialization** dialogue screen. The next SN is generated automatically, according to the setup in the **Serialization** . This means that any data entered in the '**Device Serialization**' group can be treated as temporary data. This data is downloaded to only one target device.

Current target's label (model text and serial number) can be read at any time by pressing **READ SN** button located in the '**Device Serialization**' group (see figure 8-2).

### 8.2.5 Bar Code Scanner setup

Programming software has capability to get a data from the Bar Code Scanner. Bar Code Scanner should be connected to PC computer in series with the keyboard using the Y cable or to the USB port. Refer to the Bar Code Scanner manual for details.

Bar Code Scanner when enabled by selecting the **ENABLE** in the **BarCode Scanner** group then can enter scanned data directly to the "**Next SN:**" edit line. When the new SN is entered then **AUTOPROGRAM** function can be started automatically if "**Start AUTOPROGRAM following BarCode scan**" is selected.

By default Bar Code Scanner is sending the **CR (ENTER)** character as a termination character following the scanned message. From the "**Terminator Character**" selector is possible to get other termination character then **CR** if required.

*Note: Only Ascii characters from 0x21 to 0xFE are accepted from the Bar Code Scanner. Others characters like white characters (space, tab) are ignored. All characters are converted to the lower case characters.*



Figure 8-2

### 8.3 *Serialization Report Dialogue Screen*

Serialization Report Dialogue Screen reports the results of the serialization procedure. The report contains the detailed information of the two highest serial number programmed units, quantity of programmed units along with the new created serial numbers, unmodified SN (reprogrammed units), manually created SN and quantity of the overwritten SN. Detailed information about all programmed units can be viewed using the Notepad text editor by pressing the '**NotePad**' button.

Short information of the created serial numbers, format, date and time of programming is displayed on the white report box (see Figure 8.3-1). Serial numbers are created automatically via software by incrementing the highest SN taken from the serial number files. If from any reason the highest serial number is wrong it can be removed from the database by pressing the '**Delete SN**' button. Note that the delete operation is not reversible.

Serialization File Name: E:\ElproTronic\MFG\test2.sn

The Highest Serial Number

SN: 404231204 Overwritten Model,Group, Rev: -33 R.0004-04-17

Date: Fri, Apr 23, 2004, 20:12 Code File Name: AS010x02-1 v4.txt Delete SN

Serial Number before the Highest SN

SN: 404231203 Overwritten Model,Group, Rev: -33 R.0004-04-17

Date: Fri, Apr 23, 2004, 20:07 Code File Name: AS010x02-1 v4.txt

Report

70 ☒ New SN

1 ☒ Unmodified SN

0 ☒ Manually entered SN

91 ☒ Overwritten SN

NotePad

| Format | Serial Number | Date, Time          |
|--------|---------------|---------------------|
| F0     | 20041269      | APR 10, 2004, 22:46 |
| F0     | 20041270      | APR 10, 2004, 22:47 |
| F0     | 20041271      | APR 10, 2004, 22:48 |
| F0     | 20041272      | APR 10, 2004, 22:48 |
| F0     | 20041273      | APR 10, 2004, 22:49 |
| F0     | 20041274      | APR 10, 2004, 23:06 |
| F0     | 20041275      | APR 11, 2004, 10:56 |
| F0     | 20041276      | APR 11, 2004, 17:12 |
| F0     | 20041277      | APR 11, 2004, 17:12 |
| F0     | 20041278      | APR 11, 2004, 17:25 |
| F0     | 20041279      | APR 11, 2004, 17:27 |
| F0     | 20041280      | APR 11, 2004, 17:31 |
| F0     | 20041281      | APR 11, 2004, 17:36 |
| F0     | 20041282      | APR 11, 2004, 17:36 |

Legend:

SN Formats:

F0 - YYYY1234(5)

F1 - YYMM1234(5)

F5 - YYMMDD1234(5)

F4 - YYDDD1234(5)

F2 - 12345678

F3 - 1234(5)

F6 - Custom

Delete Last SN

Exit

Figure 8.3-1 Serialization Report Dialog screen

## 9. Load/Save Setup

---

Programming software can save configuration settings. This allows the user to create several configuration file, one for a particular task, and thus eliminates the need to manually change settings every time a different configuration is desired. Furthermore, the config.ini file contains the most recently used settings and those settings will be used as default whenever the software is started.

To create a configuration file simply select **Save Setup** from the **File** menu. Current settings will be saved for future use. To restore configuration settings select **Load Setup** from **File** menu and select a file containing the settings you wish to restore.

