FlashPro430

USB-MSP430 Flash Programmer

User's Manual

Software version 4.91

PM010A04 Rev.27
06-October-2012

Elprotronic Inc.

16 Crossroads Drive
Richmond Hill,
Ontario, L4E-5C9
CANADA

Web site:	www.elprotronic.com
E-mail:	info@elprotronic.com
Fax:	905-780-2414
Voice:	905-780-5789

END USER LICENSE AGREEMENT

PLEASE READ THIS DOCUMENT CAREFULLY BEFORE USING THE SOFTWARE AND THE ASSOCIATED HARDWARE. ELPROTRONIC INC. AND/OR ITS SUBSIDIARIES (“ELPROTRONIC”) IS WILLING TO LICENSE THE SOFTWARE TO YOU AS AN INDIVIDUAL, THE COMPANY, OR LEGAL ENTITY THAT WILL BE USING THE SOFTWARE (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 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, 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 many computers;
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 a network

YOU MAY NOT:
A. sublicense, reverse engineer, decompile, disassemble, modify, translate, make any attempt to discover the Source Code of the Product; or create derivative works from the Product;
B. redistribute, in whole or in part, any part of the software component of this Product;
C. use this software with a programming adapter (hardware) that is not a product of Elprotronic Inc.

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 appereil numerique de la classe B respecte toutes les exigences du Reglement sur le material brouilleur du Canada.

Table of Contents

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 (4-wires), SBW (Spy-Bi-Wire) (2-wires) or via the BSL (Boot Strap Loader) Interface. MSP430Fxx microcontrollers can be program via the JTAG or Spy-Bi-Wire 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.

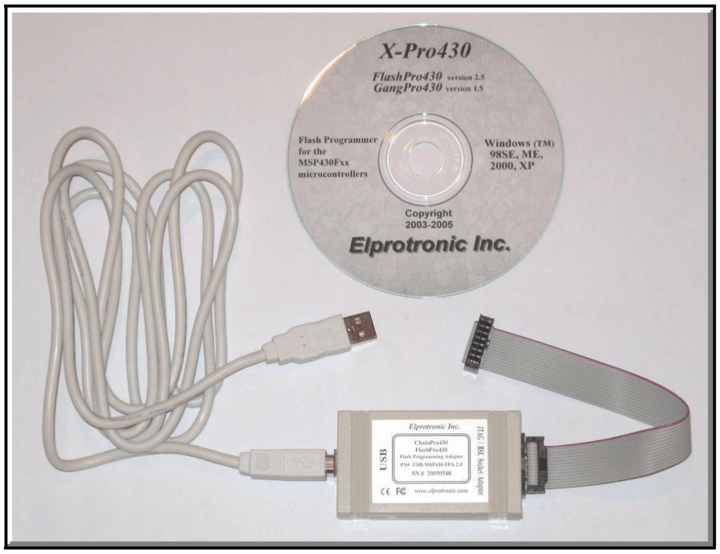


Figure 1-1

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 serials 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 (Oct-2012) seven types of the USB Flash Programming Adapters are supported by the FlashPro430 software (from Elprotronic):

1. ***FlashPro430 FPA*** (obsolete)	(model PN: USB-MSP430-FPA-1.x)
2. ***FlashPro430 FPA*** (obsolete)	(model PN: USB-MSP430-FPA-2.x)
3. ***FlashPro430 FPA (RoHS)***	(model PN: USB-FPA-4.x)
4. ***GangPro430 FPA***	(model PN: USB-MSP430-FPA-3.x)
5. ***GangPro430 FPA (RoHS)***	(model PN: USB-FPA-5.x)
6. ***FlashPro430 FPA (RoHS)***	(model PN: USB-FPA-6.x)
7. ***GangPro430 FPA (RoHS)***	(model PN: USB-FPA-6.x)

All 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 ->	***USB-MSP430-FPA-1.x***	***USB-MSP430-FPA-2.0*** and ***USB-FPA-4.x***	***USB-MSP430-FPA-3.0*** ***USB-FPA-5.x*** (GangPro430 FPA)	***USB-FPA-6.x*** (FlashPro430 or GangPro430 FPA)
1. Vcc from FPA	3.2 V	2.2 to 3.6V step 0.2V	2.2 to 3.6V step 0.2V	1.8 to 3.6V step 0.1V
2. External Vcc	1.8 to 3.6V	1.8 to 3.6V	1.8 to 3.6V	1.8 to 3.6V
3. JTAG interface	YES	YES	YES	YES
speed 4 Mb/s	YES	YES	NO	YES
speed 1 Mb/s	YES	YES	YES	YES
speed 400 kb/s	YES	YES	YES	YES
4. Spy-Bi-Wire interface	YES **	YES	YES	YES
5. BSL Interface	YES	YES	NO	YES

Note ** : ***In the old adapters - USB-MSP430-FPA-1.0, 1.1 and 1.2 the blow security fuse is not supported when the Spy-Bi-Wire is used. See chapter 13 for details.***

All other performance not listed in the table above are the same when used with the ***FlashPro430*** programming software. Note, that the ***GangPro430 FPA*** (USB-MSP430-FPA-3.0 or USB-FPA-5.x) can also be used with the ***GangPro430*** software to access few target device from one FPA adapter.

2. Features

FlashPro430 programmer is dedicated to program the Texas Instruments MSP430Fxx microcontroller family via JTAG, Spy-Bi-Wire or BSL interface.

When the JTAG or Spy-Bi-Wire Interface is used, then the ***FlashPro430*** programmer is using the standard JTAG/SBW 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 / Spy-Bi-Wire interface, 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/SBW communication port are as follows:

- support all MSP430 flash-based devices,
- programming speed approximately 29 kB/s (via JTAG interface),
- 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/SBW),

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** - three interfaces in one package allowing to program the target device via **JTAG, Spy-Bi-Wire** 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** then 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.
* No code size limitations.
* Full memory or sector memory erase capability.
* Write Check Sum verification.
* DCO calibration via JTAG, Spy-Bi-Wire and BSL
* Target device can be powered from the programming adapter or from external source.
* Easy to use Windows^{TM} 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/SBW/BSL Interface using TI-standard 14-pins header connector.

2.1 Customized features

FlashPro430 programmer can be controlled from external software or programming sequences can be customized. These features are very useful in production environment. Standard programming software FlashPro430 has a lot of options described above, but of course it can not cover all customer's requirements.

2.1.1 Encrypted Project option

Contents of the project that include code contents downloaded to target device can be encrypted and blocked against unauthorised access.

2.1.2 Script file

To extend programming features programming software supports user defined programming sequences saved in the script file. That easy method can be created by any user without knowing programming languages and techniques. Programming sequence up to 1000 lines can be created. All lines contains sequence of the pressed buttons with extra few condition options. This programming method is described in chapter 10 on this manual. Script file option is not available in ***lite software*** version.

2.1.3 DLLs

When the customized programming sequence is not covering customer's requirements, then an attached to software package DLLs can be used. DLLs allows to fully control programming adapter from external software written in MS Visual C++, MS Visual Basic, LABView, DOS or other programming packages like Borland C++ etc. See ***"Fast MSP430 Flash Programmer - Remote Control Programming User's Guide"*** for details.

2.1.4 Self Test Program

Software package contains the Self Test program, that allows to check the adapter, target device and connection functionality. The Self Test program uses the API-DLL for communication between Self Test Program and hardware. See ***Appendix B*** in this manual for details.

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 feets 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 ^{TM} ME, WinNT, 2000, XP (32 or 64b), Vista (32 or 64b), Win-7 (32 or 64b). 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

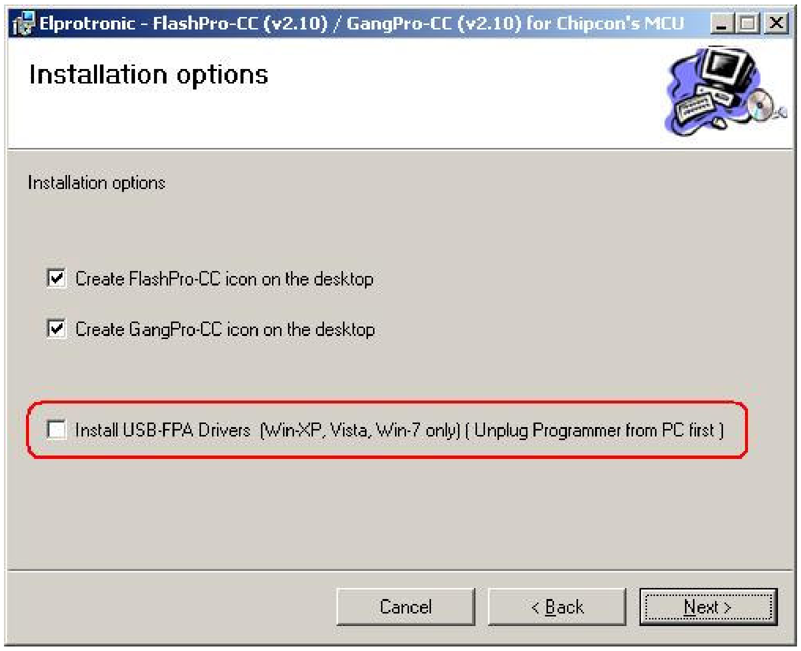
Software installation program is placing the USB driver files in the windows directories

Windows\inf

and

Windows\system32\drivers

that simplified driver installation. When the software package mentioned above starts, the following screen will pop-up (see figure 3.1-1).


Elprotronic - FlashPro-CC (v2.10) / GangPro-CC (v2.10) for Chipcon's MCU

Installation options

Installation options

☑ Create FlashPro-CC icon on the desktop

☑ Create GangPro-CC icon on the desktop

☐ Install USB-FPA Drivers (Win-XP, Vista, Win-7 only) (Unplug Programmer from PC first)

Figure 3.1-1

Select - InstallUSB-FPA Driver..... option if the driver was not installed before. When the installation software is finished, then the USB driver is preinstalled. Driver preinstallation can also be executed by running the ***USB-FPA-DriverInstaller.exe*** software located in directory

C:\Program Files\Elprotronic\Drivers USB-FPA\XP,Vista,Win-7

When the driver preinstallation is done then plug-in USB-FPA adapter to USB- PC port. The driver installation should be done automatically in Win-7 64, and in the Win 32 OS (XP, Vista, Win7 32b) the wizard should start. Follow instruction in wizad by pressing ***"recomended"*** option buttons and all should be done.

See ***USB-FPA Driver Installation.pdf*** document located in

C:\Program Files\Elprotronic\Drivers USB-FPAdirectory if you find any problems.

3.2 Hardware Setup

FlashPro430 software supports two adapter types - ***FlashPro430*** (USB-MSP430-FPA versions 1.0, 1.1, 1.2 and 2.0, USB-FPA 4.0, 4.1, 4.2 and 4.3) and ***GangPro430*** (USB-MSP430-FPA-3.0 and USB-FPA-5.0) (see figure 3.2-1). When the ***GangPro430*** adapter is detected by the FlashPro430 software then the only lines used by the first target device are activated in the FPA adapter. Lines TDI/TDO-2 to TDI/TDO-6 becomes tri-stated.

The ***FlashPro430*** adapters supports all interfaces

1. JTAG (4Mb/s, 1Mb/s and 400 kb/s),
2. Spy-Bi-Wire (fast and slow),

   (**Note:** The FPA - 1.0, 1.1 and 1.2 supports the **Spy-Bi-Wire** without fuse blown only)
3. Fast BSL (350 kb/s and 90 kb/s) and BSL (9.6 kb/s),

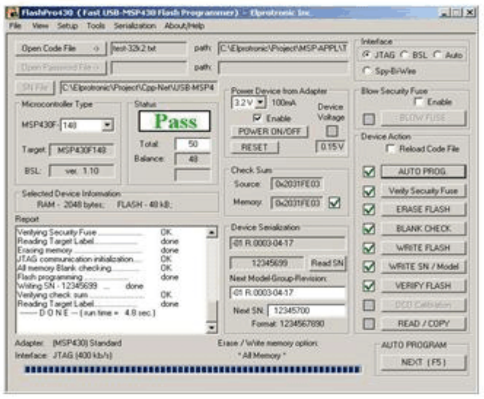
while ***GangPro430*** adapters supports only JTAG and BSL intefaces when the FlashPro430 software is used.

1. JTAG (1Mb/s and 400 kb/s),
2. Spy-Bi-Wire (fast and slow),

Figure 3.2-1 show available connection with target device using ***FlashPro430*** or ***GangPro430*** adapters.

1. Connect 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 USB-MSP430 Flash Programming Adapter to 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.

FlashPro430 software:
Supported adapters:
FlashPro430
(USB-FPA 1.x, 2.0, 4.x)
and
GangPro430
(USB-FPA 3.0, 5.x)
(one target device only)


FlashPro430 (Fast USB-MSP430 Flash Programmer) - Elprotronic Inc.

```
Open Code File   test-32k2.txt   path: C:\Elprotronic\Project\MSP-APPL\T
SN File   C:\Elprotronic\Project\Cpp-Net\USB-MSP4
Interface: (•) JTAG  ( ) BSL  ( ) Auto  ( ) Spy-Bi-Wire
Microcontroller Type: MSP430F-149
Target: MSP430F149
BSL: ver. 1.10
Status: Pass   Total: 50   Balance: 48
Power Device from Adapter: 3.2V  100mA  [✓] Enable   Device Voltage: 0.15V
Blow Security Fuse: [ ] Enable
Device Action: [ ] Reload Code File
Check Sum: Source: 0x2031FE03   Memory: 0x2031FE03 [✓]
Selected Device Information: RAM - 2048 bytes;  FLASH - 48 kB;
Report:
Verifying Security Fuse ........ OK
Reading Target Label ........... done
Erasing memory ................. done
JTAG communication initialization OK
All memory Blank checking ...... OK
Flash programming .............. done
Writing SN - 12345699 ... done
Verifying check sum ............ OK
Reading Target Label ........... done
----- D O N E -- (run time =  4.8 sec.)
Device Serialization: -01 R.0003-04-17   12345699  Read SN
Next Model-Group-Revision: -01 R.0003-04-17
Next SN: 12345700   Format: 1234567890
[✓] AUTO PROG  [✓] Verify Security Fuse  [✓] ERASE FLASH  [✓] BLANK CHECK
[✓] WRITE FLASH  [✓] WRITE SN / Model  [✓] VERIFY FLASH  [ ] DCO Calibration  [ ] READ / COPY
Adapter: (MSP430) Standard
Interface: JTAG (400 kb/s)
Erase / Write memory option: * All Memory *
AUTO PROGRAM: NEXT (F5)
```

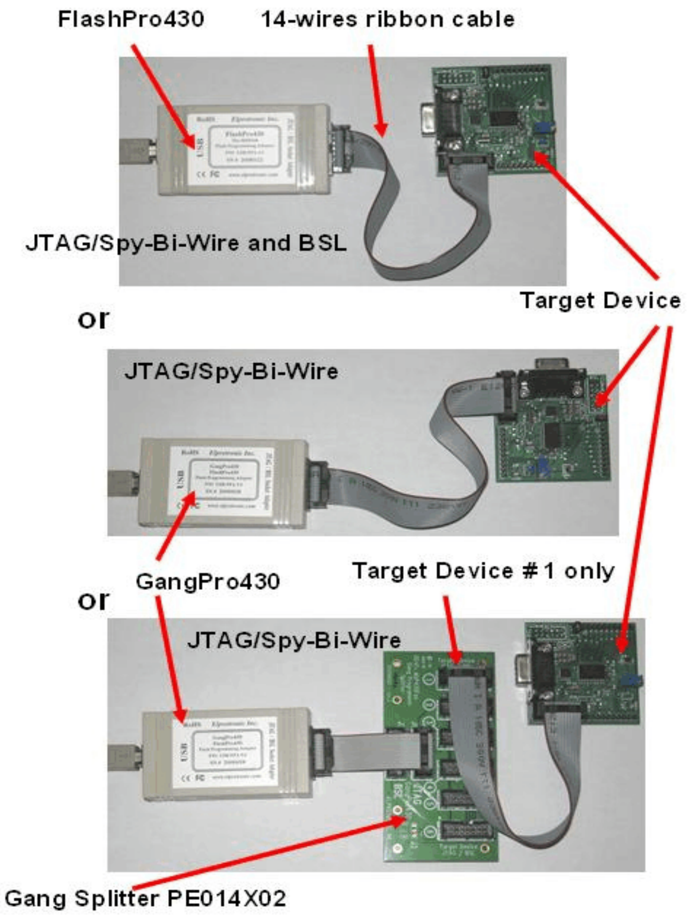


Figure 3.2-1

3.3 Starting up "FlashPro430" Flash Programmer

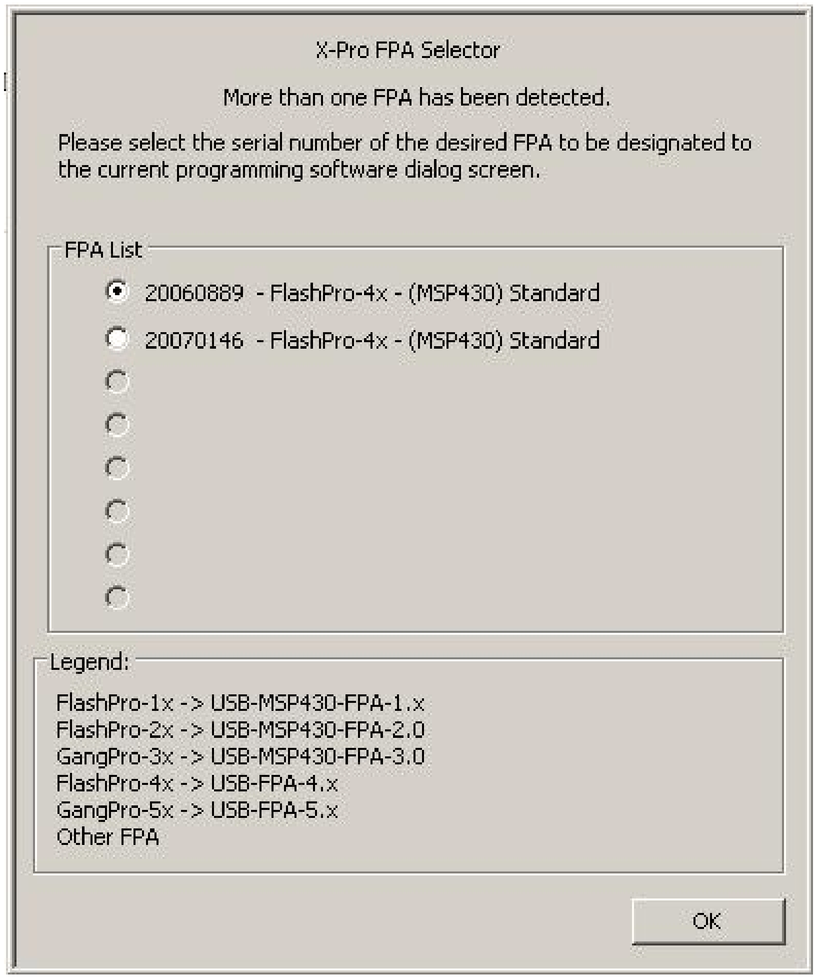
To start the ***FlashPro430*** Flash Programmer click on the Elprotronic FlashPro430 icon.

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.

Figure 3.3-1

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.)



X-Pro FPA Selector

More than one FPA has been detected.

Please select the serial number of the desired FPA to be designated to the current programming software dialog screen.