In order to prevent accidental setup changes the MSP430 Programmer provides the option to Lock configuration settings. When the user selects the **Lock/Unlock Setup** option from the Setup menu, the MSP430 Flash Programmer will prevent the user from modifying the setup. The only options that are available when the programmer is locked are **Verify**, **Read**, **Autoprogram** and **Next**. Notice that the **Next** button will immediately change to implement the **Autoprogram** function. To unlock the programmer the user must select the **Lock/Unlock Setup** option from the Setup menu.

Configuration setup file can be opened using **Load Setup** option from **File** menu or can also be opened using command line combined with the executable file name. Following command line switches are available

**-sf file\_name**

**-lock**

Using Windows **START** button (left bottom) select **Run..** Using **Browse..** find and select executable file (see Figure 9.1)

"C:\Program Files\Elprotronic\USB FlashPro430\USB-MSP430-Prg.exe"

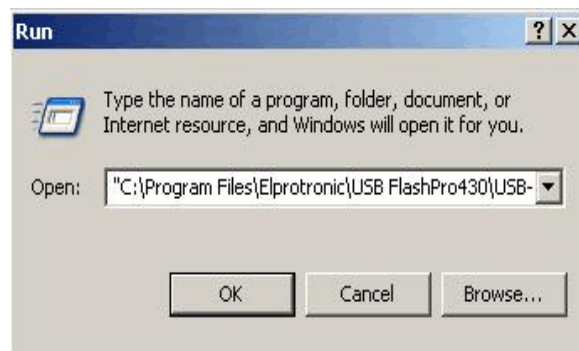


Figure9.1



and at the end enter the required key with name of the setup file eg.

```
"C:\Program Files\Elprotronic\USB FlashPro430\USB-MSP430-Prg.exe" -sf E:\ElproTronic\MFG\prg-04.cfg
```

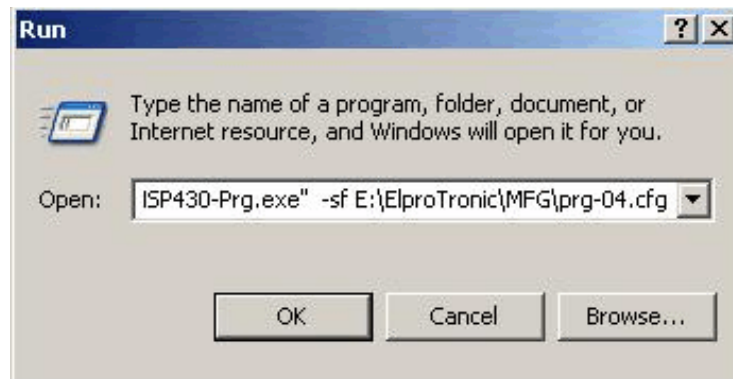


Figure 9.2

To fully lock the configuration setup the extra key “-lock” can be added in the command line eg.

```
"C:\Program Files\Elprotronic\USB FlashPro430\USB-MSP430-Prg.exe" -lock -sf E:\ElproTronic\MFG\prg-04.cfg
```

or

```
"C:\Program Files\Elprotronic\USB FlashPro430\USB-MSP430-Prg.exe" -sf E:\ElproTronic\MFG\prg-04.cfg -lock
```

Following configuration setup can be created using *Shortcut* options that allows to create a lot of icons located on the desktop - each icon with required independent configuration setup. To do that move the cursor to inactive desktop area, click right mouse button and select *New* (see Figure 9.3)

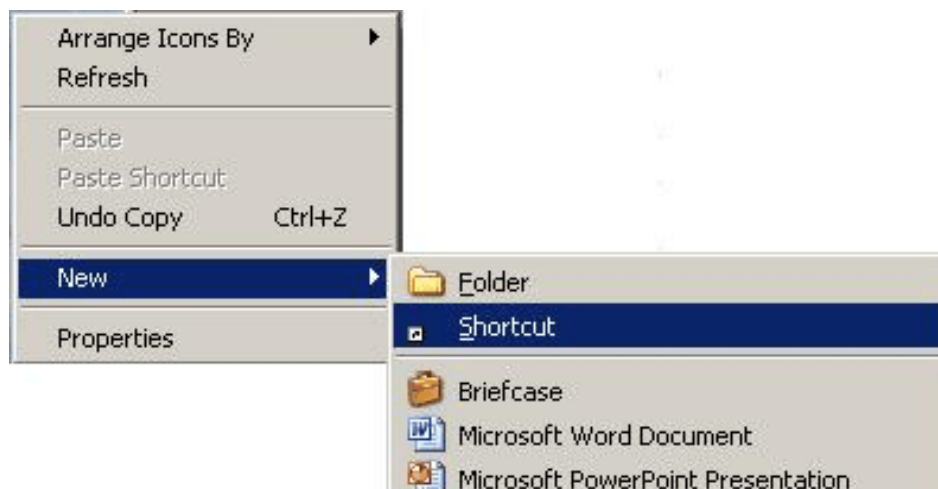


Figure 9.3

Using Browse.. in the Create Shortcut dialogue box select the following executable file

"C:\Program Files\Elprotronic\USB FlashPro430\USB-MSP430-Prg.exe"

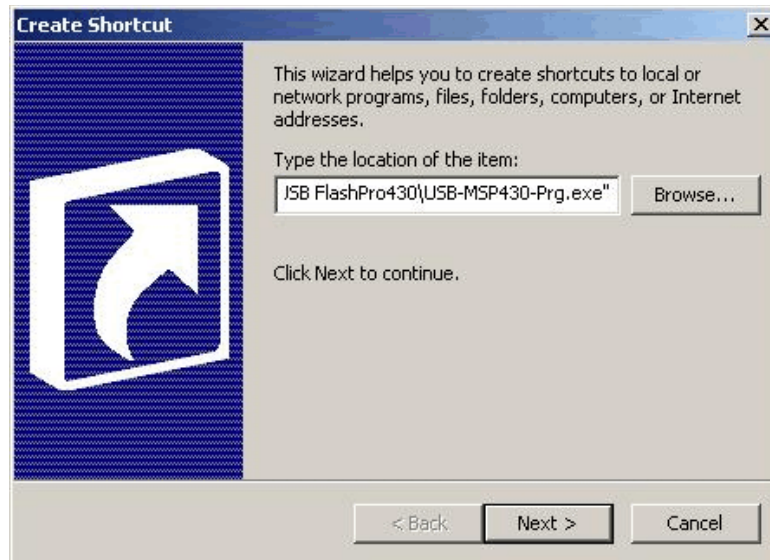


Figure 9.4

(see Figure 9.4) and at the and add the required command keys (see Figure 9.5) eg.

"C:\Program Files\Elprotronic\USB FlashPro430\USB-MSP430-Prg.exe" -lock -sf E:\MFG\prg-04.cfg



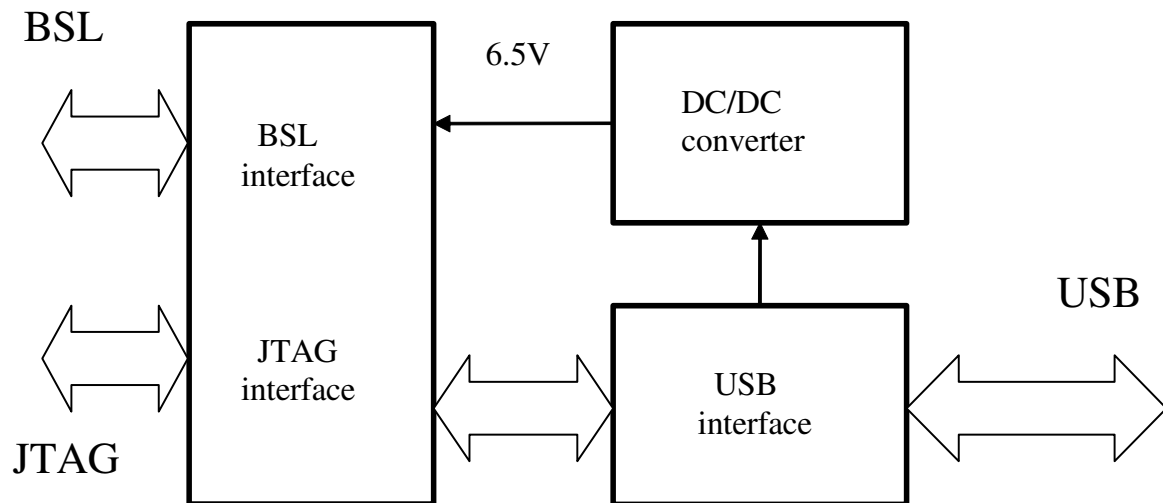
Figure 9.5

Click button *Next* and follow instruction to create icon. Using *Copy* and *Paste* and modify required configuration file names a lot of icons can be created with independent configuration setups. Clicking on the selected icon FlashPro430 programming software will start with the selected configuration setup, and locked if required.

## 10. Adapter description

---

**FlashPro430** programming adapter contains an interface for the MSP430F149 microcontroller, a 3.3V low voltage drop regulator, SPST switch to supply Vcc, flat cable with 10 pins socket connector for BSL Interface, 14 pins socket connector for JTAG/BSL Interface and DB-25 male connector, as shown in figure 10-1.