FPA List

- [✓ 20060889 - FlashPro-4x - (MSP430) Standard]
- 20070146 - FlashPro-4x - (MSP430) Standard

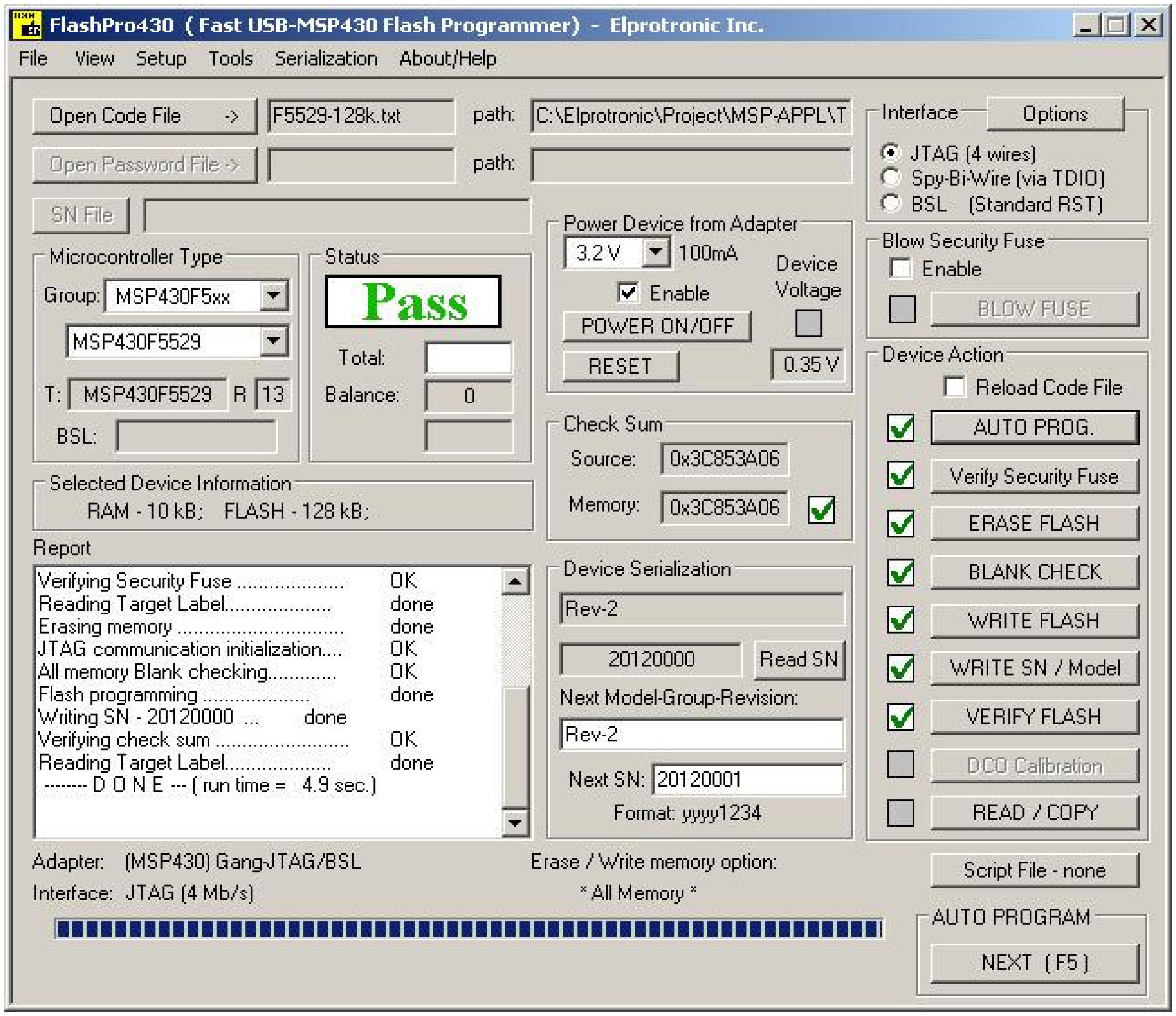
Legend:

FlashPro-1x -> USB-MSP430-FPA-1.x
FlashPro-2x -> USB-MSP430-FPA-2.0
GangPro-3x -> USB-MSP430-FPA-3.0
FlashPro-4x -> USB-FPA-4.x
GangPro-5x -> USB-FPA-5.x
Other FPA

Figure 3.4-1

When more then 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*** (models USB-MSP430-FPA-1.x, USB-MSP430-FPA-2.0 or USB-FPA-4.x) the target device can be connected to FPA via JTAG/SBW or BSL interface, while using the ***GangPro430 FPA*** (model USB-MSP430-FPA-3.0 or USB-FPA-5.x) the target device can be accessible via JTAG/SBW interface only.

4. Programming Dialogue Screen



Open Code File: F5529-128k.txt path: C:\Elprotronic\Project\MSP-APPL\T

Interface: JTAG (4 wires) ● / Spy-Bi-Wire (via TDIO) / BSL (Standard RST)

Power Device from Adapter: 3.2 V, 100mA, ✓ Enable; Device Voltage 0.35 V

Blow Security Fuse: Enable ☐

Microcontroller Type: Group: MSP430F5xx; MSP430F5529; T: MSP430F5529 R 13; BSL:

Status: Pass; Total: ; Balance: 0

Check Sum: Source: 0x3C853A06; Memory: 0x3C853A06 ✓

Device Action: ☐ Reload Code File

Selected Device Information: RAM - 10 kB; FLASH - 128 kB;

Report

```
Verifying Security Fuse ...................   OK
Reading Target Label...................       done
Erasing memory .............................  done
JTAG communication initialization....        OK
All memory Blank checking............        OK
Flash programming ...................        done
Writing SN - 20120000 ...      done
Verifying check sum ........................ OK
Reading Target Label...................       done
 -------- D O N E --- ( run time =   4.9 sec.)
```

Device Serialization: Rev-2; 20120000; Next Model-Group-Revision: Rev-2; Next SN: 20120001; Format: yyyy1234

Adapter: (MSP430) Gang-JTAG/BSL
Interface: JTAG (4 Mb/s)
Erase / Write memory option: * All Memory *

Figure 4-1. Programming dialogue 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***, ***Spy-Bi-Wire (SBW)***, ***BSL*** or ***Auto*** can be selected in the Interface group. When the programming adapter has implemented all types of communication interface, then all buttons in the interface group are active, otherwise only one selecting button is active - JTAG/Spy-Bi-Wire or BSL. Proper communication interface should be selected, otherwise communication with the target device can fail.

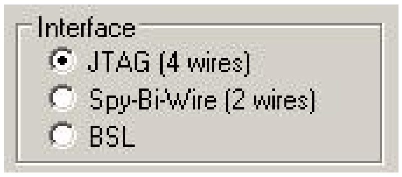

Interface
[✓ JTAG (4 wires)]
Spy-Bi-Wire (2 wires)
BSL

Figure 4.1

It is recommended to prepare in the target device board one 14-pin JTAG connector with JTAG/SBW and BSL connections (see chapter 11 for details). When the JTAG/SBW and BSL 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/SBW communication port. When the security fuse is blown, then JTAG/SBW interfaces are 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.
Note: Auto option can use only JTAG or BSL interface.

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.

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.

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:**'

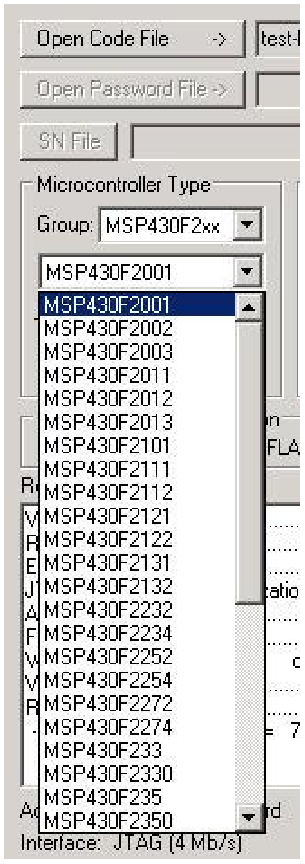


Figure 4.2-1

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/SBW or BSL communication. Particular type of the microcontroller in the same group ID can be detected when communication via JTAG/SBW 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 al long as size of FLASH is available.

In the example the four groups 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
MSP430F1121A	**256**	**4 k**

[1132]   F11x2

Name	RAM size [bytes]	ROM size [kbytes]
MSP430F1122	256	4 k
MSP430F1132	**256**	**8 k**

[F123]   F122..F123

Name	RAM size [bytes]	ROM size [kbytes]
MSP430F122	256	4 k
MSP430F123	**256**	**8 k**

[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
MSP430F149	**2048**	**60 k**
MSP430F1491	2048	60 k

When ***'Any'*** microcontroller type is selected then only name of the microcontroller's group like ***F13x..F14x*** 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 ***F13x..F14x*** (see table above). If the 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 ***MSP430F149*** instead the group name ***F13x..F14x*** 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 ***F13x..F14x***.

4.3 Code File Management

FlashPro430 flash programmer provides a few 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. Following code formats are supplied - Texas Instruments *.txt, TI's Code Composer Essentials *.out, Intel *.hex, Motorola *.s19, *.s28, *.s37, IAR UBROF-9 *.d43 and IAR debug (Intel or Motorola) *.a43 formats. When the TI's CCE file is used then the path for the TI's hex430.exe file should be specified. See Preferences dialog for details.

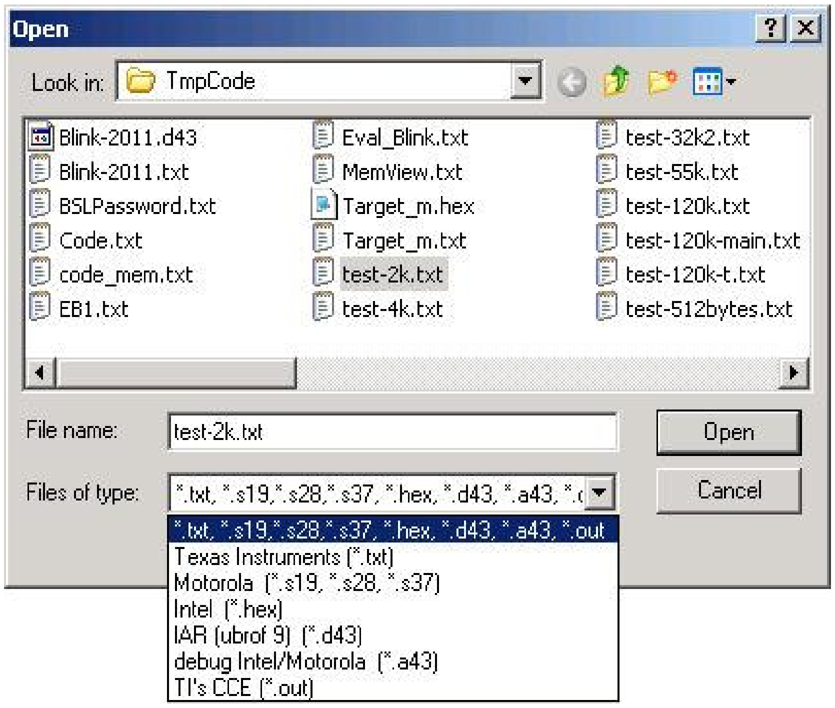
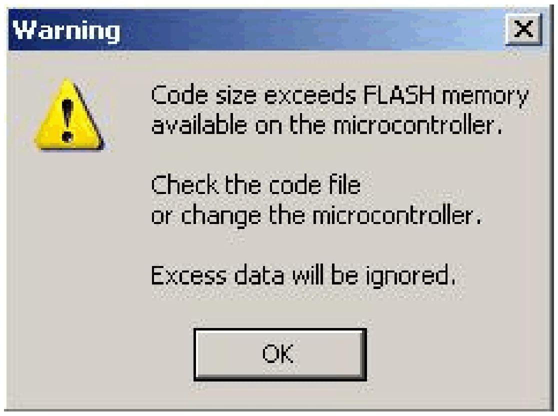


Figure 4.3-1

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.



Code size exceeds FLASH memory available on the microcontroller.

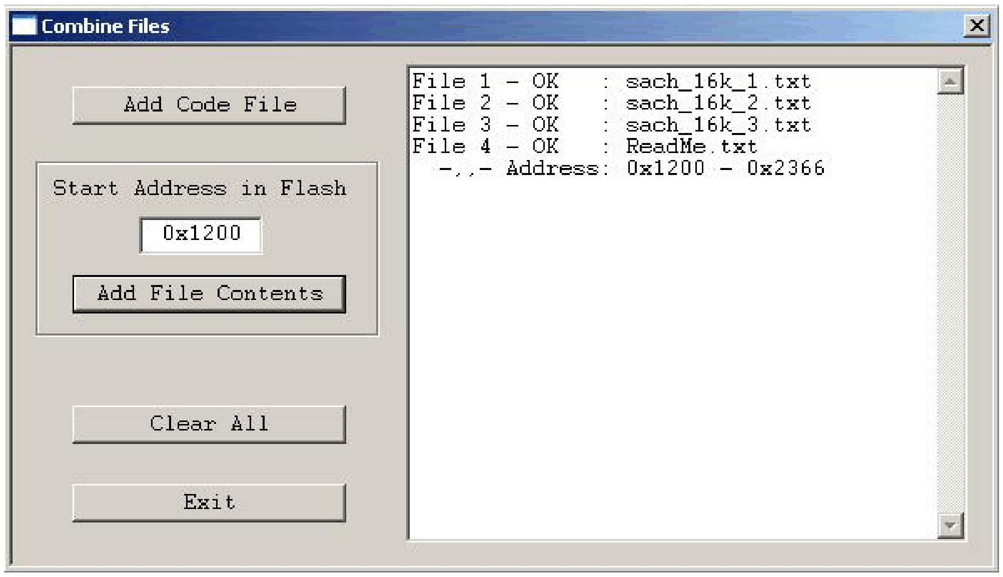
Check the code file or change the microcontroller.

Excess data will be ignored.

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). Check sum calculated from the code file will be displayed in the ***"Check sum - Source"*** window. 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



0x1200

```
File 1 - OK    : sach_16k_1.txt
File 2 - OK    : sach_16k_2.txt
File 3 - OK    : sach_16k_3.txt
File 4 - OK    : ReadMe.txt
  -.,- Address: 0x1200 - 0x2366
```

Figure 4.3-3

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.

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.

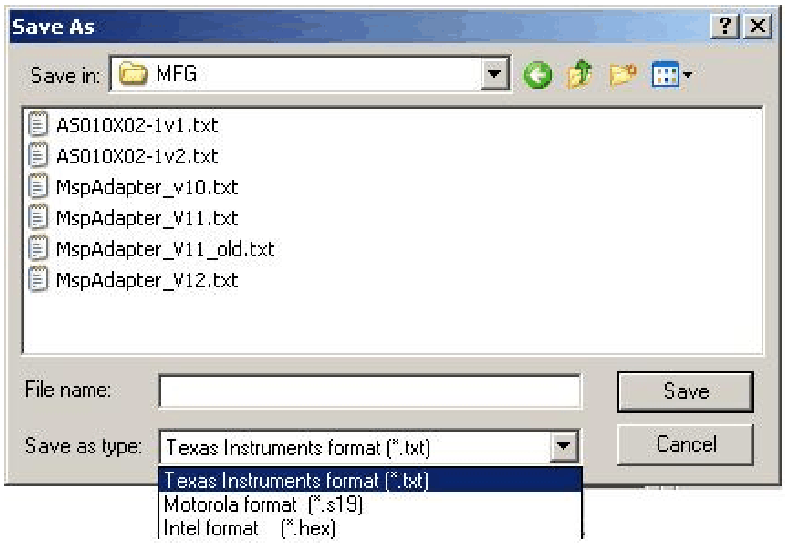


Figure 4.3-4

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/SBW 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/SBW interface becomes unusable.

When JTAG/SBW 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).

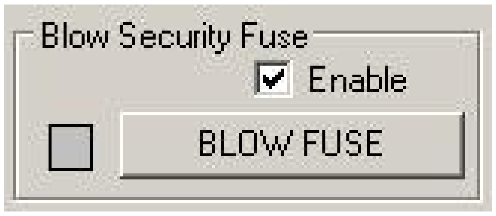
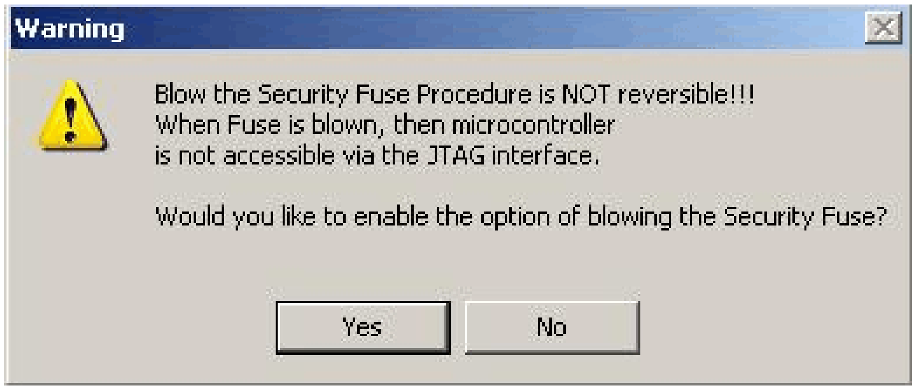


Figure 4.4-1

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



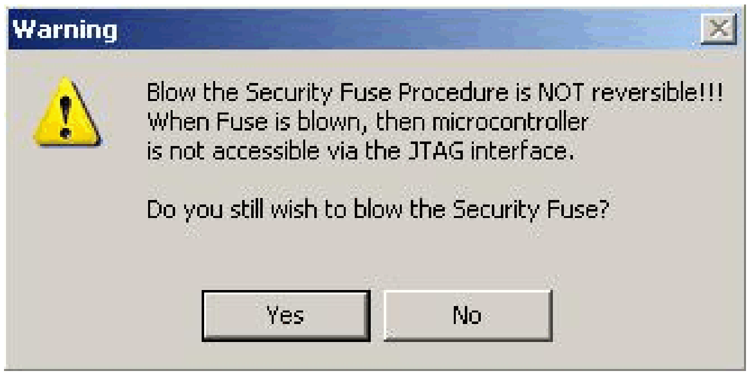
Blow the Security Fuse Procedure is NOT reversible!!!
When Fuse is blown, then microcontroller
is not accessible via the JTAG interface.

Would you like to enable the option of blowing the Security Fuse?

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.

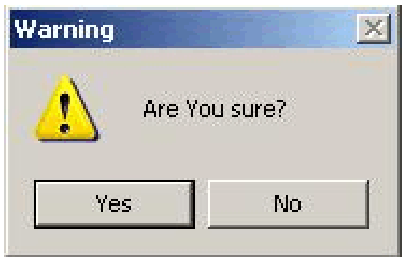
When ***'BLOW FUSE'*** button is pressed, then two following warnings are displayed, before fuse will be blown.


Warning

Blow the Security Fuse Procedure is NOT reversible!!!
When Fuse is blown, then microcontroller
is not accessible via the JTAG interface.

Do you still wish to blow the Security Fuse?

Figure 4.4-3


Warning

Are You sure?

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/SBW   interface becomes inoperable.

In the newer MSP430 MCU family - MSP430F5xx, MSP430F6xx, CC430F5xx, CC430F6xx the security fuse is located in the locked Flash memory and it is possible to unlock the security fuse. The restore process can be provided using the BSL access, erasing the whole memory flash contents first. The FlashPro430 programmer has an option to restore the security fuse via BSL interface. Option can be selected from the pull down menu ***Tools->Restore JTAG Security Fuse***.