**Figure 10-1:** Block diagram of the USB-JTAG/BSL adapter.

**FlashPro430** programming adapter is powered from the USB port. The adapter's current consumption is below 50 mA. Due to the low power requirement of the **FlashPro430** and to the ultra-low power of the MSP430Fxx programmed microcontroller, a stand-alone adapter with a target microcontroller does not exceed the available current from the USB port.

**FlashPro430** contains 3.3V voltage regulator with limited output current to 100mA dedicated to supply the target device. However, if additional circuitry is added to the target board the current requirements can exceed 100mA. In this case the microcontroller should be powered from its own power supply, battery, or external power supply. The DC voltage required to power the MSP430Fxx

microcontroller should be between 2.7V and 3.6V. Although the MSP430Fxx microcontroller can run on DC voltage as low as 1.8V, this low voltage is insufficient to reliably program FLASH memory.

All data I/O pins on the adapter to target connection are protected and have output impedance of 220 ohms to 1k $\Omega$ . Input impedance of the BLTx-In and TDO-In is higher than 100 k $\Omega$ . Target board input resistance of the RST, BLRX, TEST, TDI, TMS and TCK inputs should be at least 20 k $\Omega$ . Output impedance of the BLTX and TDO output should not exceed 1k $\Omega$ .

# 11. Target connection

---

The Fast USB MSP430 Flash Programmers with the JTAG and the BSL Interfaces use the **STANDARD 14-pin TI-JTAG** connector's pinout to facilitate the JTAG communication. Some of the unused pins on this connector are utilized to facilitate Bootstrap Loader (BSL) communication.

Texas Instruments created the standard for the MSP430 JTAG interface connector and for the Bootstrap Loader (BSL) connector. The JTAG and the BSL connectors share several common signals, such as RST, GND, Vcc, Vcc-Out, TEST and TCK. The BSL connector uses signals BSL-Tx and BSL-Rx that are not found in the JTAG connector. We can notice that the TI-JTAG connector has specified a maximum of 11 pins, and the remaining 3 pins are not used. These three pins can be used for the two BSL signals, namely BSL-Tx and BSL-Rx. By utilizing these unused pins enables us to facilitate both the JTAG and the BSL communication interfaces on a SINGLE 14-pin JTAG connector.

This modification **DOES NOT** affect the JTAG adapter, as the pins assigned to the BSL-Tx and BSL-Rx signals are unused by the JTAG Interface. This can save one connector and can simplify communication with the target device.

The pinout for the standard JTAG connector with added BSL-Tx and BSL-Rx signals is shown in figure 11-1. BSL-Tx and BSL-Rx signals are connected to the pins 12 and 14 respectively. In addition, a ground line is connected to pin 13. The JTAG signal lines are connected to pins number 1 through 11 in compliance with the standard JTAG specification provided by Texas Instruments.

The definition of all the pins is given in the tables 11-1.

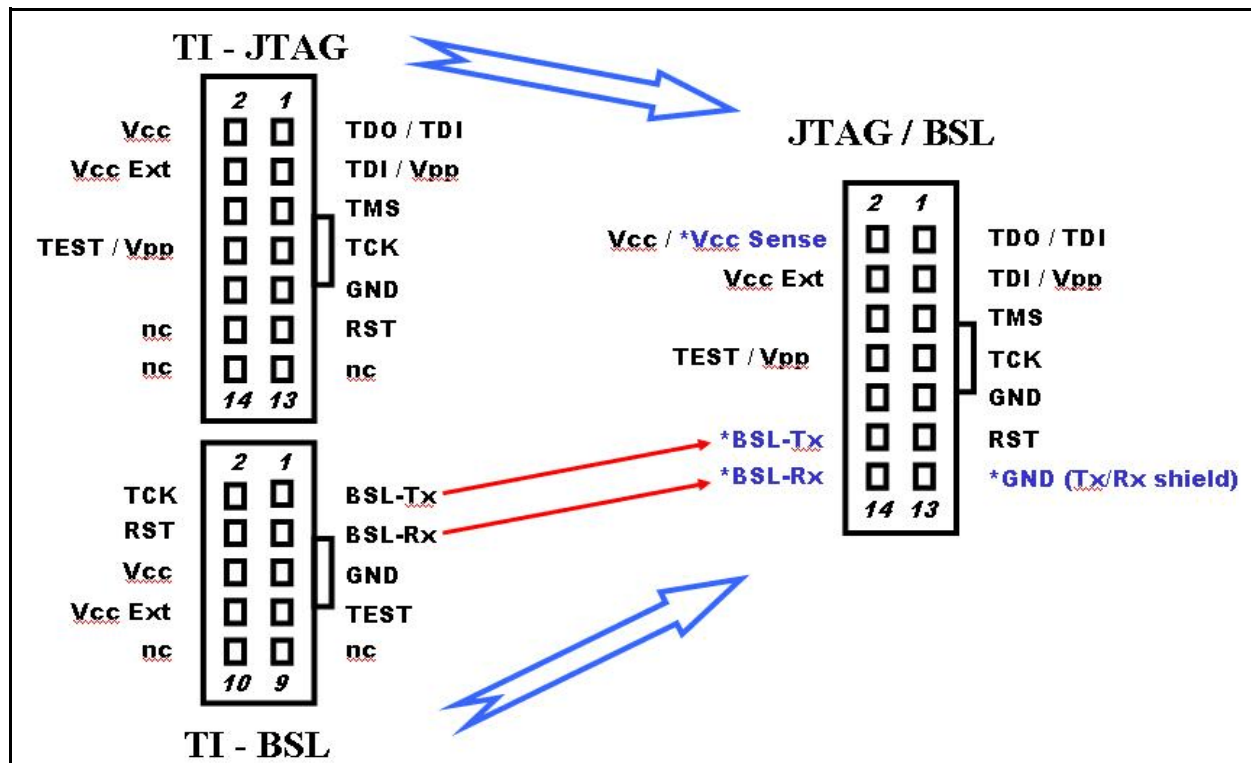


Figure 11-1. JTAG/BSL header connector - solder side.

Table 11.1 JTAG/BSL Interface connector

| Pin #   | Name           | Description   |
|---------|----------------|---|
| 1 (Red) | TDO/TDI        | Data output   |
| 2       | VCC-In / Sense | Vcc supplied to the target ( 3.3V/ max 100 mA) and the target's Vcc voltage sense.<br>This pin should be connected to Vcc of the microcontroller if microcontroller is supplied from the Flash Programming Adapter. If the Target's Device microcontroller is energized from his own battery or external power supply then the pin 2 or 4 (Vcc sense) should be connected to the Vcc of the microcontroller.<br><b>Note</b> , that in the old version of the USB-MSP430-FPA- rev. 1.0 the pin 4 is not connected.<br>Vcc sense is available on the pin number 2 only. |
| 3       | TDI-Vpp        | Data Input - Blow Fuse voltage Vpp (+6.5V)  |
| 4       | Sense          | Target's Device Vcc Sense<br>( used in the USB-MSP430-FPA - rev.1.1 and up )<br>( not used in the USB-MSP430-FPA rev.1.0 )  |
| 5       | TMS-In         | TMS Input   |
| 6       | NC             | Not used.   |
| 7       | TCK-In         | Boot Strap Loader / JTAG TCK Input pin (note-3)   |
| 8       | TEST-Vpp       | Boot Strap Loader / JTAG TEST Input pin, Blow Fuse voltage Vpp (+6.5V) (note-4)   |
| 9       | GND            | Ground  |
| 10      | NC             | Not used.   |
| 11      | VRST           | Microcontroller Reset Input pin.  |
| 12      | BLTX-Out (*)   | Boot Strap Loader Tx Output from the target (note-1)  |
| 13      | GND (*)        | Ground  |
| 14      | BLRx-In (*)    | Boot Strap Loader Rx Input to the target (note-2)   |

**NOTE (\*):** Pins numbers 12,13 and 14 of the JTAG connector has modified connection compared to the standard TI JTAG FET adapter. Typically those pins are not used in TI JTAG FET but has been used in the **FlashPro430**. Programming adapter to pass the Tx and Rx signals of the BSL communication port. When this modification is done, then one modified 14-pins JTAG connector can be used for JTAG and BSL communication between target device and programming adapter.

**Note-1.** BLTX-Output - Transmit data output pin from the bootstrap loader.

Port pin 1.1 for microcontrollers MSP430F1xx.

Port pin 1.0 for microcontrollers MSP430F4xx.

**Note-2** BLRx-Input - Receive data input pin to the bootstrap loader.

Port pin 2.2 for microcontrollers MSP430F1xx.

Port pin 1.1 for microcontrollers MSP430F4xx.