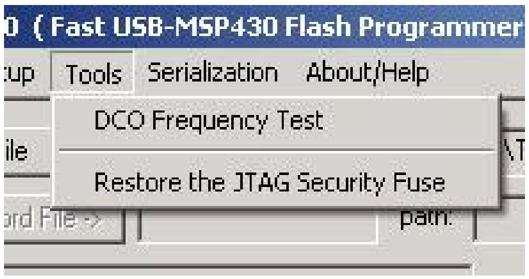
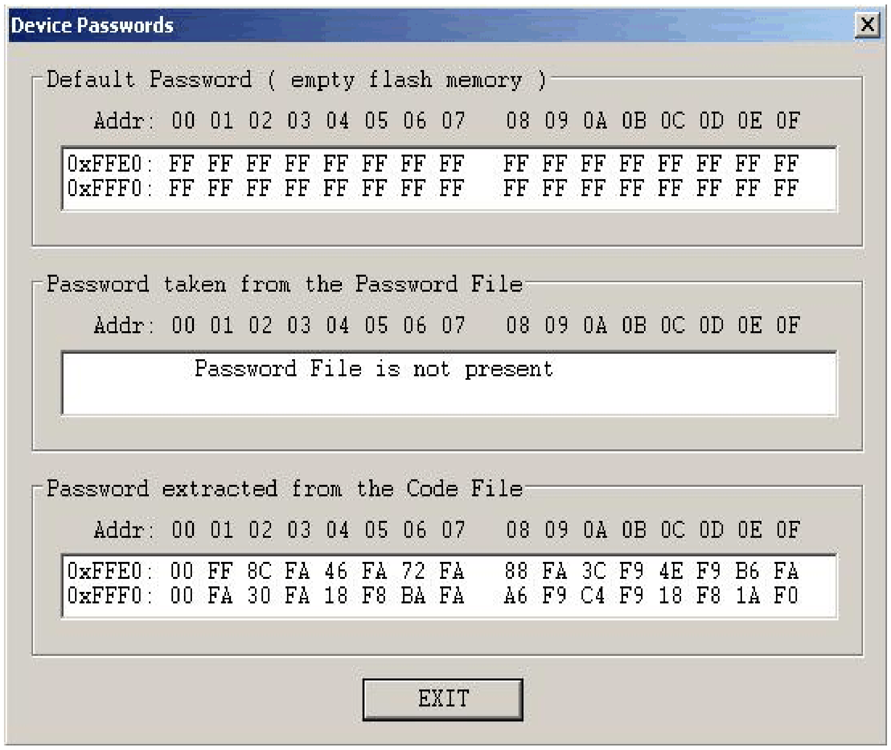


Figure 4.4-5

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 0xFFE0-0xFFFF and 16 bytes of code at location 0xFFF0-0xFFFF for the F543xx MCU. 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 (Figure 4.4-6). 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.


Device Passwords
Default Password (empty flash memory)
Password taken from the Password File
Password extracted from the Code File
EXIT

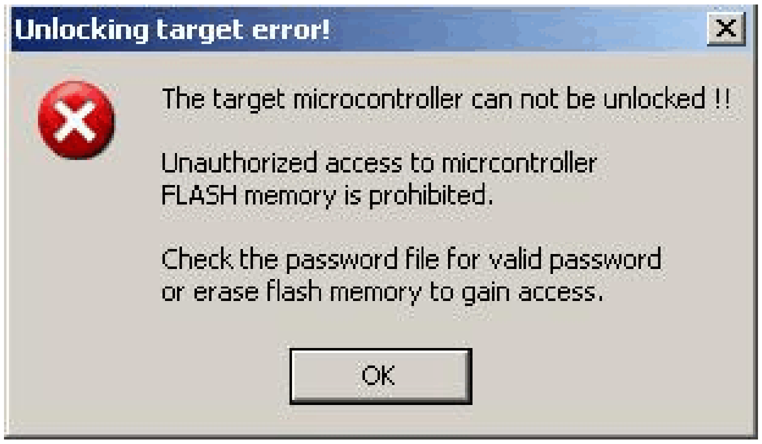
```
Addr: 00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F
0xFFE0: FF FF FF FF FF FF FF FF   FF FF FF FF FF FF FF FF
0xFFF0: FF FF FF FF FF FF FF FF   FF FF FF FF FF FF FF FF

Addr: 00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F
          Password File is not present

Addr: 00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F
0xFFE0: 00 FF 8C FA 46 FA 72 FA   88 FA 3C F9 4E F9 B6 FA
0xFFF0: 00 FA 30 FA 18 F8 BA FA   A6 F9 C4 F9 18 F8 1A F0
```

Figure 4.4-6

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 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.4.4-7).

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


Unlocking target error!

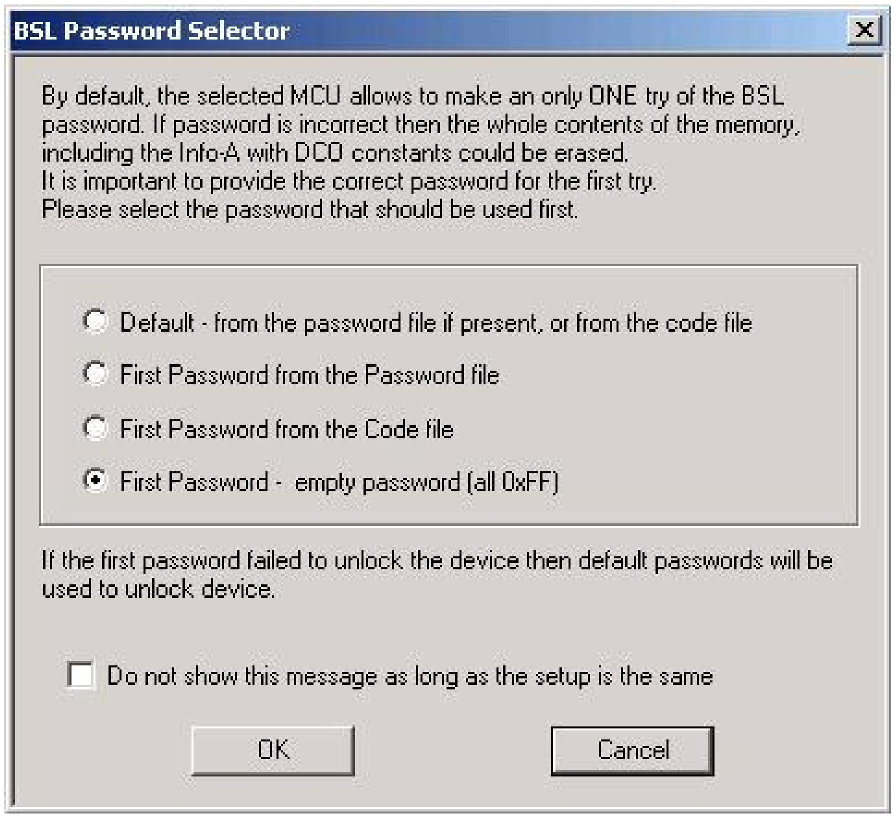
The target microcontroller can not be unlocked !!

Unauthorized access to micrcontroller
FLASH memory is prohibited.

Check the password file for valid password
or erase flash memory to gain access.

Figure 4.4-7

When the F2xx or F4xx with BSL firmware version2.0 and higher are used then by default the firmware allows to verify the BSL password only once. If the BSL password is wrong then the whole contents of flash memory, including protected Info-A segment (with DCO calibration data) could be erased. When that kind of microcontroller is selected then the following dialog screen (Figure 4.4-8) is displayed that allows to select desired BSL password to be used first.



By default, the selected MCU allows to make an only ONE try of the BSL password. If password is incorrect then the whole contents of the memory, including the Info-A with DCO constants could be erased.
It is important to provide the correct password for the first try.
Please select the password that should be used first.

- ○ Default - from the password file if present, or from the code file
- ○ First Password from the Password file
- ○ First Password from the Code file
- ● First Password - empty password (all 0xFF)

If the first password failed to unlock the device then default passwords will be used to unlock device.

☐ Do not show this message as long as the setup is the same

Figure 4.4-8

If the first password failed to unlock the device then the default passwords will be used to unlock the device. If that happened, then of course some application cannot use the MSP if the application is using the DCO calibration content saved in the Info-A segment. The DCO calibration data should be restored first. If the DCO calibration data are erased then the FlashPro430 software can recalibrate the DCO constants using access via JTAG, Spy-Bi-Wire or Fast BSL. No extra hardware is required. The DCO are calibrated with tolerance +/- 1% for desired frequency and the DCO constants are saved in the flash memory. See chapter 6 for details how to make the setup for the DCO recalibration.

The BSL firmware version 2.0 and higher allows to disable the feature that the memory is erased when the password is incorrect and retry the BSL password again. If this option is used then the Info-A segment will not be erased and the DCO constants would be saved. See the ***BSL Password and Access*** - chapter 9 for details.

In the F543x MCU (first few types of the F5xx only) the new BSL firmware is implemented with shorter password (16 bytes only) and with fixed option that allows to try the password only once. If password is wrong then the whole flash memory is erased including protected Info segments. It should be noticed that in the F5xx all calibration data are not located in the regular flash location (info or main memory segments) and all calibration data will not be erased.

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.

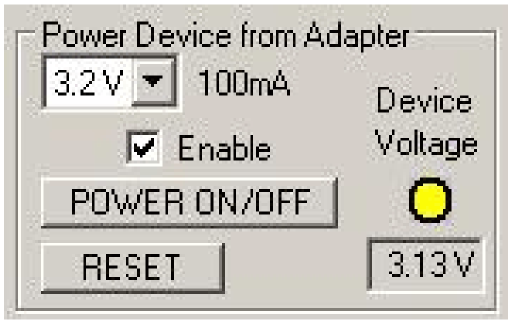
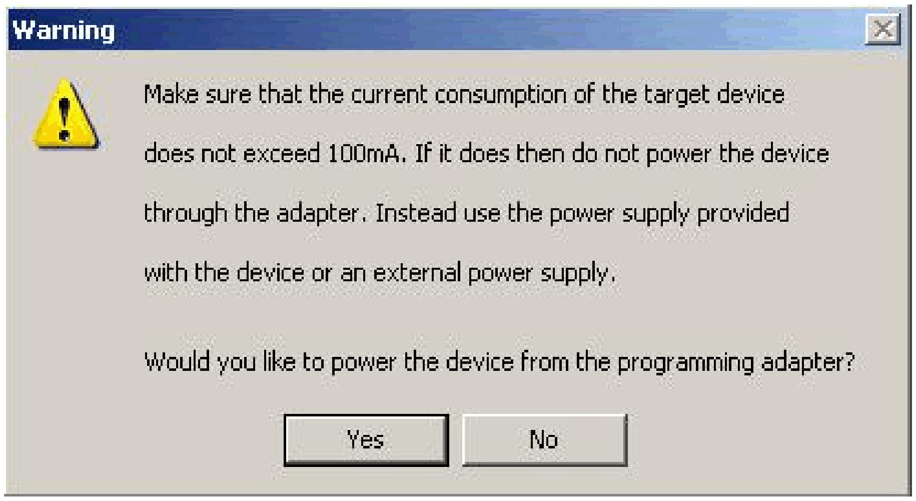


Figure 4.5-1


Warning

Make sure that the current consumption of the target device does not exceed 100mA. If it does then do not power the device through the adapter. Instead use the power supply provided with the device or an external power supply.

Would you like to power the device from the programming adapter?

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

Device Action box contains 9 buttons (see Figure 4.6-1 and 4.6-2) and 9 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 then 2.7V, the communication with the target device is initiated via JTAG/SBW or BSL Interface. When the JTAG/SBW Interface is selected then the security fuse

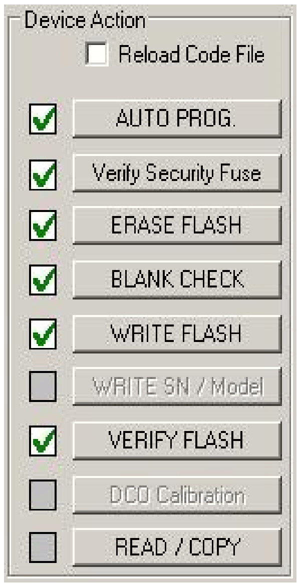


Figure 4.6-1
JTAG/SBW Interface selected

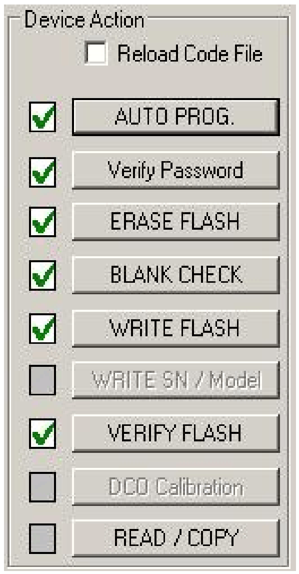
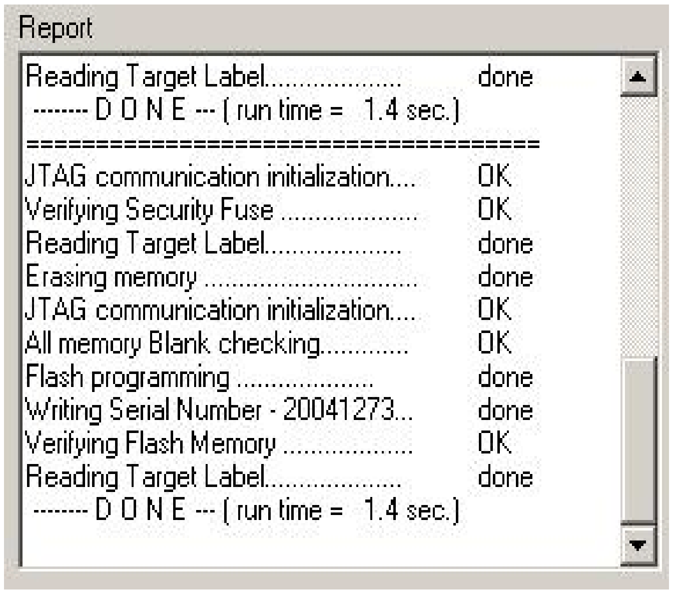


Figure 4.6-2
BSL Interface selected.

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.



```
Reading Target Label..................         done
 ------- D O N E --- ( run time =   1.4 sec.)
=====================================
JTAG communication initialization....         OK
Verifying Security Fuse ...................   OK
Reading Target Label..................         done
Erasing memory .............................  done
JTAG communication initialization....         OK
All memory Blank checking............          OK
Flash programming ...................          done
Writing Serial Number - 20041273...           done
Verifying Flash Memory ..................     OK
Reading Target Label..................         done
 ------- D O N E --- ( run time =   1.4 sec.)
```

Figure 4.6-3

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)
- calculate defined Check Sum (or CRC) if selected,
- erase flash memory,
- confirm if memory has been erase,
- flash programming and verification,
- labelling information generation,
- flash memory check sum verification,
- retrieve labelling information,
- calibrating DCO frequency (if enabled),
- 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. When the serialization from file option is used , then in the bottom line is displayed the number of the serial numbers left in the list.

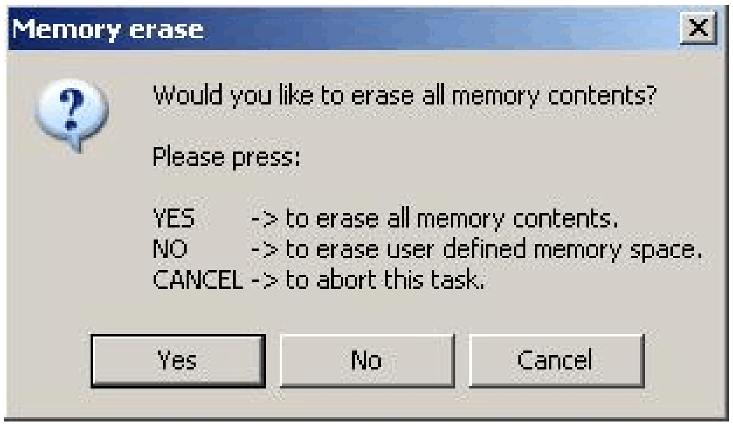
Note: *Balance counter works only with Auto Program procedure.*

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 then ***'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:


Memory erase

Would you like to erase all memory contents?

Please press:

```
YES    -> to erase all memory contents.
NO     -> to erase user defined memory space.
CANCEL -> to abort this task.
```

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:

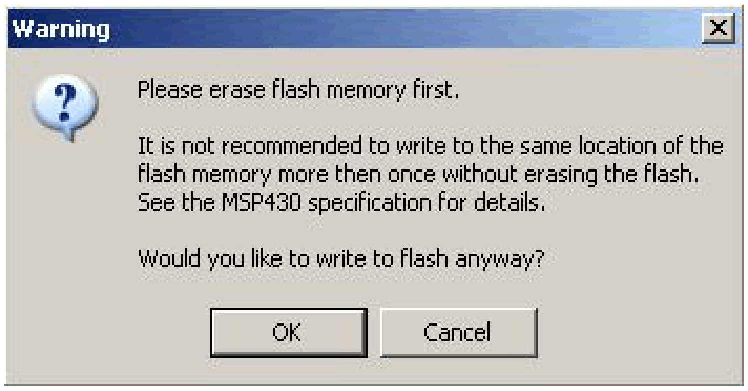
✓	- all memory is blank
✗✓	- all memory is not blank, but selected part of it is.
✗	- memory is not blank.

4.6.5 *Write Flash button*

When write flash button is clicked, then contents from the code file with defined Check Sum (if selected) will be written to the flash memory.

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

When the second time target device is programmed, then the following warning message is displayed



Warning

Please erase flash memory first.

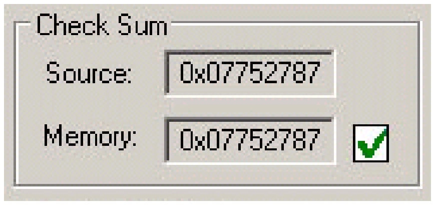
It is not recommended to write to the same location of the flash memory more then once without erasing the flash. See the MSP430 specification for details.

Would you like to write to flash anyway?

Figure 4.6.5-1

4.6.6 *Verify Flash button*

The Verify Flash function compares the contents of the flash memory with data from the code



Check Sum

Source: 0x07752787

Memory: 0x07752787 ✓

Figure 4.6.6-1

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***).

If the JTAG/SBW 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.4.6.6-1), and check sum calculated from the target microcontroller flash memory data is displayed in the ***Memory*** line of this group. If the JTAG/SBW 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.*

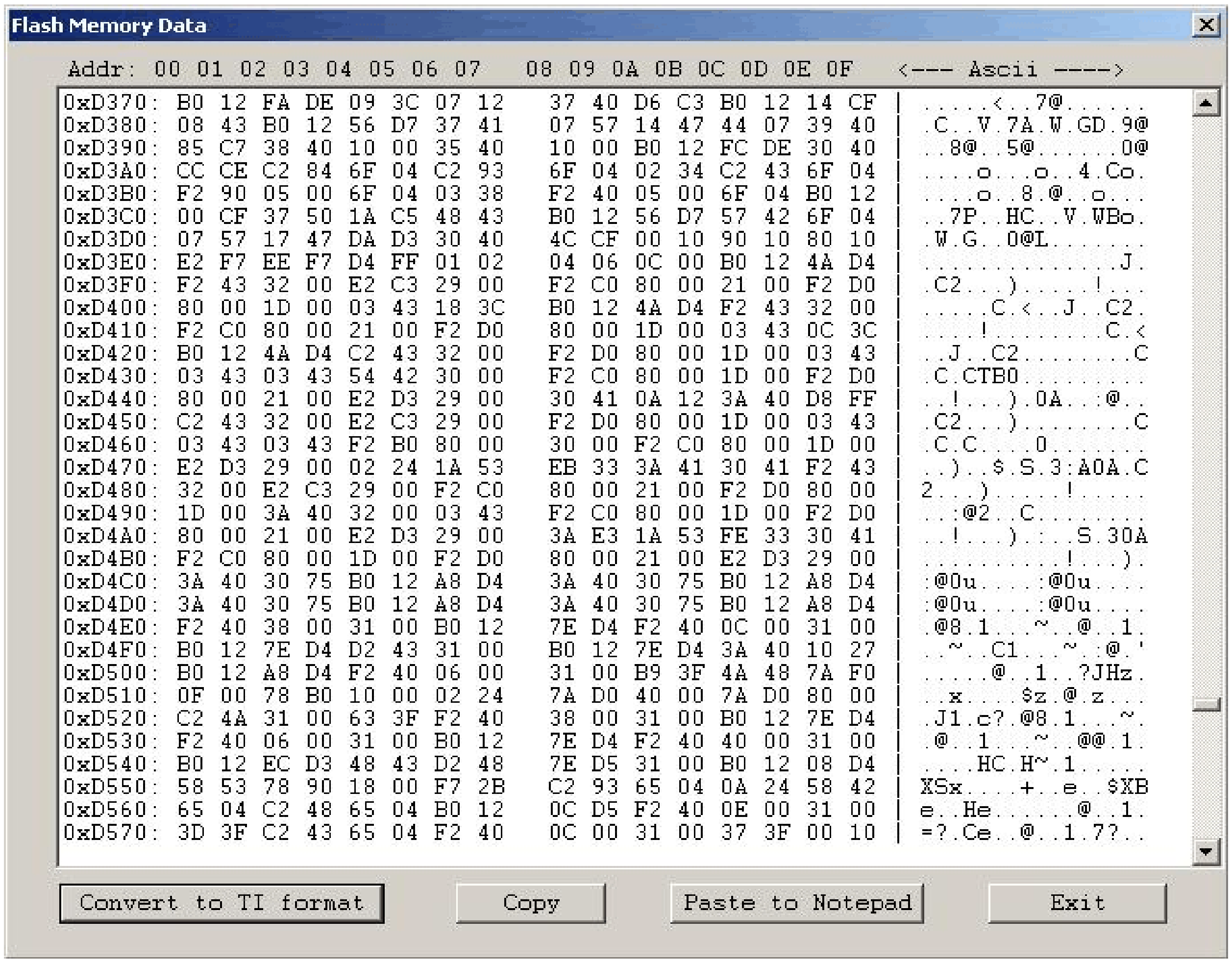
4.6.7 Read/ Copy Flash button

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.