**Note-3** TCK-Input - for BSL used only for microcontrollers with package over 28 pins.

**Note-4** TEST-Input - for BSL used only for microcontrollers with package up to 28 pins.



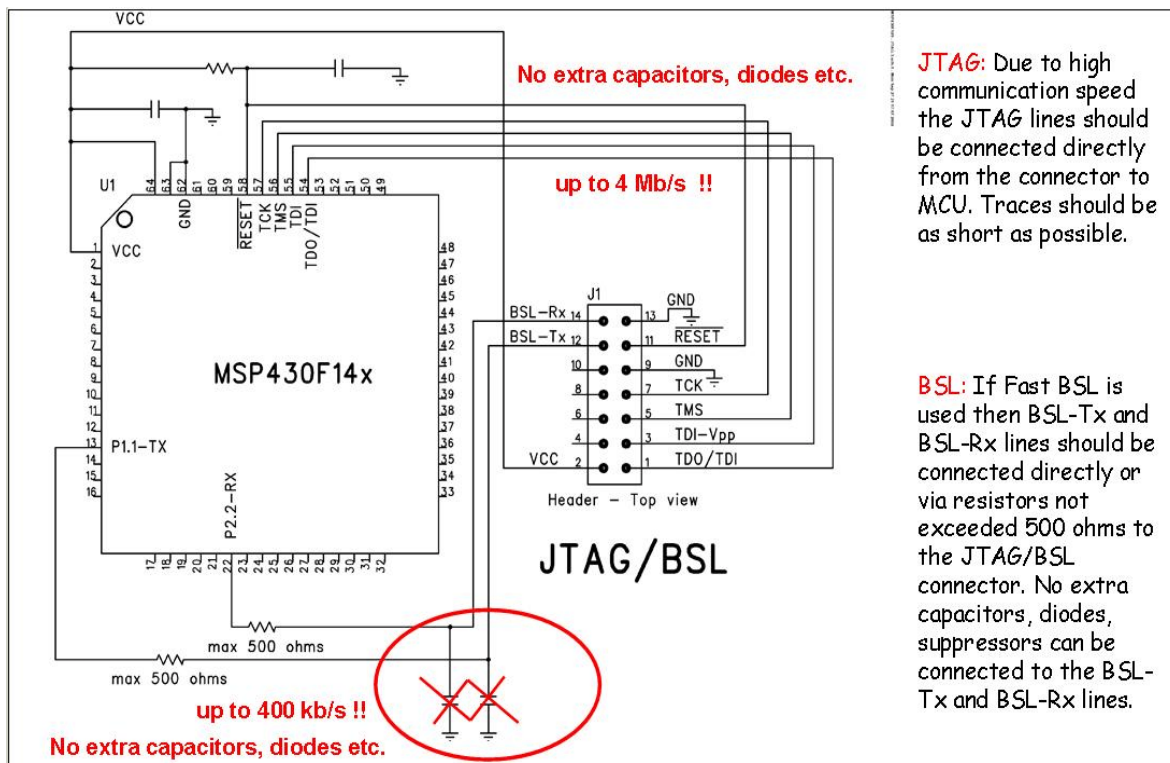


Figure 11-2

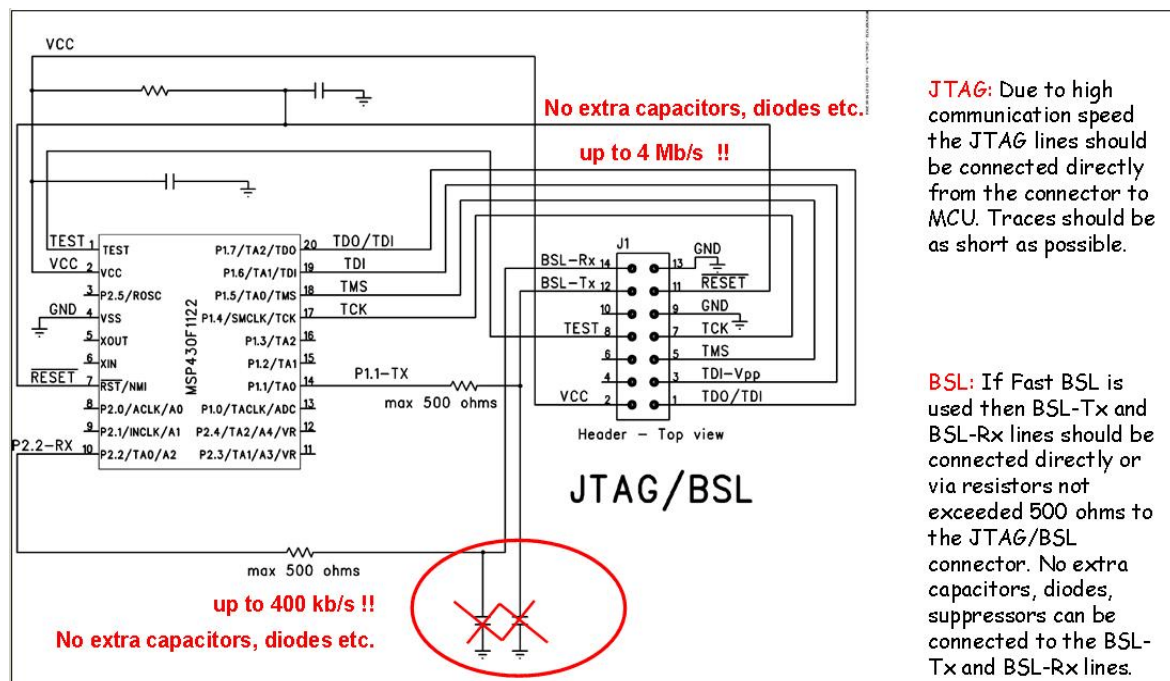


Figure 11-3

Figure11-2 show interconnection between modified JTAG connector and MSP430F149 microcontroller. Two lines - BLRx and BLTx have been connected to not used pins 12 and 14 of the standard JTAG connector. This modification allows use of the only one modified JTAG connector to connect the JTAG and BSL communication ports to programming adapter.

Figure11-3 show interconnection between modified JTAG connector and MSP430F1122 microcontroller.

Refer to the Texas Instruments data sheet for detailed information related to pin numbering of a particular microcontroller.

In a typical design all lines are connected directly between BSL connector and the dedicated microcontroller's pins. In some designs, protecting components are added, such as resistors, capacitors, diodes or suppressors. Care should be taken, when such components are added, especially on the data Rx and Tx lines. When the standard BSL communication speed is used between the target microcontroller and programming adapter (9.6kb/s), then the protecting circuitry in the data path will not affect the communication between them. If a faster communication speed is used with MSP430 Fast BSL programming adapter, then the protecting circuitry can create a problem. This prevents the use of 350kb/s communication speed. In extreme cases the user may need to revert to standard BSL speed. It is recommended that maximum resistance between Tx/Rx microcontroller pins and BSL connector does not exceed 500  $\Omega$ . Additional capacitors, if installed between data path and ground, should not exceed 50pF.