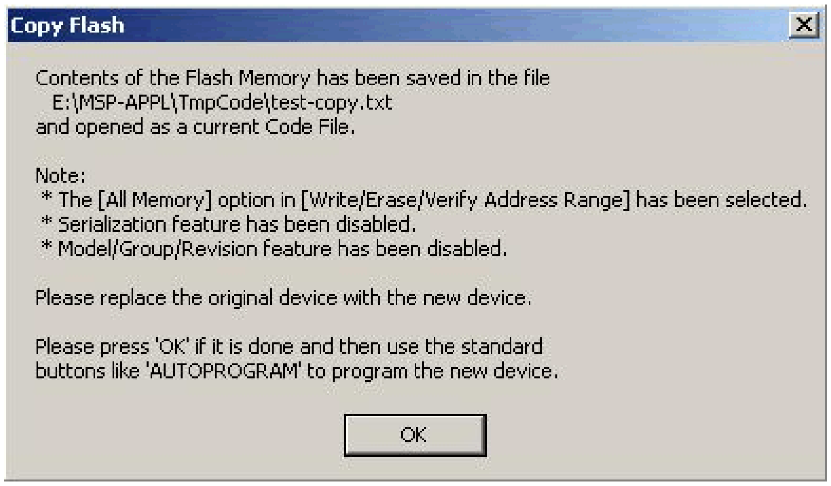
When the ***'Copy'*** button is clicked, then the contents of the read target device memory will be saved in the specified by user file name and opened as a current Code File. Also programmer setup will be modified for the copy procedure. Especially the serialization will be disabled and the ***'All Memory'*** option will be selected in the ***'Write/Erase/Verify Address Range***'. Following message shown on figure 4.6.7-2 is displayed.

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



```
Addr: 00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F   <--- Ascii ---->
0xD370: B0 12 FA DE 09 3C 07 12   37 40 D6 C3 B0 12 14 CF | .....<..7@......
0xD380: 08 43 B0 12 56 D7 37 41   07 57 14 47 44 07 39 40 | .C..V.7A.W.GD.9@
0xD390: 85 C7 38 40 10 00 35 40   10 00 B0 12 FC DE 30 40 | ..8@..5@......0@
0xD3A0: CC CE C2 84 6F 04 C2 93   6F 04 02 34 C2 43 6F 04 | ....o...o..4.Co.
0xD3B0: F2 90 05 00 6F 04 03 38   F2 40 05 00 6F 04 B0 12 | ....o..8.@..o...
0xD3C0: 00 CF 37 50 1A C5 48 43   B0 12 56 D7 57 42 6F 04 | ..7P..HC..V.WBo.
0xD3D0: 07 57 17 47 DA D3 30 40   4C CF 00 10 90 10 80 10 | .W.G..0@L.......
0xD3E0: E2 F7 EE F7 D4 FF 01 02   04 06 0C 00 B0 12 4A D4 | ..............J.
0xD3F0: F2 43 32 00 E2 C3 29 00   F2 C0 80 00 21 00 F2 D0 | .C2...).....!...
0xD400: 80 00 1D 00 03 43 18 3C   B0 12 4A D4 F2 43 32 00 | .....C.<..J..C2.
0xD410: F2 C0 80 00 21 00 F2 D0   80 00 1D 00 03 43 0C 3C | ....!........C.<
0xD420: B0 12 4A D4 C2 43 32 00   F2 D0 80 00 1D 00 03 43 | ..J..C2........C
0xD430: 03 43 03 43 54 42 30 00   F2 C0 80 00 1D 00 F2 D0 | .C.CTB0.........
0xD440: 80 00 21 00 E2 D3 29 00   30 41 0A 12 3A 40 D8 FF | ..!...).0A..:@..
0xD450: C2 43 32 00 E2 C3 29 00   F2 D0 80 00 1D 00 03 43 | .C2...)........C
0xD460: 03 43 03 43 F2 B0 80 00   30 00 F2 C0 80 00 1D 00 | .C.C....0.......
0xD470: E2 D3 29 00 02 24 1A 53   EB 33 3A 41 30 41 F2 43 | ..)..$.S.3:A0A.C
0xD480: 32 00 E2 C3 29 00 F2 C0   80 00 21 00 F2 D0 80 00 | 2...).....!.....
0xD490: 1D 00 3A 40 32 00 03 43   F2 C0 80 00 1D 00 F2 D0 | ..:@2..C........
0xD4A0: 80 00 21 00 E2 D3 29 00   3A E3 1A 53 FE 33 30 41 | ..!...).:..S.30A
0xD4B0: F2 C0 80 00 1D 00 F2 D0   80 00 21 00 E2 D3 29 00 | ..........!...).
0xD4C0: 3A 40 30 75 B0 12 A8 D4   3A 40 30 75 B0 12 A8 D4 | :@0u....:@0u....
0xD4D0: 3A 40 30 75 B0 12 A8 D4   3A 40 30 75 B0 12 A8 D4 | :@0u....:@0u....
0xD4E0: F2 40 38 00 31 00 B0 12   7E D4 F2 40 0C 00 31 00 | .@8.1...~..@..1.
0xD4F0: B0 12 7E D4 D2 43 31 00   B0 12 7E D4 3A 40 10 27 | ..~..C1...~.:@.'
0xD500: B0 12 A8 D4 F2 40 06 00   31 00 B9 3F 4A 48 7A F0 | .....@..1..?JHz.
0xD510: 0F 00 78 B0 10 00 02 24   7A D0 40 00 7A D0 80 00 | ..x....$z.@.z...
0xD520: C2 4A 31 00 63 3F F2 40   38 00 31 00 B0 12 7E D4 | .J1.c?.@8.1...~.
0xD530: F2 40 06 00 31 00 B0 12   7E D4 F2 40 40 00 31 00 | .@..1...~..@@.1.
0xD540: B0 12 EC D3 48 43 D2 48   7E D5 31 00 B0 12 08 D4 | ....HC.H~.1.....
0xD550: 58 53 78 90 18 00 F7 2B   C2 93 65 04 0A 24 58 42 | XSx....+..e..$XB
0xD560: 65 04 C2 48 65 04 B0 12   0C D5 F2 40 0E 00 31 00 | e..He......@..1.
0xD570: 3D 3F C2 43 65 04 F2 40   0C 00 31 00 37 3F 00 10 | =?.Ce..@..1.7?..
```

Figure 4.6.7-1



Contents of the Flash Memory has been saved in the file
E:\MSP-APPL\TmpCode\test-copy.txt
and opened as a current Code File.

Note:
* The [All Memory] option in [Write/Erase/Verify Address Range] has been selected.
* Serialization feature has been disabled.
* Model/Group/Revision feature has been disabled.

Please replace the original device with the new device.

Please press 'OK' if it is done and then use the standard
buttons like 'AUTOPROGRAM' to program the new device.

Figure 4.6.7-2

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.

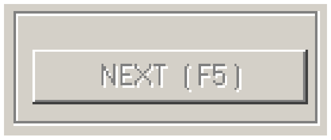


Figure 4.7-1

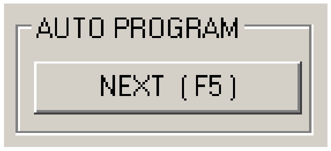


Figure 4.7-2

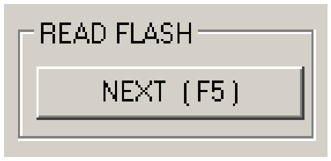
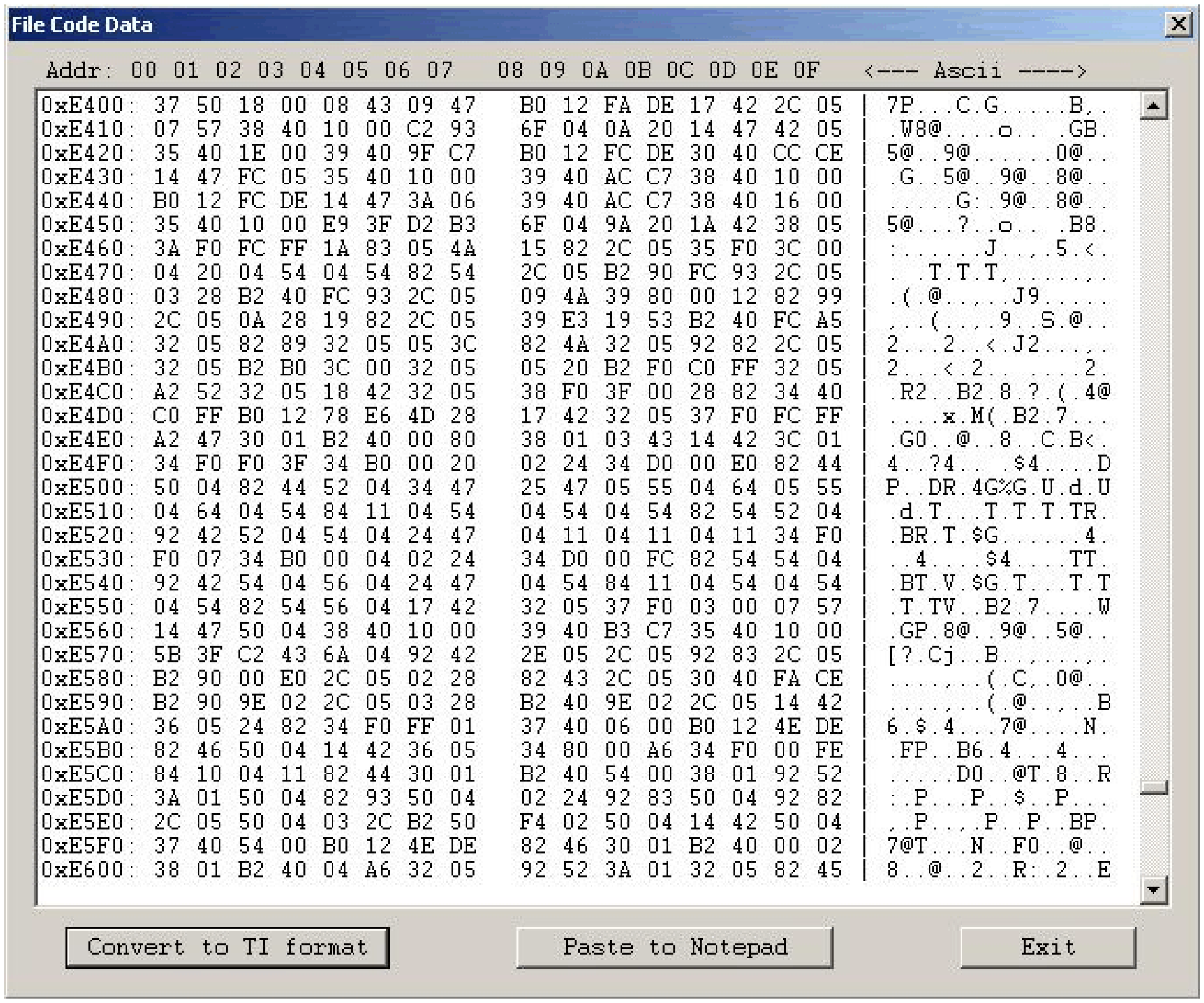


Figure 4.7-3

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 is displayed as double dots '..'(if data does not exist in the code file) . When code size exceeds Flash



```
Addr: 00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F   <--- Ascii ---->
0xE400: 37 50 18 00 08 43 09 47   B0 12 FA DE 17 42 2C 05 | 7P...C.G.....B,.
0xE410: 07 57 38 40 10 00 C2 93   6F 04 0A 20 14 47 42 05 | .W8@....o.. .GB.
0xE420: 35 40 1E 00 39 40 9F C7   B0 12 FC DE 30 40 CC CE | 5@..9@......0@..
0xE430: 14 47 FC 05 35 40 10 00   39 40 AC C7 38 40 10 00 | .G..5@..9@..8@..
0xE440: B0 12 FC DE 14 47 3A 06   39 40 AC C7 38 40 16 00 | .....G:.9@..8@..
0xE450: 35 40 10 00 E9 3F D2 B3   6F 04 9A 20 1A 42 38 05 | 5@...?..o.. .B8.
0xE460: 3A F0 FC FF 1A 83 05 4A   15 82 2C 05 35 F0 3C 00 | :......J..,.5.<.
0xE470: 04 20 04 54 04 54 82 54   2C 05 B2 90 FC 93 2C 05 | . .T.T.T,.....,.
0xE480: 03 28 B2 40 FC 93 2C 05   09 4A 39 80 00 12 82 99 | .(.@..,..J9.....
0xE490: 2C 05 0A 28 19 82 2C 05   39 E3 19 53 B2 40 FC A5 | ,..(..,.9..S.@..
0xE4A0: 32 05 82 89 32 05 05 3C   82 4A 32 05 92 82 2C 05 | 2...2..<.J2...,.
0xE4B0: 32 05 B2 B0 3C 00 32 05   05 20 B2 F0 C0 FF 32 05 | 2...<.2.. ....2.
0xE4C0: A2 52 32 05 18 42 32 05   38 F0 3F 00 28 82 34 40 | .R2..B2.8.?.(.4@
0xE4D0: C0 FF B0 12 78 E6 4D 28   17 42 32 05 37 F0 FC FF | ....x.M(.B2.7...
0xE4E0: A2 47 30 01 B2 40 00 80   38 01 03 43 14 42 3C 01 | .G0..@..8..C.B<.
0xE4F0: 34 F0 F0 3F 34 B0 00 20   02 24 34 D0 00 E0 82 44 | 4..?4.. .$4....D
0xE500: 50 04 82 44 52 04 34 47   25 47 05 55 04 64 05 55 | P..DR.4G%G.U.d.U
0xE510: 04 64 04 54 84 11 04 54   04 54 04 54 82 54 52 04 | .d.T...T.T.T.TR.
0xE520: 92 42 52 04 54 04 24 47   04 11 04 11 04 11 34 F0 | .BR.T.$G......4.
0xE530: F0 07 34 B0 00 04 02 24   34 D0 00 FC 82 54 54 04 | ..4....$4....TT.
0xE540: 92 42 54 04 56 04 24 47   04 54 84 11 04 54 04 54 | .BT.V.$G.T...T.T
0xE550: 04 54 82 54 56 04 17 42   32 05 37 F0 03 00 07 57 | .T.TV..B2.7....W
0xE560: 14 47 50 04 38 40 10 00   39 40 B3 C7 35 40 10 00 | .GP.8@..9@..5@..
0xE570: 5B 3F C2 43 6A 04 92 42   2E 05 2C 05 92 83 2C 05 | [?.Cj..B..,...,.
0xE580: B2 90 00 E0 2C 05 02 28   82 43 2C 05 30 40 FA CE | ....,..(.C,.0@..
0xE590: B2 90 9E 02 2C 05 03 28   B2 40 9E 02 2C 05 14 42 | ....,..(.@..,..B
0xE5A0: 36 05 24 82 34 F0 FF 01   37 40 06 00 B0 12 4E DE | 6.$.4...7@....N.
0xE5B0: 82 46 50 04 14 42 36 05   34 80 00 A6 34 F0 00 FE | .FP..B6.4...4...
0xE5C0: 84 10 04 11 82 44 30 01   B2 40 54 00 38 01 92 52 | .....D0..@T.8..R
0xE5D0: 3A 01 50 04 82 93 50 04   02 24 92 83 50 04 92 82 | :.P...P..$..P...
0xE5E0: 2C 05 50 04 03 2C B2 50   F4 02 50 04 14 42 50 04 | ,.P..,.P..P..BP.
0xE5F0: 37 40 54 00 B0 12 4E DE   82 46 30 01 B2 40 00 02 | 7@T...N..F0..@..
0xE600: 38 01 B2 40 04 A6 32 05   92 52 3A 01 32 05 82 45 | 8..@..2..R:.2..E
```

Figure 5-1

memory space of the selected microcontroller, then warning message

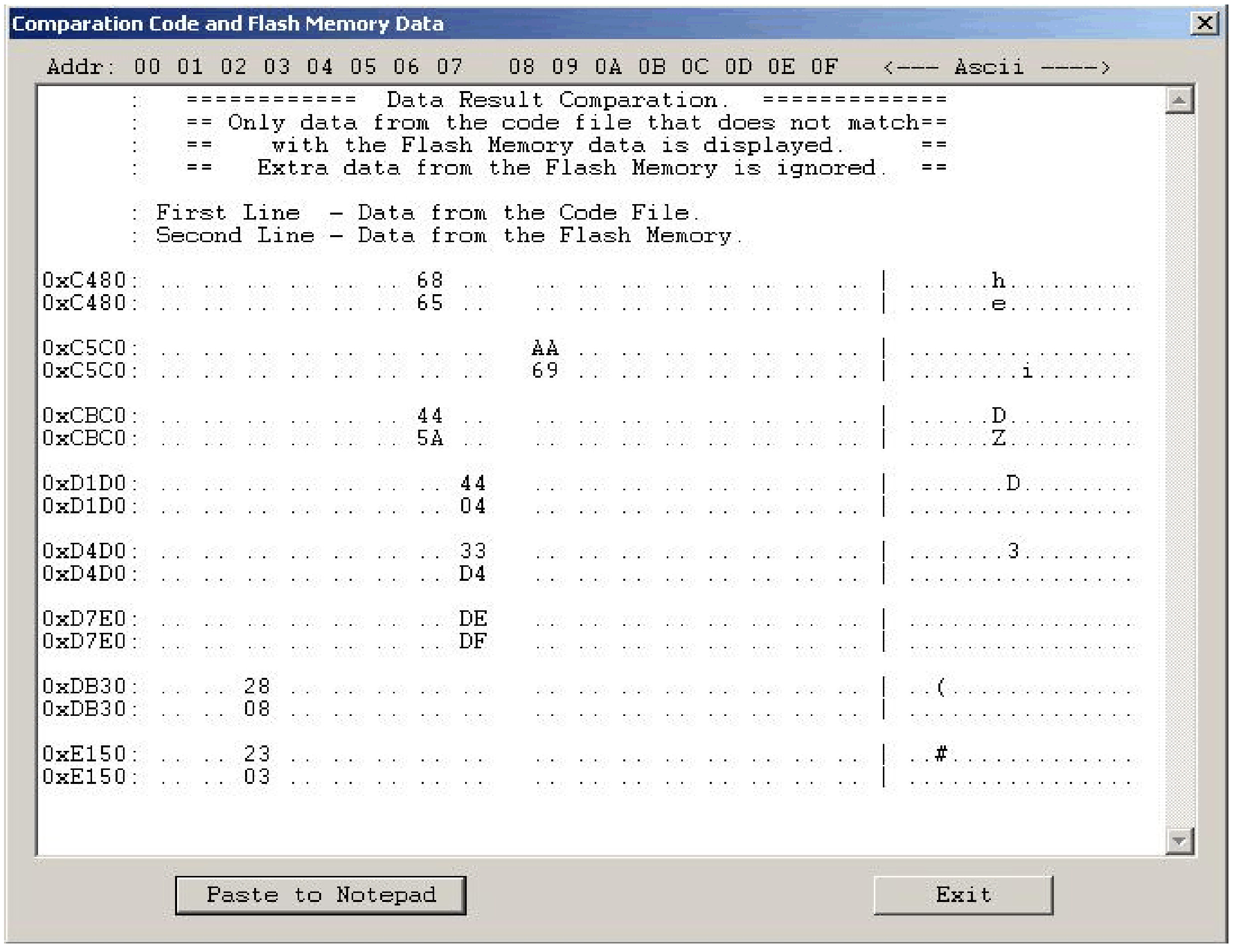
```
    ':== Data out of the Flash Memory Space of the selected MSP430. =='
```

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.

Contents of the Code File data and Flash Memory Data can be compared and differences



```
 Addr:  00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F   <--- Ascii ---->
       :   ============  Data Result Comparation.  ============
       :   == Only data from the code file that does not match==
       :   ==    with the Flash Memory data is displayed.     ==
       :   ==   Extra data from the Flash Memory is ignored.  ==

       : First Line  - Data from the Code File.
       : Second Line - Data from the Flash Memory.

0xC480: .. .. .. .. .. .. 68 ..   .. .. .. .. .. .. .. .. | ......h.........
0xC480: .. .. .. .. .. .. 65 ..   .. .. .. .. .. .. .. .. | ......e.........

0xC5C0: .. .. .. .. .. .. .. ..   AA .. .. .. .. .. .. .. | ................
0xC5C0: .. .. .. .. .. .. .. ..   69 .. .. .. .. .. .. .. | ........i.......

0xCBC0: .. .. .. .. .. .. 44 ..   .. .. .. .. .. .. .. .. | ......D.........
0xCBC0: .. .. .. .. .. .. 5A ..   .. .. .. .. .. .. .. .. | ......Z.........

0xD1D0: .. .. .. .. .. .. .. 44   .. .. .. .. .. .. .. .. | .......D........
0xD1D0: .. .. .. .. .. .. .. 04   .. .. .. .. .. .. .. .. | ................

0xD4D0: .. .. .. .. .. .. .. 33   .. .. .. .. .. .. .. .. | .......3........
0xD4D0: .. .. .. .. .. .. .. D4   .. .. .. .. .. .. .. .. | ................

0xD7E0: .. .. .. .. .. .. .. DE   .. .. .. .. .. .. .. .. | ................
0xD7E0: .. .. .. .. .. .. .. DF   .. .. .. .. .. .. .. .. | ................

0xDB30: .. .. 28 .. .. .. .. ..   .. .. .. .. .. .. .. .. | ..(.............
0xDB30: .. .. 08 .. .. .. .. ..   .. .. .. .. .. .. .. .. | ................

0xE150: .. .. 23 .. .. .. .. ..   .. .. .. .. .. .. .. .. | ..#.............
0xE150: .. .. 03 .. .. .. .. ..   .. .. .. .. .. .. .. .. | ................
```

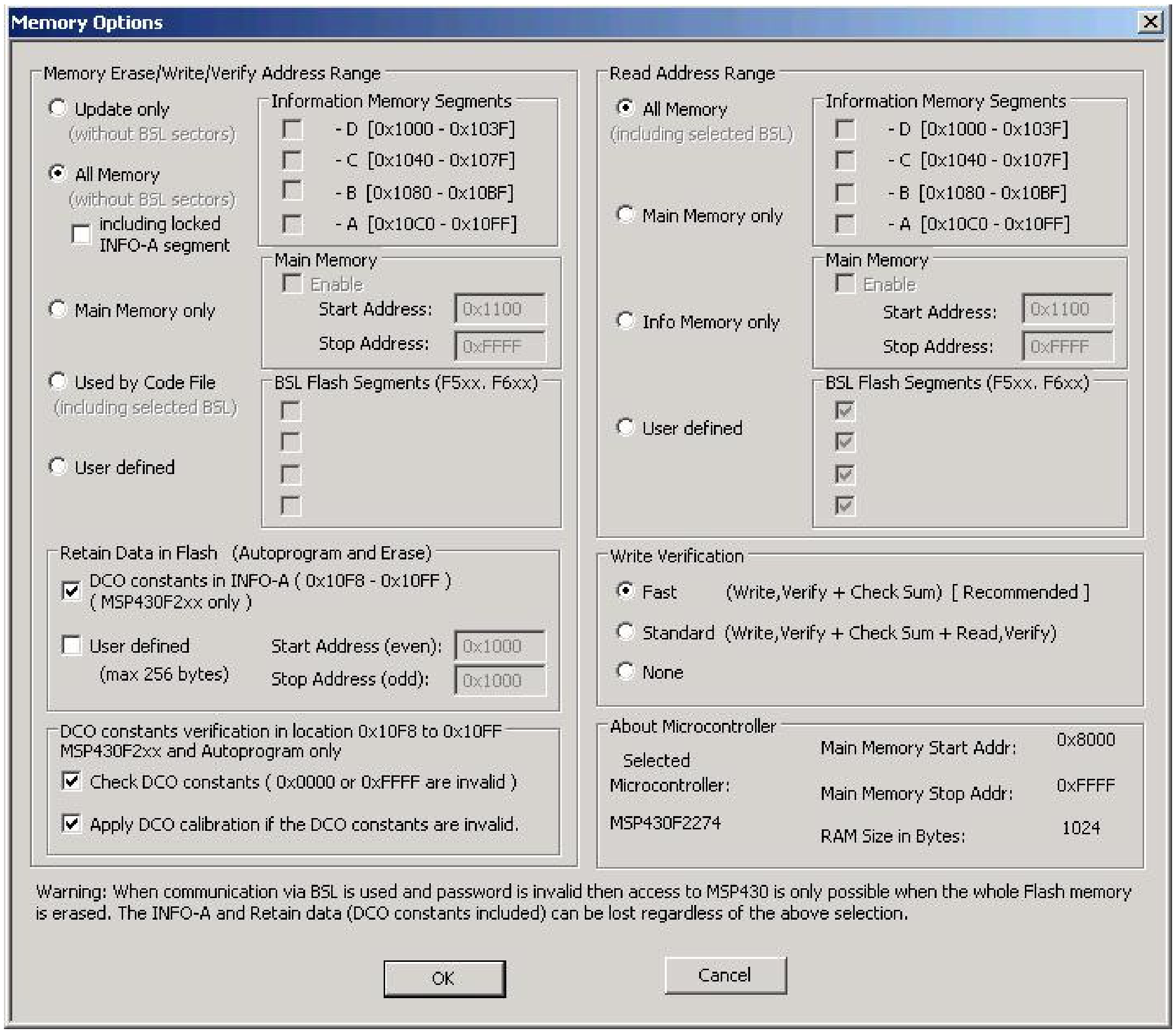
Figure 5-2

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.



Memory Options

Memory Erase/Write/Verify Address Range

- Update only (without BSL sectors)
- [✓ All Memory (without BSL sectors)]
  - including locked INFO-A segment
- Main Memory only
- Used by Code File (including selected BSL)
- User defined

Information Memory Segments

- - D [0x1000 - 0x103F]
- - C [0x1040 - 0x107F]
- - B [0x1080 - 0x10BF]
- - A [0x10C0 - 0x10FF]

Main Memory

- Enable
- Start Address: 0x1100
- Stop Address: 0xFFFF

BSL Flash Segments (F5xx. F6xx)

Retain Data in Flash (Autoprogram and Erase)

- [✓ DCO constants in INFO-A (0x10F8 - 0x10FF) (MSP430F2xx only)]
- User defined (max 256 bytes) Start Address (even): 0x1000 Stop Address (odd): 0x1000

DCO constants verification in location 0x10F8 to 0x10FF MSP430F2xx and Autoprogram only

- [✓ Check DCO constants (0x0000 or 0xFFFF are invalid)]
- [✓ Apply DCO calibration if the DCO constants are invalid.]

Read Address Range

- [✓ All Memory (including selected BSL)]
- Main Memory only
- Info Memory only
- User defined

Information Memory Segments

- - D [0x1000 - 0x103F]
- - C [0x1040 - 0x107F]
- - B [0x1080 - 0x10BF]
- - A [0x10C0 - 0x10FF]

Main Memory

- Enable
- Start Address: 0x1100
- Stop Address: 0xFFFF

BSL Flash Segments (F5xx. F6xx)

- ✓
- ✓
- ✓
- ✓

Write Verification

- [✓ Fast (Write,Verify + Check Sum) [Recommended]]
- Standard (Write,Verify + Check Sum + Read,Verify)
- None

About Microcontroller

Selected Microcontroller: MSP430F2274

Main Memory Start Addr:	0x8000
Main Memory Stop Addr:	0xFFFF
RAM Size in Bytes:	1024

Warning: When communication via BSL is used and password is invalid then access to MSP430 is only possible when the whole Flash memory is erased. The INFO-A and Retain data (DCO constants included) can be lost regardless of the above selection.