Due to high communication speed (up to 4Mbit/s) between Programming Adapter and the target's device when JTAG Interface is used the target's device JTAG lines should be connected directly to the JTAG/BSL connector without extra components to avoid communication problem. If from any reason the target device contains extra components like capacitors or suppressors in the JTAG lines then the slower JTAG communication speed can be selected. The **FlashPro430** software allows to select 4Mb/s, 1Mb/s and 400 kb/s JTAG communication speed between programming adapter and target device.

# ***Appendix A - specification***

---

## **Specification:**

|  |  |
|--|--|
| PC Communication Interface:            | - Full Speed USB-1.1 (12Mbps/s)  |
| USB connector                          | - Adapter site: USB-type B, Computer site: USB-type A  |
| Target connector                       | - 14 pins header connector - standard JTAG pinhead with added BSL connection to unused pins.                 |
| DC Power - from USB Interface          | - 5V +/- 20%, 50mA + target's current (0-100mA)  |
| Target Device DC supply                |  |
| - external                             | - 2.7 V to 3.6 V   |
| - from programming adapter             | - 3.3 V / 100 mA max.  |
| Communication speed via JTAG Interface |  |
| Flash programming                      | - up to 29 kbytes/s  |
| Flash/RAM uploading                    | - up to 80 kbytes/s  |
| Ram downloading                        | - up to 130 kbytes/s   |
| Communication speed via BSL Interface  |  |
| Flash programming                      | - up to 20 kbytes/s  |
| Flash/RAM uploading                    | - up to 19 kbytes/s  |
| Size:                                  | - 76 x 43 x 20 mm ( 3.0 x 1.68 x 0.8 inch )  |
| Verification Compliance:               | - CE ( European CISPR 22 and EN 55022 ).   |
|  | - FCC Part 15, Subpart B- Class B Unintentional Radiators for Uses in Home, Commercial and Industrial Areas. |