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 6.2 (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
25 CA 80 40 39 E3 F8 02
@2200
48 35 59 72 AC B8
q
```

Figure 6.2

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. When microcontroller contains INFO-A segment that can be locked (eg in the MSP430F2xx series, contains DCO constants at locations 0x10F8 to 0x10FF), then the INFO-A can be erased or left unmodified. The ***"including***

INFO-A in MSP430F2xx" should be selected on unselected respectively. When the INFO-A is not erased, then none of the data will be saved in the INFO-A, even if the data are specified in the code file. When any data should be downloaded to INFO-A segment, then the ***"including INFO-A in MSP430F2xx"*** should be selected. The ***"DCO constants"*** in the ***"Retain Data in Flash"*** group should be selected, if the DCO constants should be restored after erasing the INFO-A segment.

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 pe 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 Retain Data in Flash

The MSP430F2xx series has the DCO constants saved in the INFO-A memory at addresses 0x10F8 to 0x10FF. However, when the info segment is erased, then the DCO constants will be erased also. When the ***DCO Calibration Data*** box is selected in the ***Retain Data in Flash*** group, then contents of the info memory at location 0x10F8 to 0x10FF is read before erase process and contents of the original DCO constants (info at location 0x10F8 to 0x10FF) are restored after erasing process. The DCO constants are restored when the ***ERASE*** or ***AUTOPROGRAM*** button is pressed.
User defined option in the ***Retain Data in Flash*** group allows to specify other region to be restored after erasing the flash. This option can be used with any MSP430 microcontroller type. Location of the retain data block is not limited and can be used at any part of flash - info or main memory. Maximum size of the retain data block is limited to 256 bytes only.

Note: *When the Retain data are selected (DCO constants or used defined), then the full blank check will always failed, that of course is obvious. The selected block blank check related to the code location must pass, otherwise program will be terminated.*

6.3 DCO constants verification

In the MSP43-F2xx family the DCO is calibrated in factory and calibrated data for DCO frequencies 1, 8, 12 and 16 MHz are saved in the INFO-A segment at location 0x10F8 to 0x10FF. If from any reason the DCO constants are erased, but these data are used in the application software to tune the DCO to desired frequency, then application can not work at all due to unpredicted DCO frequency. The FlashPro430 allows to verify if the DCO constants are valid, this means if data are not 0x00 or 0xFF. If these data ale not blank and contains any data, then it can be assumed that the INFO-A segment contains correct DCO constants. When the ***"Check DCO constants"*** in the ***"DCO constants verification"*** group is selected, then software at the end of ***Autoprogram*** process is verified the DCO constants and create a warning or reporting failed ***Autoprogram*** process. The FlashPro430 can recalibrate the DCO constants, if the DCO constants are invalid. The ***"Apply DCO calibration...."*** should be selected in this case. The DCO calibration and DCO constant location are fixed regardless setup in the ***"DCO calibration"*** dialogue screen.

DCO constants location	0x10F8 to 0x10FF
DCO frequencies	16, 12, 8 and 1 MHz

6.4 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.

6.5 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/SBW 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/SBW 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/SBW 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

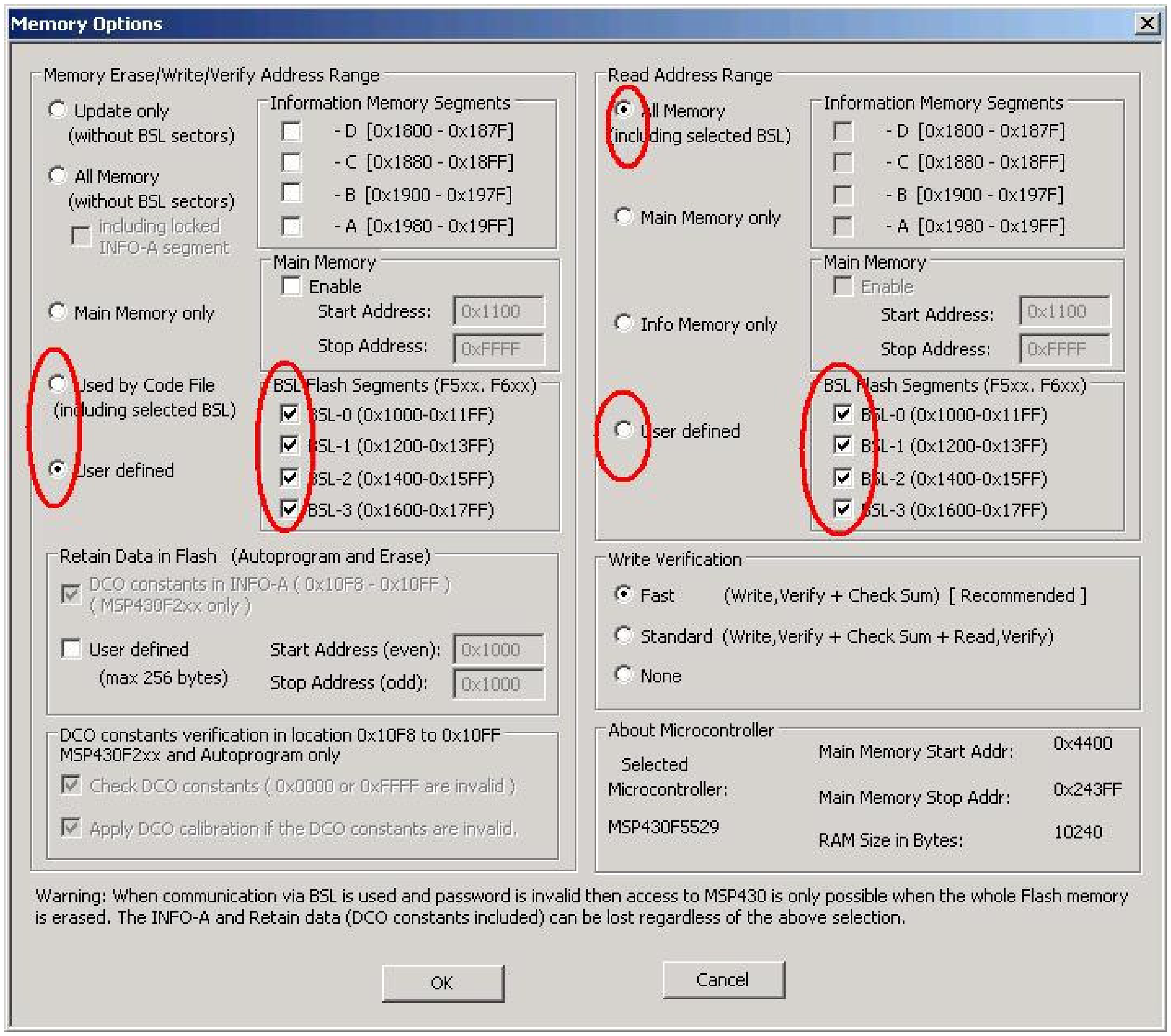
same, then verification process is finished successfully. Typically, the standard verification procedure requires the same amount of time as read/write procedure. Total programming time with standard verification is around two times longer than read/write procedure time.

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.

6.6 Write/Read the BSL Flash sectors in the F5xx/F6xx MCUs

The MSP430F5xx and MSP430F6xx microcontrollers have the BSL firmware saved in the Flash Memory sectors. By default, access to these sectors (Read/Write) is blocked, however it is possible to modify the BSL firmware if required - that allows to download the newer or custom defined BSL firmware. These BSL sectors are located in the memory space 0x1000 to 0x17FF. The FlashPro430 software allows to modify these sectors using the same method accessing the BSL memory sectors as all other memory sectors. However - to avoid unintentional erasing the BSL sectors the most used memory option - ***All memory*** - has blocked access to the BSL sectors. Access to BSL sectors is unlocked only when the ***Used by Code file*** or ***Used defined*** option is selected and desired selected BSL sectors are enabled (see Figure 6.3). Contents of the BSL sectors can be read when ***All memory*** or ***Used defined*** and desired BSL sectors are selected.

When the code file is read with code contents in the BSL sectors and the BSL sectors are not selected in the memory option ,then the following warning message will be displayed (Figure 6.4).



Memory Options

Memory Erase/Write/Verify Address Range

- ○ Update only (without BSL sectors)
- ○ All Memory (without BSL sectors)
  - ☐ including locked INFO-A segment
- ○ Main Memory only
- ○ Used by Code File (including selected BSL)
- ● User defined

Information Memory Segments

- ☐ - D [0x1800 - 0x187F]
- ☐ - C [0x1880 - 0x18FF]
- ☐ - B [0x1900 - 0x197F]
- ☐ - A [0x1980 - 0x19FF]

Main Memory

- ☐ Enable
- Start Address: 0x1100
- Stop Address: 0xFFFF

BSL Flash Segments (F5xx. F6xx)

- ✓ BSL-0 (0x1000-0x11FF)
- ✓ BSL-1 (0x1200-0x13FF)
- ✓ BSL-2 (0x1400-0x15FF)
- ✓ BSL-3 (0x1600-0x17FF)

Retain Data in Flash (Autoprogram and Erase)

- ✓ DCO constants in INFO-A (0x10F8 - 0x10FF) (MSP430F2xx only)
- ☐ User defined (max 256 bytes) Start Address (even): 0x1000 Stop Address (odd): 0x1000

DCO constants verification in location 0x10F8 to 0x10FF MSP430F2xx and Autoprogram only

- ✓ Check DCO constants (0x0000 or 0xFFFF are invalid)
- ✓ Apply DCO calibration if the DCO constants are invalid.

Read Address Range

- ● All Memory (including selected BSL)
- ○ Main Memory only
- ○ Info Memory only
- ○ User defined

Information Memory Segments

- ☐ - D [0x1800 - 0x187F]
- ☐ - C [0x1880 - 0x18FF]
- ☐ - B [0x1900 - 0x197F]
- ☐ - A [0x1980 - 0x19FF]

Main Memory

- ☐ Enable
- Start Address: 0x1100
- Stop Address: 0xFFFF

BSL Flash Segments (F5xx. F6xx)

- ✓ BSL-0 (0x1000-0x11FF)
- ✓ BSL-1 (0x1200-0x13FF)
- ✓ BSL-2 (0x1400-0x15FF)
- ✓ BSL-3 (0x1600-0x17FF)

Write Verification

- ● Fast (Write,Verify + Check Sum) [Recommended]
- ○ Standard (Write,Verify + Check Sum + Read,Verify)
- ○ None

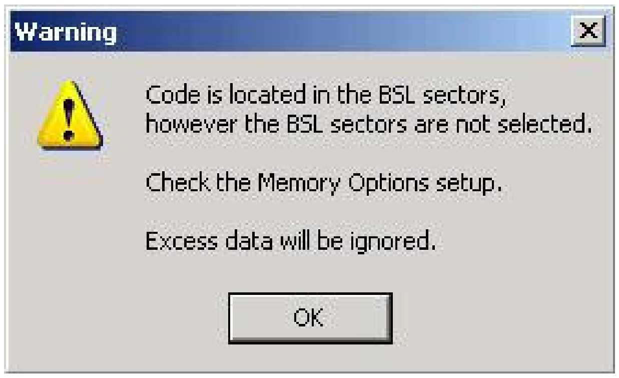
About Microcontroller

Selected Microcontroller: MSP430F5529

Main Memory Start Addr:	0x4400
Main Memory Stop Addr:	0x243FF
RAM Size in Bytes:	10240

Warning: When communication via BSL is used and password is invalid then access to MSP430 is only possible when the whole Flash memory is erased. The INFO-A and Retain data (DCO constants included) can be lost regardless of the above selection.

Figure 6.3



Warning

Code is located in the BSL sectors, however the BSL sectors are not selected.

Check the Memory Options setup.

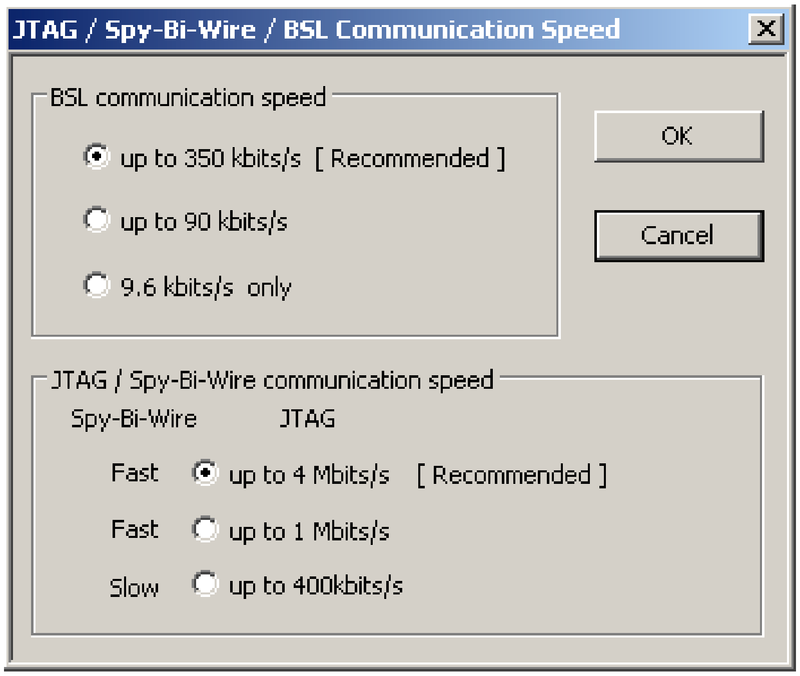
Excess data will be ignored.

Figure 6.4

7. Adapter Options

7.1 JTAG / SBW / BSL Communication Speed Dialogue Box

The JTAG/Spy-Bi-Wire/BSL Communication Speed Dialogue screen enables the user to select the communication speed between programming adapter and target microcontroller.



BSL communication speed

[✓ up to 350 kbits/s [Recommended]]
up to 90 kbits/s
9.6 kbits/s only

JTAG / Spy-Bi-Wire communication speed

Spy-Bi-Wire	JTAG
Fast	[✓ up to 4 Mbits/s [Recommended]]
Fast	up to 1 Mbits/s
Slow	up to 400kbits/s

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 two communication speeds - 90 kb/s and 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 is 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).

7.1.3 Spy-Bi-Wire Communication Speed

Default ***Spy-Bi-Wire*** communication speed between programming adapter and target device is set as a ***"Fast"*** with following specifications:

Minimum clock pulse width	- 0.25us
Minimum delay between data and clock	- 0.25us

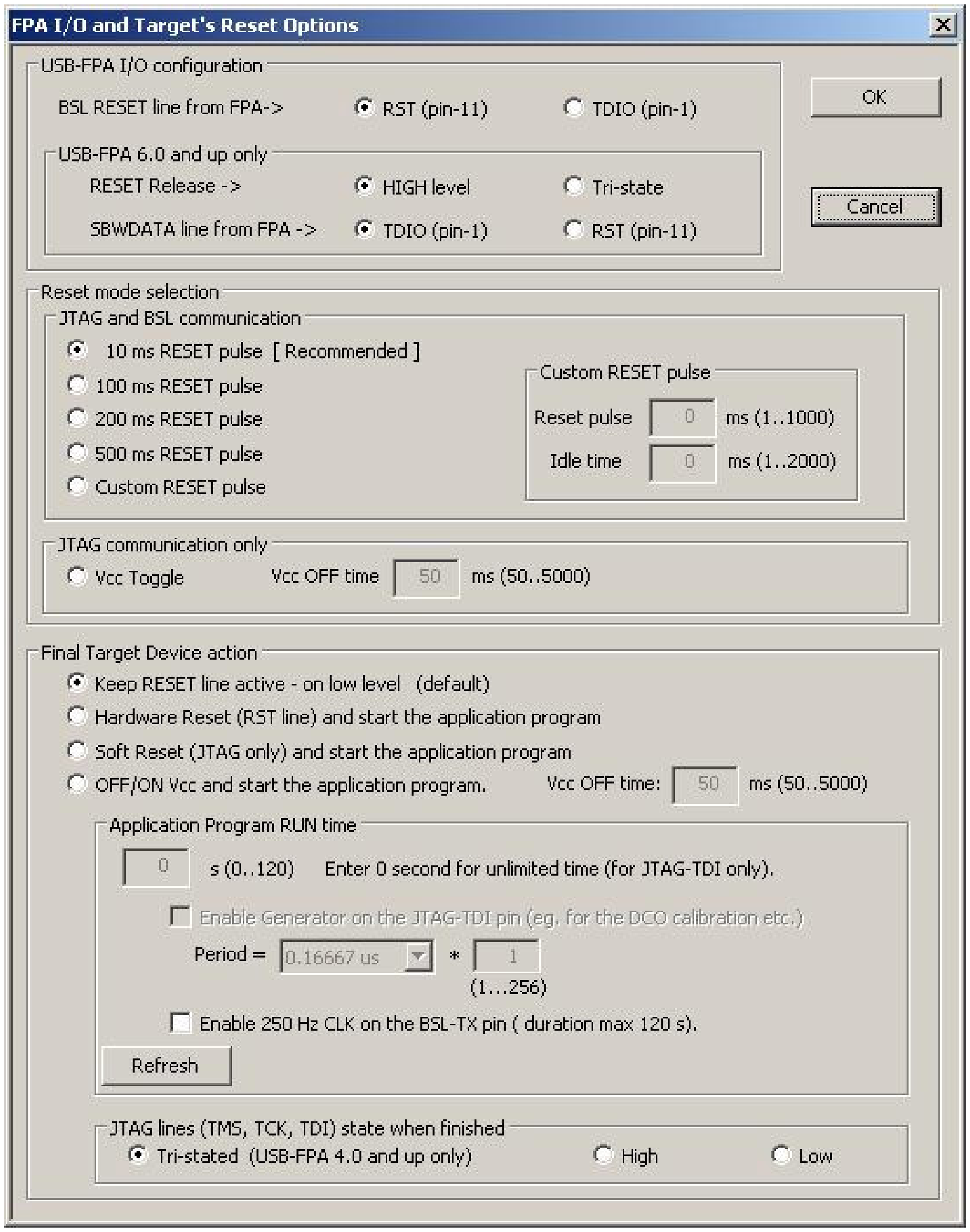
This delay allows to use the maximum capacitor in the SBWTDIO line and ground (used in the reset circuit together with the TDIO data line - see Figure 12-4 and 12-5) up to 330 pF. When this capacitor has higher value (up to 2.2 nF), or the JTAG/SBW cables are long, then the ***"Slow" Spy-Bi-Wire*** communication should be selected.

The ***"Slow" Spy-Bi-Wire*** communication has a following specifications:

Minimum clock pulse width	- 1.0 us
Minimum delay between data and clock	- 1.5 us

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.



FPA I/O and Target's Reset Options

USB-FPA I/O configuration

BSL RESET line from FPA-> [✓ RST (pin-11)] TDIO (pin-1)

USB-FPA 6.0 and up only

RESET Release -> [✓ HIGH level] Tri-state

SBWDATA line from FPA -> [✓ TDIO (pin-1)] RST (pin-11)

Reset mode selection

JTAG and BSL communication

[✓ 10 ms RESET pulse [Recommended]]
100 ms RESET pulse
200 ms RESET pulse
500 ms RESET pulse
Custom RESET pulse

Custom RESET pulse

Reset pulse 0 ms (1..1000)

Idle time 0 ms (1..2000)

JTAG communication only

Vcc Toggle Vcc OFF time 50 ms (50..5000)

Final Target Device action

[✓ Keep RESET line active - on low level (default)]
Hardware Reset (RST line) and start the application program
Soft Reset (JTAG only) and start the application program
OFF/ON Vcc and start the application program. Vcc OFF time: 50 ms (50..5000)

Application Program RUN time

0 s (0..120) Enter 0 second for unlimited time (for JTAG-TDI only).

Enable Generator on the JTAG-TDI pin (eg. for the DCO calibration etc.)

Period = 0.16667 us * 1 (1...256)

Enable 250 Hz CLK on the BSL-TX pin (duration max 120 s).

JTAG lines (TMS, TCK, TDI) state when finished

[✓ Tri-stated (USB-FPA 4.0 and up only)] High Low

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 definde 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.

If the RESET line is not connected to the target's microcontroller, then hardware reset can be created by toggling the Vcc line. In this case the **Vcc Toggle** should be selected. This reset option is available only with JTAG interface. When the BSL interface is used then the RESET line must be connected to target's device, otherwise access to the target device can not be established.

Note: It is recommended to always connect the target's device RESET pin to the RESET line of the programming adapter, even if only the JTAG communication is used. The RESET pin is used to reliable verify the JTAG security fuse during the JTAG communication initialization.

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. This allows to verify functionality of the programmed device if required. In order to do this check the

'Hardware Reset (RST line) and start the application program'

or

'Soft Reset (JTAG only) and start the application program'

or

'ON/OFF Vcc and start the application program'

option in the Reset Options window, shown in Figure 7-2. Application run time can be unlimited or limited up to 120 seconds. Limited time is specified in the ***"Application Program RUN time"*** box. When entered '0' in the ***"Program RUN time"*** box then time is unlimited. When the program run time is limited, then also it is possible to turn on a programmable generator with duty cycle 50% at frequency 250 Hz on the BSL-TX pin, or with programmable frequency on the JTAG TDI pin. Output period on the TDI pin can be programmed from 0.1667 to 42.6 us when the period increment is set to 0.1667us, or from 0.5 us to 128 us when the period increment is set to 0.5 us (see Figure 7.2). The programmable generator can be used as a reference clock for target device test (eg. DCO calibration etc). The reference clock from the TDI pin can be provided to tested circuit via resistor 4.7 k or higher to avoid overload the TDI line during communication via JTAG interface.

7.3 Preferences Dialogue Box

In the Preference Dialogue screen is possible to specify an external tools location and define a preferable audio tones during programming.

In the first edit line it can be specified the ***pdf Reader*** file name. By default it is used the Acrobat Reader ***AcroRd32.exe*** file. However it is possible to change the ***pdf Reader*** if required in the PDF Reader edit line. Using the ***Browse..*** button please select location of the pdf Reader executable file.

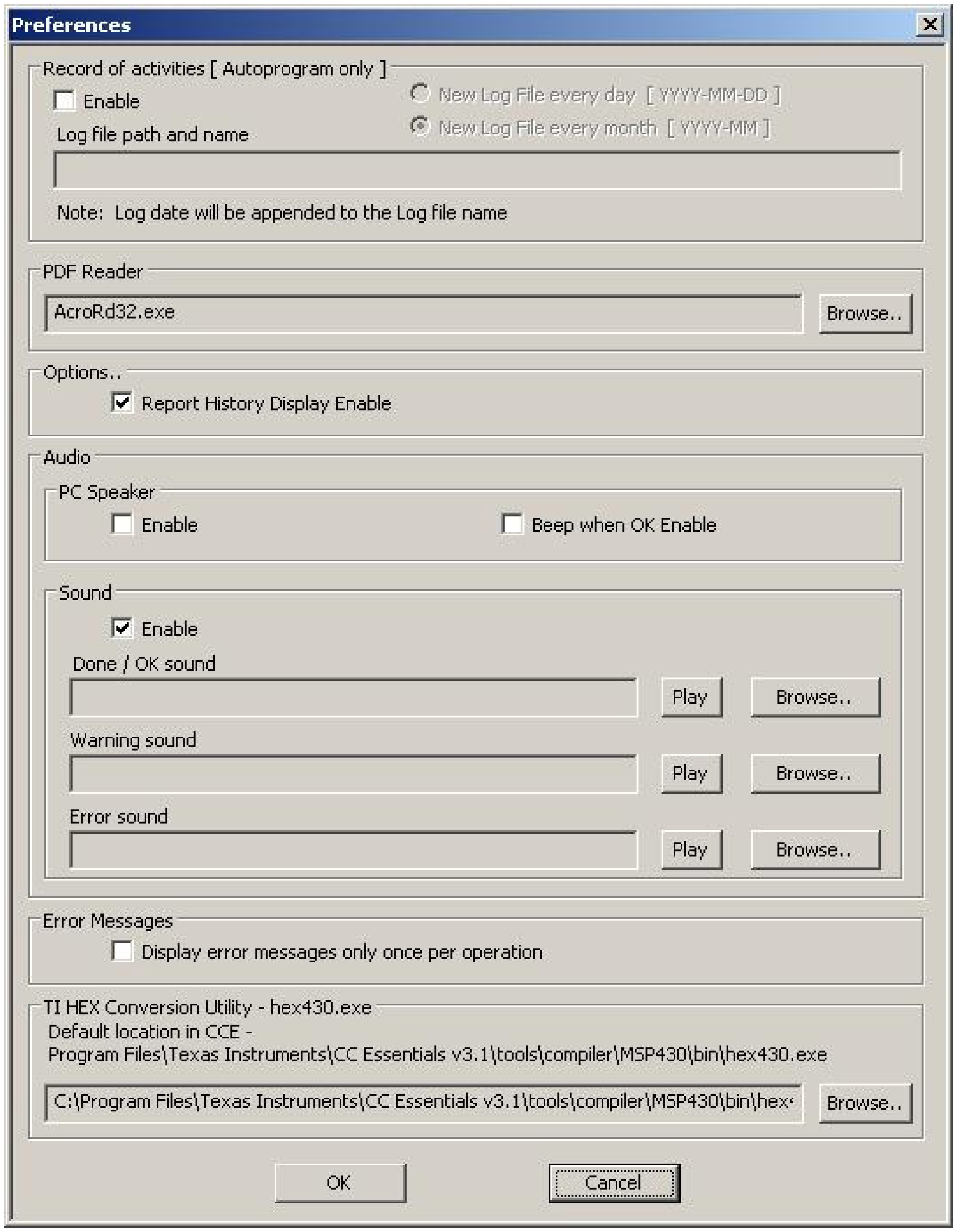
In the second edit line it should be specified location of the Texas Instruments' hex conversion utility file - **hex430.exe**. This tool is used to convert the *.out file generated by the Code Composer Essentials debugger to the Intel.hex file when the Open Code option I used and selected the TI's CCE (*.out) file. The hex430.exe file is supplied with the TI's Code Composer Essentials debugger and by default is located in directory

```
C:\Program Files\Texas Instruments\CC Essentials v3.1\tools\compiler\MSP430\bin
```

The FlashPro430 uses following keys when the *.out file is converted to *.hex file