# Appendix B - performance

Table B-1

*Programming / reading times for a target microcontroller with 60 kB of Flash Memory*

| <i>Interface type =&gt;<br/>Communication Speed [ kb/s ] =&gt;</i>   | <i>BSL<br/>350 kb/s</i> | <i>BSL<br/>9.6 kb/s</i> | <i>JTAG</i>  |
|--|-------------------------|-------------------------|--------------|
| <i>1. Flash Programming time with Fast Verification</i>              | <i>5.0 s</i>            | <i>99 s</i>             | <i>2.3 s</i> |
| <i>2. Flash Reading time</i>   | <i>5.2 s</i>            | <i>127 s</i>            | <i>1.4 s</i> |
| <i>3. Flash Erasing time</i>   | <i>0.5 s</i>            | <i>0.5 s</i>            | <i>0.4 s</i> |
| <i>4. Flash Blank Checking time</i>                                  | <i>1.4 s</i>            | <i>127 s</i>            | <i>0.4 s</i> |
| <i>5. Standard Verification time</i>                                 | <i>5.2 s</i>            | <i>127 s</i>            | <i>1.2 s</i> |
| <i>6. Auto Program with fast verification time <sup>1)</sup></i>     | <i>6.3 s</i>            | <i>-</i>                | <i>3.5 s</i> |
| <i>7. Auto Program with standard verification time <sup>1)</sup></i> | <i>10 s</i>             | <i>226 s</i>            | <i>4.0 s</i> |
| <i>8. Blowing Security Fuse time</i>                                 | <i>-</i>                | <i>-</i>                | <i>0.2 s</i> |

*Note: \* - add 0.2 second, if blow the Security Fuse is enabled.*

*Auto Program performs:*

*Target DC voltage verification,  
Communication initialization,  
Fuse checking or password verification,  
Memory erase (all memory, or specified sectors memory),  
Blank checking,  
Flash programming and fast verification  
Serial Number and Model writing ( if selected ),  
Check sum verification,  
Standard verification ( if selected ).  
Blowing Security Fuse (if selected)*

*Specified times include target DC voltage verification, communication initialization and access verification.*

***Test conditions:*** *PC Pentium III 600MHz with 512 MB RAM, Operating System - Windows<sup>TM</sup> XP.*

Supported microcontrollers in a MSP430 family:

MSP430F110, MSP430F1101, MSP430F1101A, MSP430F1111, MSP430F1111A, MSP430F112, MSP430F1121, MSP430F1121A, MSP430F1122, MSP430F1132, MSP430F122, MSP430F1222, MSP430F123, MSP430F1232, MSP430F133, MSP430F135, MSP430F147, MSP430F1471, MSP430F148, MSP430F1481, MSP430F149, MSP430F1491, MSP430F155, MSP430F156, MSP430F157, MSP430F167, MSP430F168, MSP430F169, MSP430F1610, MSP430F1611, MSP430F412, MSP430F413, MSP430F415, MSP430F417, MSP430F423, MSP430F425, MSP430F427, MSP430F435, MSP430F436, MSP430F437, MSP430F447, MSP430F448,

MSP430F449, MSP430FE423, MSP430FE425, MSP430FE427, MSP430FG437, MSP430FG438,  
MSP430FG439, MSP430FW423, MSP430FW425, MSP430FW427 .

## Appendix C - software options

---

Programmer is supplied with three software options - Lite-BSL, Lite-JTAG and standard software. Each option provides different features, summarized in table C-1.

Table C-1. Features of limited and standard versions.

| No | Name  | Lite-BSL | Lite-JTAG | Standard |
|----|---|----------|-----------|----------|
| 1  | BSL Communication Interface                         | +        | -         | +        |
| 2  | JTAG Communication Interface                        | -        | +         | +        |
| 3. | BSL Communication Speed - 9.6 kb/s<br>350 kb/s      | +        | n/a       | +        |
|    |   | +        | n/a       | +        |
| 4. | Serialization, Model, Rev etc.                      | -        | -         | +        |
| 5  | Memory Erase - All memory                           | +        | +         | +        |
|    | - Info Memory only                                  | -        | -         | +        |
|    | - Main Memory only                                  | -        | -         | +        |
|    | - Used in Code File                                 | -        | -         | +        |
|    | - User Defined                                      | -        | -         | +        |
| 6  | Write Memory - All memory                           | +        | +         | +        |
|    | - Limited to Info Memory only                       | -        | -         | +        |
|    | - Limited to Main Memory only                       | -        | -         | +        |
|    | - User Defined                                      | -        | -         | +        |
| 7  | Blank check, Check Sum verification                 | +        | +         | +        |
| 8  | Flash memory read                                   | +        | +         | +        |
| 9  | Counter of programmed devices<br>( Total, Balance ) | +        | +         | +        |
| 10 | Configuration save / load                           | -        | -         | +        |
| 11 | Blow Security Fuse                                  | -        | +         | +        |
| 12 | API - DLL Library                                   | -        | -         | +        |

note: n/a - not applicable.