```
hex430.exe  -romwidth=8 -memwidth=8 -i -o=file_name.hex   input_file.out
```

If the TI's CCE (*.out) option is used and the hex430.exe file cannot be found, then following message is displayed (Figure 7.4).



Preferences

Record of activities [Autoprogram only]

[ ] Enable

Log file path and name

New Log File every day [YYYY-MM-DD]

[✓ New Log File every month [YYYY-MM]]

Note: Log date will be appended to the Log file name

PDF Reader

AcroRd32.exe

Options..

[✓] Report History Display Enable

Audio

PC Speaker

[ ] Enable

[ ] Beep when OK Enable

Sound

[✓] Enable

Done / OK sound

Warning sound

Error sound

Error Messages

[ ] Display error messages only once per operation

TI HEX Conversion Utility - hex430.exe

Default location in CCE -

Program Files\Texas Instruments\CC Essentials v3.1\tools\compiler\MSP430\bin\hex430.exe

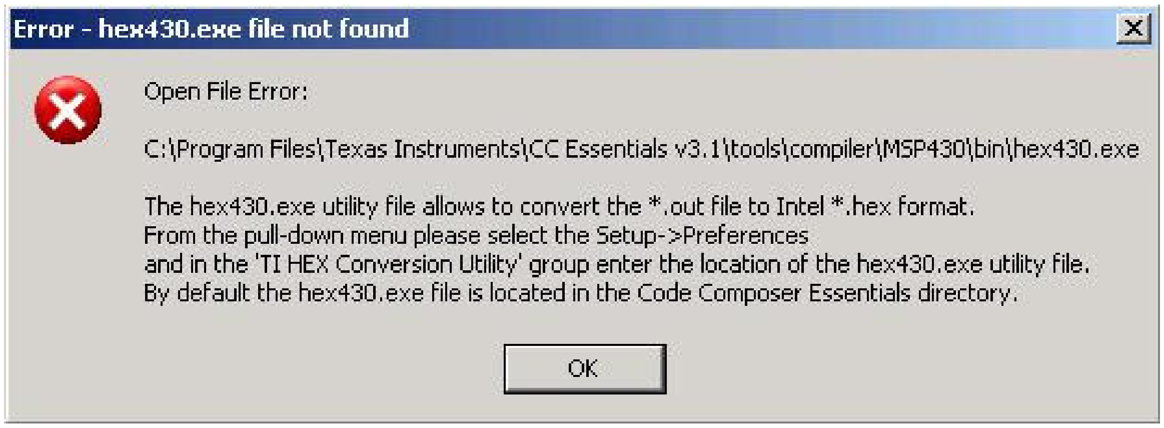
C:\Program Files\Texas Instruments\CC Essentials v3.1\tools\compiler\MSP430\bin\hex

Figure 7.3

Using the ***Browse..*** button in the ***Preferences*** Dialogue screen the new location of the hex430.exe file should be specified.

In the ***Option*** group the report history in the report window (see figure 4.1) can be enabled or disabled . 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).



Open File Error:

C:\Program Files\Texas Instruments\CC Essentials v3.1\tools\compiler\MSP430\bin\hex430.exe

The hex430.exe utility file allows to convert the *.out file to Intel *.hex format.
From the pull-down menu please select the Setup->Preferences
and in the 'TI HEX Conversion Utility' group enter the location of the hex430.exe utility file.
By default the hex430.exe file is located in the Code Composer Essentials directory.

Figure 7.4

8. Serialization

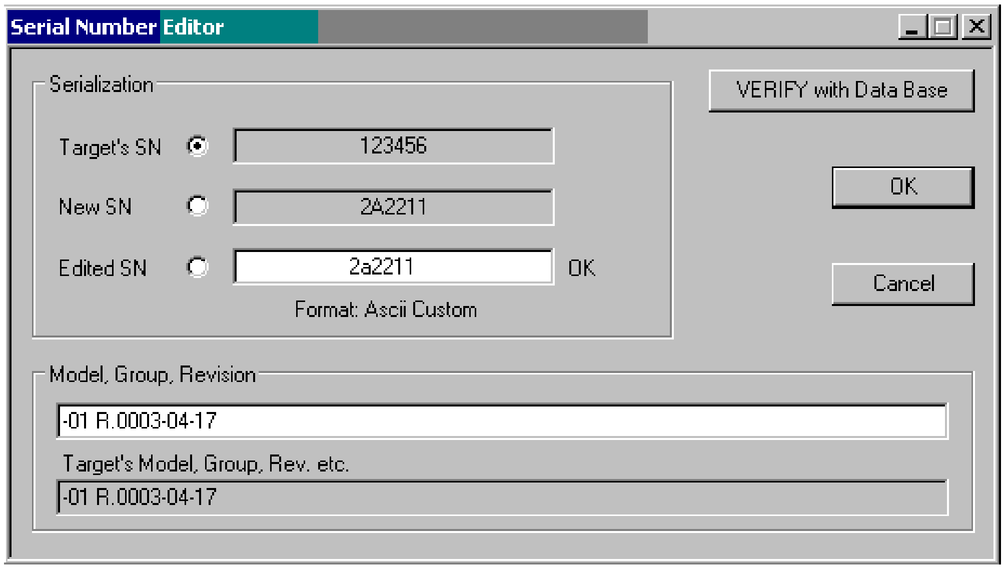
8.1 Introduction

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 ***Serial Number*** can be created automatically by incrementing the ***Serial Number*** or can be taken from the file created by user. Used serial numbers are 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, new serial number or serial number modified manually.

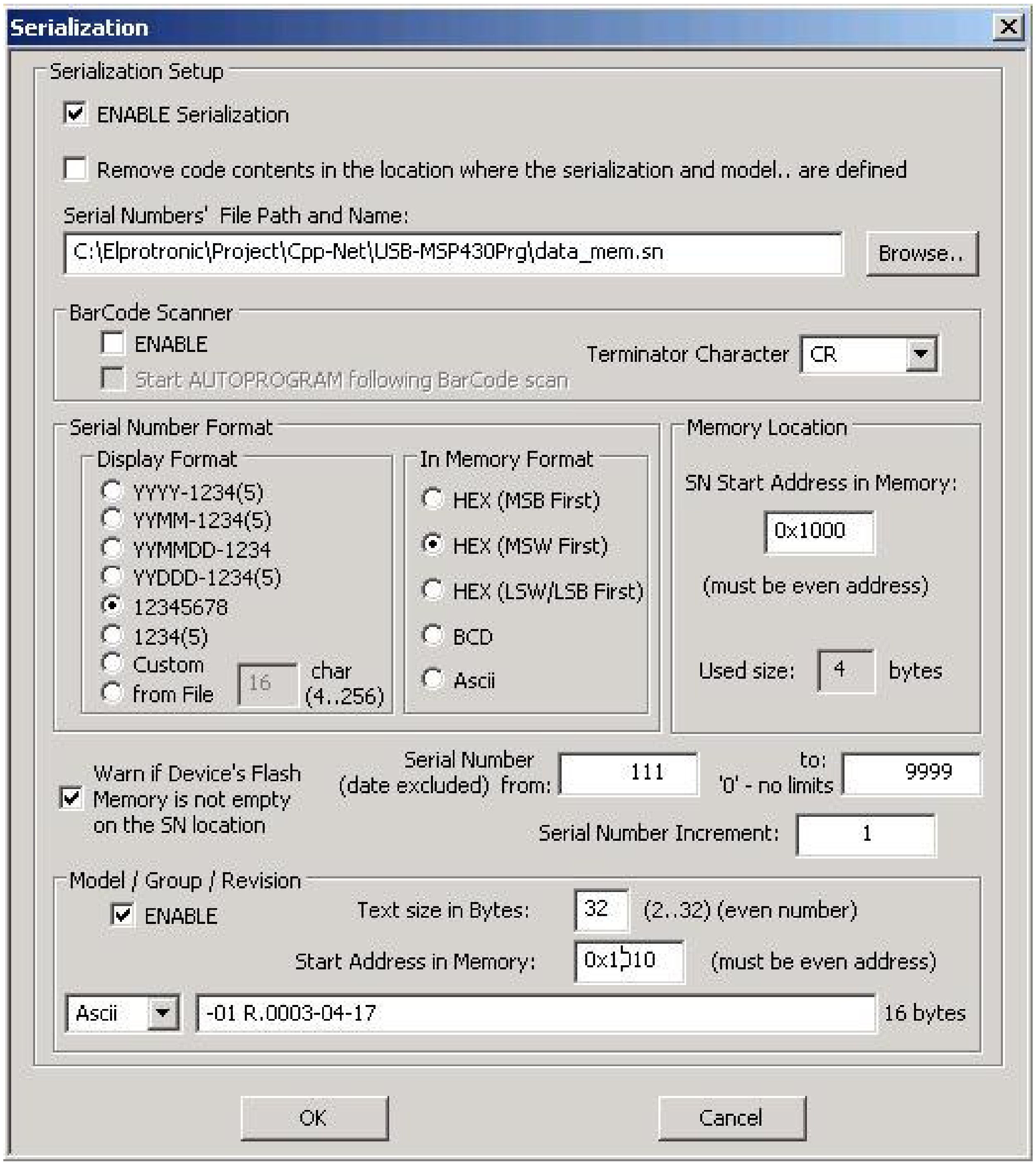


[✓ Target's SN] 123456
New SN 2A2211
Edited SN 2a2211 OK
Format: Ascii Custom

-01 R.0003-04-17
Target's Model, Group, Rev. etc.
-01 R.0003-04-17

Figure 8.1-1

8.2 Serialization Dialogue Screen



Serialization

Serialization Setup

[✓] ENABLE Serialization

[ ] Remove code contents in the location where the serialization and model.. are defined

Serial Numbers' File Path and Name: C:\Elprotronic\Project\Cpp-Net\USB-MSP430Prg\data_mem.sn

BarCode Scanner

[ ] ENABLE

[ ] Start AUTOPROGRAM following BarCode scan

Terminator Character: CR

Serial Number Format

Display Format: YYYY-1234(5) / YYMM-1234(5) / YYMMDD-1234 / YYDDD-1234(5) / [✓ 12345678] / 1234(5) / Custom / from File — 16 char (4..256)

In Memory Format: HEX (MSB First) / [✓ HEX (MSW First)] / HEX (LSW/LSB First) / BCD / Ascii

Memory Location

SN Start Address in Memory: 0x1000 (must be even address)

Used size: 4 bytes

[✓] Warn if Device's Flash Memory is not empty on the SN location

Serial Number (date excluded) from: 111 to: 9999 ('0' - no limits)

Serial Number Increment: 1

Model / Group / Revision

[✓] ENABLE

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

Start Address in Memory: 0x1010 (must be even address)

Ascii: -01 R.0003-04-17 — 16 bytes

Figure 8-2

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.

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.
7. Custom	- (SN Format - **F6**) 4 to 16 Ascii characters or hexadecimal numbers entered manually or from the Bar-Code Reader.
8. From the file	- (SN Format - **F7**) 4 to 16 Ascii characters or hexadecimal numbers taken from the user created file.

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 and from the file serial number can be stored in Ascii or HEX format.

8.2.2.1 HEX (MSB first, MSW first, LSW/LSB first) formats

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(MSB first)*** is selected then the first hex integer number is saved as a first byte and the second number - as a next byte etc. in the Flash memory location.

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/LSB 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   -> ***HEX(MSW first)***
007B   07D3   -> ***HEX(LSW first)***

when integer numbers are viewed, or as

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

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   -> ***HEX(MSW first)***
007B   012F   -> ***HEX(LSW first)***

or

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

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: 123456768 - 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 61 4E -> ***HEX(MSB first)***

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

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

30 39 (bytes) -> ***HEX(MSB first)***

39 30 (bytes) -> ***HEX(MSW first)*** or ***HEX(LSW/LSB 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 (MSB first)
or
      C1 69 A5 02     ->  HEX (LSW/LSB 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: 123456768 - 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 ------>          <– Ascii format –>
          32  30  30  33 30 31 32  33           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 ------>          <– Ascii format –>
          30  33  30  33 30 31 32  33           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: 123456768 - 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*** or ***from the file*** - size in FLASH - defined size in bytes

Taken from the file or entered manually Ascii string will be saved in the flash memory.
When the ***Ascii*** format is selected, then the Ascii string is saved in memory **"as is".**
All Ascii characters can be used**.** For example the entered following string

02WX24S234

will be saved in memory as

30 32 57 58 32 34 53 32 33 34 -> "02WX24S234"

When the ***HEX*** format is selected, then the string is converted to HEX format (only hex characters are accepted - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
All two character pairs are converted to hex format and saved in memory.
For example the entered following string

02A3B109E12F

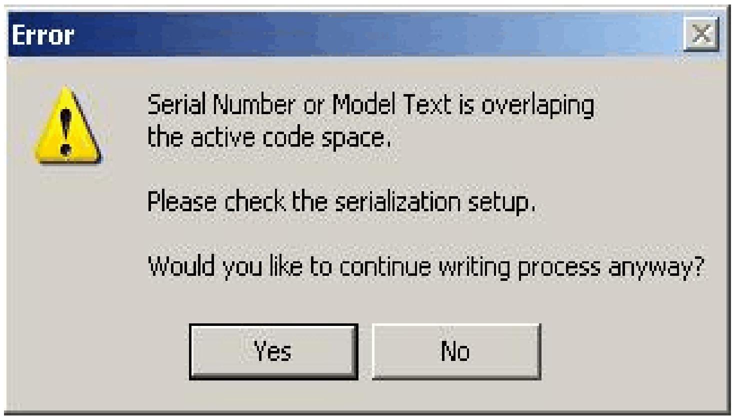
will be saved in memory as

HEX(MSB first) -> 02 A3 B1 09 E1 2F

or ***HEX(LSB first)*** -> 2F E1 09 B1 A3 02

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:


Error

Serial Number or Model Text is overlaping the active code space.

Please check the serialization setup.

Would you like to continue writing process anyway?

Figure 8.2.1-1

8.2.3 Model, Group, Revision

Custom text or data (hex), saved in target device's flash memory is a string or data, up to 32 characters (bytes) long, in Ascii or hex format. It can contain any text or data, 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 or data must be specified in the field ***'Size in Bytes***'. Size value can be any *even* number between 2 and 32. The location of the text/data 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 will be displayed.

The text to be saved in the flash memory can be entered in the edit line. Bytes can be entered as an Ascii, if ***Ascii*** option is selected, or in hex bytes, if the ***Hex*** option is selected. When the ***Ascii/Hex*** selector is modified, then the contents data is displayed as an Ascii string or as a hex bytes data.

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.

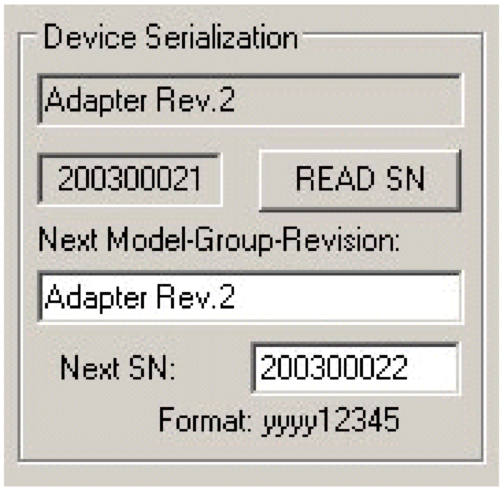
Current target's label (model text and serial number) can be read at any time by pressing ***READ SN*** button located in the ***'Device Serialization'*** group (see figure 8-2).

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.


Device Serialization
Adapter Rev.2
200300021

Next Model-Group-Revision:
Adapter Rev.2
Next SN: 200300022
Format: yyyy12345

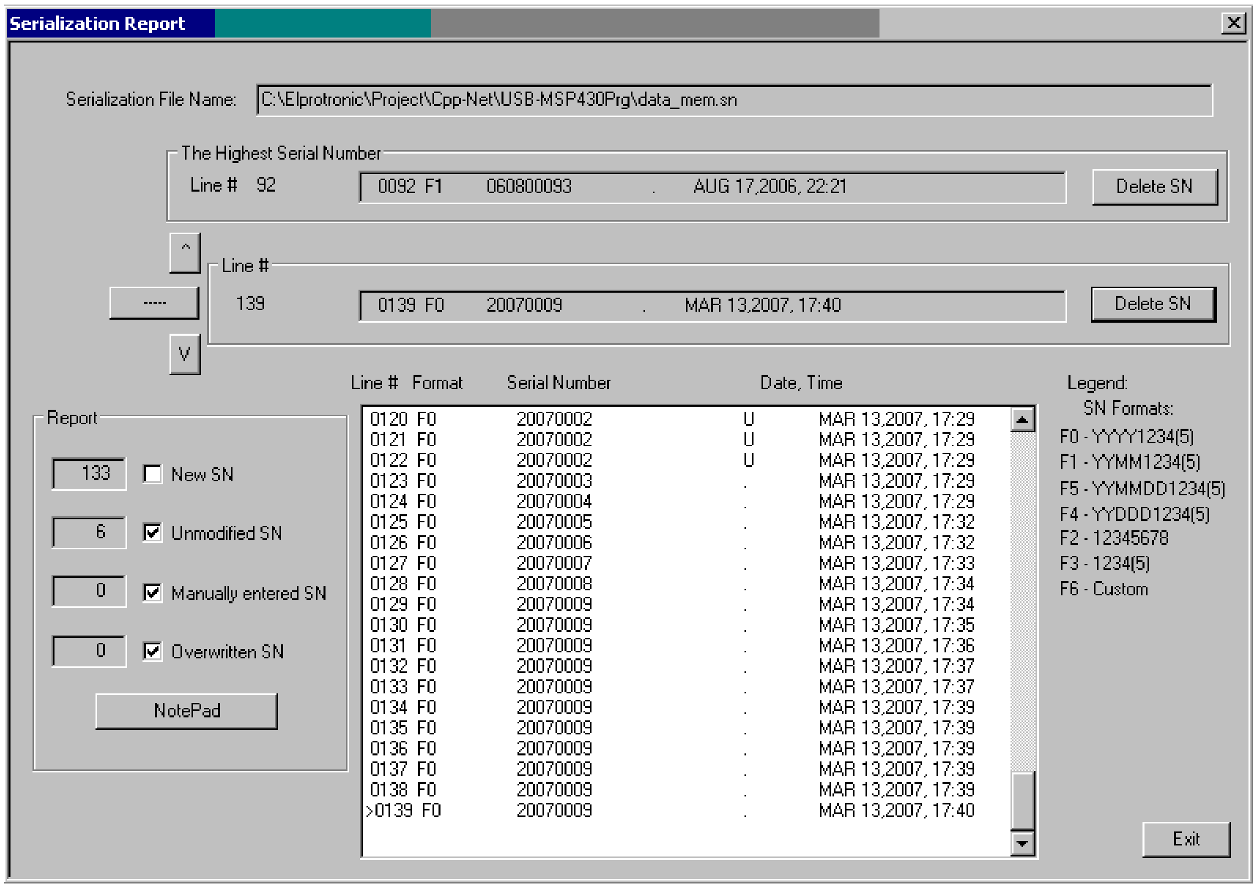
Figure 8-2

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.

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: C:\Elprotronic\Project\Cpp-Net\USB-MSP430Prg\data_mem.sn

The Highest Serial Number
Line # 92   0092 F1   060800093   .   AUG 17,2006, 22:21

Line #
139   0139 F0   20070009   .   MAR 13,2007, 17:40

Report
133 [ ] New SN
6 [✓] Unmodified SN
0 [✓] Manually entered SN
0 [✓] Overwritten SN

```
Line #  Format   Serial Number          Date, Time
 0120  F0        20070002         U     MAR 13,2007, 17:29
 0121  F0        20070002         U     MAR 13,2007, 17:29
 0122  F0        20070002         U     MAR 13,2007, 17:29
 0123  F0        20070003         .     MAR 13,2007, 17:29
 0124  F0        20070004         .     MAR 13,2007, 17:29
 0125  F0        20070005         .     MAR 13,2007, 17:32
 0126  F0        20070006         .     MAR 13,2007, 17:32
 0127  F0        20070007         .     MAR 13,2007, 17:33
 0128  F0        20070008         .     MAR 13,2007, 17:34
 0129  F0        20070009         .     MAR 13,2007, 17:34
 0130  F0        20070009         .     MAR 13,2007, 17:35
 0131  F0        20070009         .     MAR 13,2007, 17:36
 0132  F0        20070009         .     MAR 13,2007, 17:37
 0133  F0        20070009         .     MAR 13,2007, 17:37
 0134  F0        20070009         .     MAR 13,2007, 17:39
 0135  F0        20070009         .     MAR 13,2007, 17:39
 0136  F0        20070009         .     MAR 13,2007, 17:39
 0137  F0        20070009         .     MAR 13,2007, 17:39
 0138  F0        20070009         .     MAR 13,2007, 17:39
>0139  F0        20070009         .     MAR 13,2007, 17:40
```

Legend:
SN Formats:
F0 - YYYY1234(5)
F1 - YYMM1234(5)
F5 - YYMMDD1234(5)
F4 - YYDDD1234(5)
F2 - 12345678
F3 - 1234(5)
F6 - Custom

Figure 8.3-1 Serialization Report Dialogue screen

8.4 SN data file

The FlashPro430 software allows to download the serial number from custom defined data file. When the data file is used then in the serialization dialogue screen the ***Serial Number Format*** -> ***From File*** should be selected.

The SN data file can contains list of serial numbers. Format of the serial numbers can be specified in the serialization dialogue screen (Figure 8.2) as Ascii or HEX. The SN data file can be created in any DOS editor like Notepad.exe. In this file any data specified after semicolon (;) will be ignored and can be used as a comment only. Data file should contains header and serial number list. Following list of commands started from # can be specified in the header:

`#SN_LIST`

Data file contains Serial number list.

```
#SN_SIZE          number           ;optional
```

Overwrite size of the custom defined serial number size (see Figure 8.2). If the #SN_SIZE is not specified, then the data specified in the serialization dialogue screen is used.

```
#SN_HEX_MSB                        ;optional
```

Select the HEX MSB first format regardless setup in the serialization dialogue screen.

```
#SN_HEX_LSB                        ;optional
```

Select the HEX LSB first format regardless setup in the serialization dialogue screen.

When the format is specified as the hex format (**`#SN_HEX_MSB`** or **`#SN_HEX_LSB)`** then string can contains gaps between numbers or 0x on the front. All gaps will be removed. Up to 256 characters (gaps excluded) can be specified for one number. All characters must be located in one line for the same SN.

Example:

The hex number **`A434BC5696AD10ACF0`**

can be specified as

`A434BC5696AD10AC`

or

`0xA434BC5696AD10AC`

or

```
    A434 BC56 96AD 10AC
```
or
```
   A4 34 BC 56 96 AD 10 AC F0
```

```
#SN_ASCII                                ;optional
```
Select an ASCII format regardless setup in the serialization dialogue screen.

```
#SN_PREFIX        string                 ;optional
#SN_SUFFIX        string                 ;optional
```
Serial number can contains up to 32 characters. If part of characters are the same in specified serial number list, then the repeatable part can be specified in the SN_PREFIX, or SN_SUFFIX, and only modified part of serial numbers can be listed. Serial number is combined as a string starting from prefix, modified part and ending with suffix.. For example if the following serial number should be created

```
AB2007X-0001-BMR
AB2007X-0002-BMR
AB2007X-0003-BMR
```

can the SN be specified as follows

```
#SN_PREFIX    AB2007X-
#SN_SUFFIX    -BMR
```

and list of following serial numbers

```
0001
0002
0003
```

Prefix and /or suffix numbers can be modified in the list if required, eg.

```
#SN_PREFIX    AB2007X-
#SN_SUFFIX    -BMR
0001
0002
0003
#SN_PREFIX    AB2007V-
0001
```

```
        0002
        0003
```

that defined following serial numbers

```
        AB2007X-0001-BMR
        AB2007X-0002-BMR
        AB2007X-0003-BMR
        AB2007V-0001-BMR
        AB2007V-0002-BMR
        AB2007V-0003-BMR
```

Example of the Serial Number list (5 lines only in this example)

```
; =====================================================
;   Serial Number List
; SN format - Ascii
; =====================================================
#IEEE_SN_LIST
#SN_SIZE    12

WX5E2007001P
WX5E2007002P
WX5E2007003P
WX5E2007004P
WX5E2007005P
; =====================================================
```

The same Serial Number list with specified prefix /suffix

```
; =====================================================
;   Serial Number List
; SN format - Ascii
; =====================================================
#IEEE_SN_LIST
#SN_SIZE    12
#SN_PREFIX       WX5E2007      ;any Ascii character
#SN_SUFFIX       P

001
002
003
```

```
004
005
; ==================================================
```

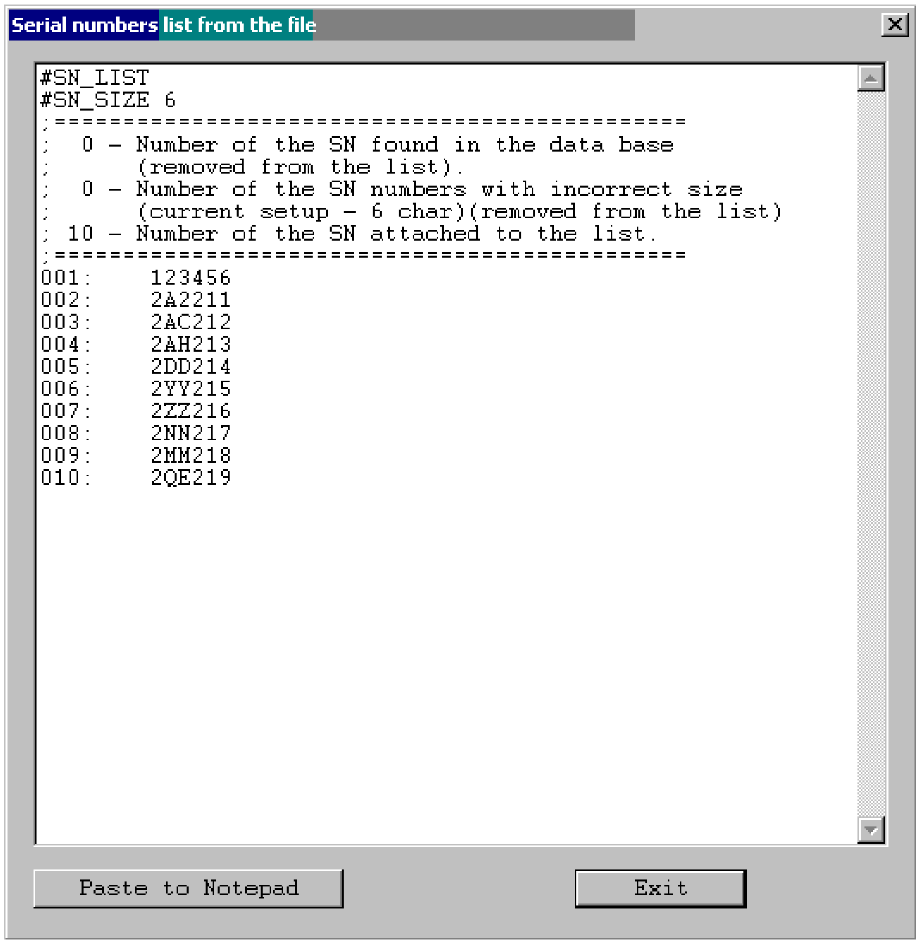
When the SN data file is prepared, then at the first the data base file should be opened(see Figure 8.2). When the desired ***Serial Number Format*** is selected, then using the ***SN/IEEE file*** button located in the main dialogue screen (Figure 4.1) the desired SN file should be opened. Selected file is converted to final format and all listed serial numbers are verified with the data base file if there was note used before. If the specified SN have been used before, then these numbers are removed from the SN list. When the SN file is read and verified, then the pending SN list is displayed in the screen (Figure 8.4-1) with following information displayed on the top of the list

* number of the SN found in data base and removed from the pending list

* number of the Serial Numbers with incorrect size and removed from the pending list

* number of the accepted SN

When the ***"Paste to Notepad"*** button is pressed, then the pending Serial Number list can be saved in format ready to be used as a valid SN data file if required.



```
#SN_LIST
#SN_SIZE 6
;============================================
;  0 - Number of the SN found in the data base
;      (removed from the list).
;  0 - Number of the SN numbers with incorrect size
;      (current setup - 6 char)(removed from the list)
; 10 - Number of the SN attached to the list.
;============================================
001:    123456
002:    2A2211
003:    2AC212
004:    2AH213
005:    2DD214
006:    2YY215
007:    2ZZ216
008:    2NN217
009:    2MM218
010:    2QE219
```

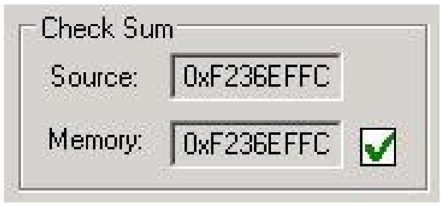
Figure 9.1

9. Check Sum Options

Programming software has two groups of check sum (CS) calculation. The first group is used for internal programming verification and the second group can be used for firmware verification in application software.

The CS used for internal verification is calculating CS only for specified words in the code file regardless of the flash memory size, location etc. This CS is useful only inside the programmer, because programmer has all information about programmed and empty bytes location. This method is also useful if only part of the code is programmed in the flash (append option). All not programmed words in the programming process are ignored, even if these words are not empty in the flash.

The check sum used for internal programming verification is displayed in the Check Sum Group (Figure 9.1) (see the Main Dialog screen - Figure 4.1)



Source: 0xF236EFFC

Memory: 0xF236EFFC ✓

Figure 9.1

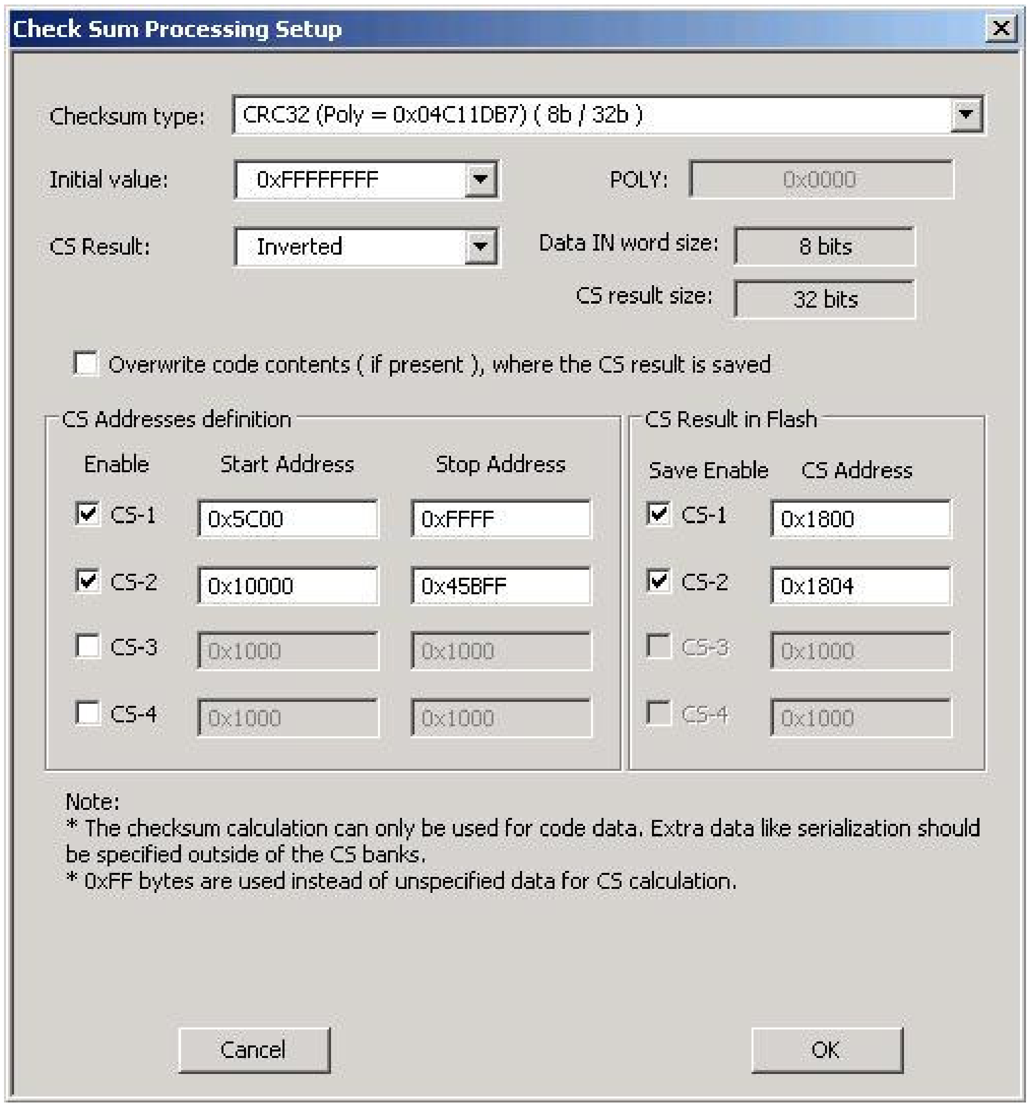
In the source line is displayed the arithmetic sum of the code contents with added contents of the serialization, model etc. if selected. Arithmetic sum is calculated as the sum of 16-bits unsigned words - result is 32 bits unsigned. Only programmed words are taken for calculation. All other not used words are ignored. All bytes are converted to 16-bits words as follows (for simplicity - format casting is not present in this example):

word = data[address] + (data[address+1]<<8)

where address is even and incremented by 2.

In the memory line is displayed the CS result taken from the flash memory, calculated in the same way as the CS taken from the source. Only words defined in the source are taken from the flash memory for calculation.

Second group of the CS is custom defined Check Sum that can be used by firmware for code verification in the flash. Up to four CS block can be specified and CS results can be saved in the flash for verification. Size of each CS block and CS result location in flash are defined by the user. The Check Sum Options dialog (figure 9-2) is selected from following pull down menu:
Setup -> Check Sum Options


Check Sum Processing Setup

Checksum type: CRC32 (Poly = 0x04C11DB7) (8b / 32b)

Initial value: 0xFFFFFFFF

POLY: 0x0000

CS Result: Inverted

Data IN word size: 8 bits

CS result size: 32 bits

☐ Overwrite code contents (if present), where the CS result is saved

CS Addresses definition

Enable	Start Address	Stop Address
☑ CS-1	0x5C00	0xFFFF
☑ CS-2	0x10000	0x45BFF
☐ CS-3	0x1000	0x1000
☐ CS-4	0x1000	0x1000

CS Result in Flash

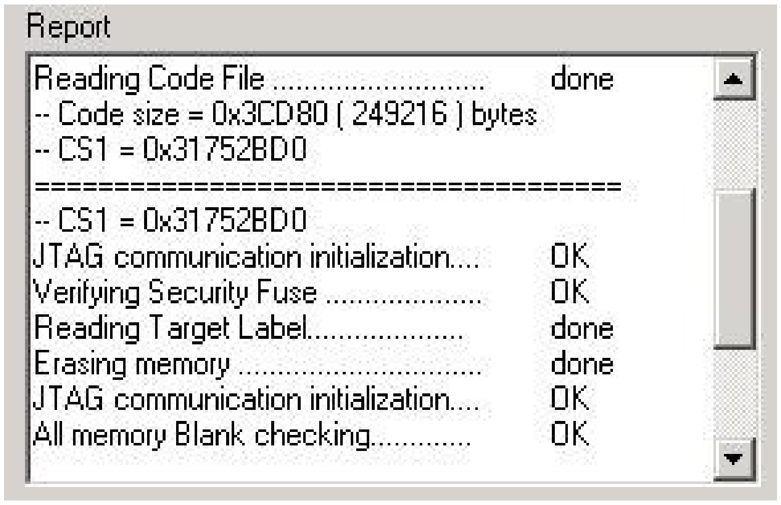
Save Enable	CS Address
☑ CS-1	0x1800
☑ CS-2	0x1804
☐ CS-3	0x1000
☐ CS-4	0x1000

Note:
* The checksum calculation can only be used for code data. Extra data like serialization should be specified outside of the CS banks.
* 0xFF bytes are used instead of unspecified data for CS calculation.

Figure 9.2

Start Address should be even, and the Stop Address should be odd. CS result address in the flash should be even. Make sure that the CS result is saved out of the CS block space. Otherwise the CS result will modify the contents of the CS inside the specified block. CS result after the second calculation would not be the same and CS result would be useless.

When the ***CS Result Save*** option is not selected then the CS of the selected block is calculated and CS result displayed in the report window only (Figure 9.3) This option can be used for CS code verification defined as the code form Start to End Addresses with 0xFF data in the not specified code location.

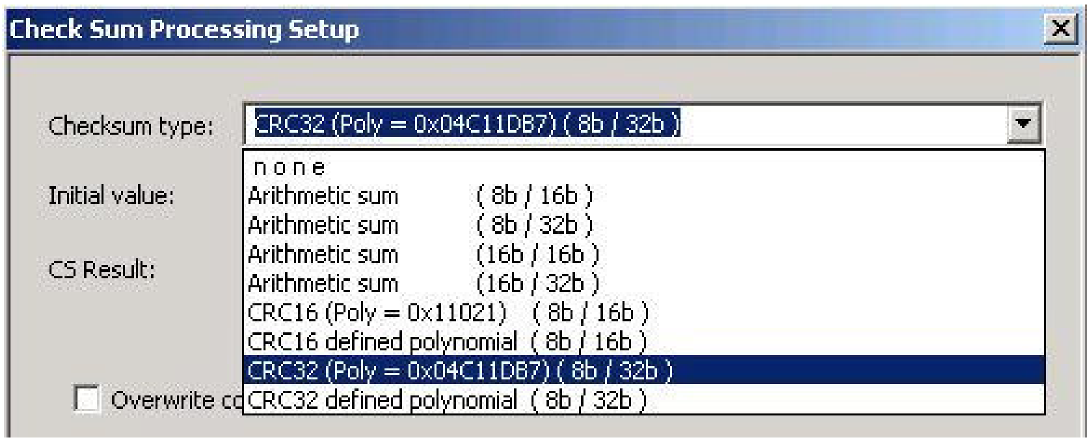


Report

```
Reading Code File ..........................     done
-- Code size = 0x3CD80 ( 249216 ) bytes
-- CS1 = 0x31752BD0
=====================================
-- CS1 = 0x31752BD0
JTAG communication initialization....     OK
Verifying Security Fuse ...................     OK
Reading Target Label...................     done
Erasing memory ..............................     done
JTAG communication initialization....     OK
All memory Blank checking............     OK
```

Figure 9.3

Type of the CS can be selected from the following list (Figure 9.4)



Check Sum Processing Setup

Checksum type: CRC32 (Poly = 0x04C11DB7) (8b / 32b)

Initial value:

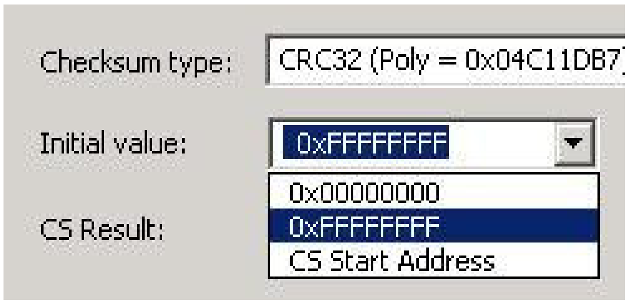
CS Result:

Overwrite co

```
n o n e
Arithmetic sum          ( 8b / 16b )
Arithmetic sum          ( 8b / 32b )
Arithmetic sum          (16b / 16b )
Arithmetic sum          (16b / 32b )
CRC16 (Poly = 0x11021)    ( 8b / 16b )
CRC16 defined polynomial  ( 8b / 16b )
CRC32 (Poly = 0x04C11DB7) ( 8b / 32b )
CRC32 defined polynomial  ( 8b / 32b )
```

Figure 9.4

Initial value for CS calculation can be selected as zero, all 0xFFs or as the Start Address from pull down menu (Figure 9.5).

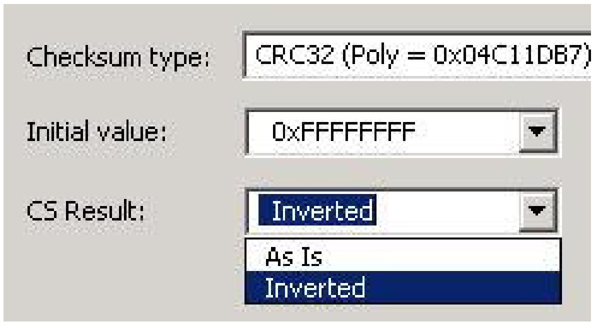

Checksum type: CRC32 (Poly = 0x04C11DB7)

Initial value: 0xFFFFFFFF

CS Result:

Figure 9.5

CS result can be used ***As Is*** or can be ***inverted*** (Figure 9.6).


Checksum type: CRC32 (Poly = 0x04C11DB7)

Initial value: 0xFFFFFFFF

CS Result: Inverted

Figure 9.6

Data size (byte or 16 bits word) used for calculation and CS result size is displayed in the dialog screen as ***Data IN word size*** and ***CS Result size*** (Figure 9.2). Polynomial contents (if required) can be specified in the POLY edit line in HEX format (eg. 0x1234).

9.1 Check Sum types

Following Check Sum types are implemented (Figure 9.4)

Arithmetic Sum (8b / 16b)

Check Sum is calculated as modulo 16-bits sum of all bytes (unsigned) from Start to the End Addresses as follows

```
CS = CS_initial_value;
for ( addr = StartAddress;  addr <= EndAddress; addr++ )
{
   CS = CS + (unsigned int) data[addr];
}
CS = 0xFFFF & CS;
if( cs_inverted )
     CS = 0xFFFF ^ CS;
```

Arithmetic Sum (8b / 32b)

Check Sum is calculated as modulo 32-bits sum of all bytes (unsigned) from Start to the End Addresses as follows

```
CS = CS_initial_value;
for ( addr = StartAddress;  addr <= EndAddress; addr++ )
{
   CS = CS + (unsigned long) data[addr];
}
CS = 0xFFFFFFFF & CS;
if( cs_inverted )
    CS = 0xFFFFFFFF ^ CS;
```

Arithmetic Sum (16b / 16b)

Check Sum is calculated as modulo 16-bits sum of all 2-byte words (unsigned) from Start to the End Addresses as follows

```
CS = CS_initial_value;
for ( addr = StartAddress;  addr <= EndAddress; addr=addr+2 )
{
 CS = CS + (unsigned int)data[addr] + (unsigned int)data[addr+1];
}
CS = 0xFFFF & CS;
if( cs_inverted )
     CS = 0xFFFF ^ CS;
```

Arithmetic Sum (16b / 32b)

Check Sum is calculated as modulo 32-bits sum of all 2-byte words (unsigned) from Start to the End Addresses as follows

```
CS = CS_initial_value;
for ( addr = StartAddress;  addr <= EndAddress; addr=addr+2 )
{
 CS = CS+(unsigned long)data[addr] + (unsigned long)data[addr+1];
}
CS = 0xFFFFFFFF & CS;
if( cs_inverted )
    CS = 0xFFFFFFFF ^ CS;
```

CRC16 (Poly 0x11201)   - (8b / 16b)   (Named as CRCCCITT)

and

CRC16 defined polynomial   - (8b / 16b)

Check Sum is calculated as CRC16 from each bytes from Startto the End Addresses as follows

```
CS = CS_initial_value;
for ( addr = StartAddress;  addr <= EndAddress; addr++ )
{
  CS = CS_CRC16_8to16( (long)data[addr], CS );
}
CS = 0xFFFF & CS;
if( cs_inverted )
     CS = 0xFFFF ^ CS;
```

where

```
unsigned long     CS_CRC16_8to16( long data, unsigned long crc )
{
 unsigned long tmp;
      tmp = 0xFF & ((crc >> 8) ^ data );
      crc = (crc << 8) ^ crc_tab32[tmp];
  return( 0xFFFF & crc );
}
```

The CRC table is generated first as follows:

```
      CS_init_crc16_tab( 0x1021 );              for CRC CCITT
      CS_init_crc16_tab( CRC_def_POLY  );       for CRC16 defined polynomial
```

where

```
void CS_init_crc16_tab( unsigned short poly )
{
    int i, j;
    unsigned short crc, c;

    for (i=0; i<256; i++)
      {
        crc = 0;
        c   = ((unsigned short) i) << 8;

        for (j=0; j<8; j++)
            {
            if ( (crc ^ c) & 0x8000 )
                  crc = ( crc << 1 ) ^ poly;
            else
                  crc =   crc << 1;
            c = c << 1;
        }
        crc_tab32[i] = (unsigned long)(0xFFFF & crc);
    }
}
```

CRC32 (Poly 0x04C11DB7) - (8b / 32b) (Named as IEEE 802-3)

and

CRC32 defined polynomial - (8b / 32b)

Check Sum is calculated as CRC32 from each bytes from Start to the End Addresses as follows

```
      CS = CS_initial_value;
      for ( addr = StartAddress;  addr <= EndAddress; addr++ )
      {
        CS = CS_CRC32_8to32( (long)data[addr], CS );
      }
      CS = 0xFFFFFFFF & CS;
      if( cs_inverted )
           CS = 0xFFFFFFFF ^ CS;
```

where

```
unsigned long     CS_CRC32_8to32( long data, unsigned long crc )
{
  return( ((crc >> 8) & 0x00FFFFFF) ^ crc_tab32[0xFF & (crc ^ data )] );
}
```

The CRC table is generated first as follows:

```
  CS_init_crc32_tab( 0x04C11DB7 ) for IEEE 802-3
```

a polynomial of
x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

and

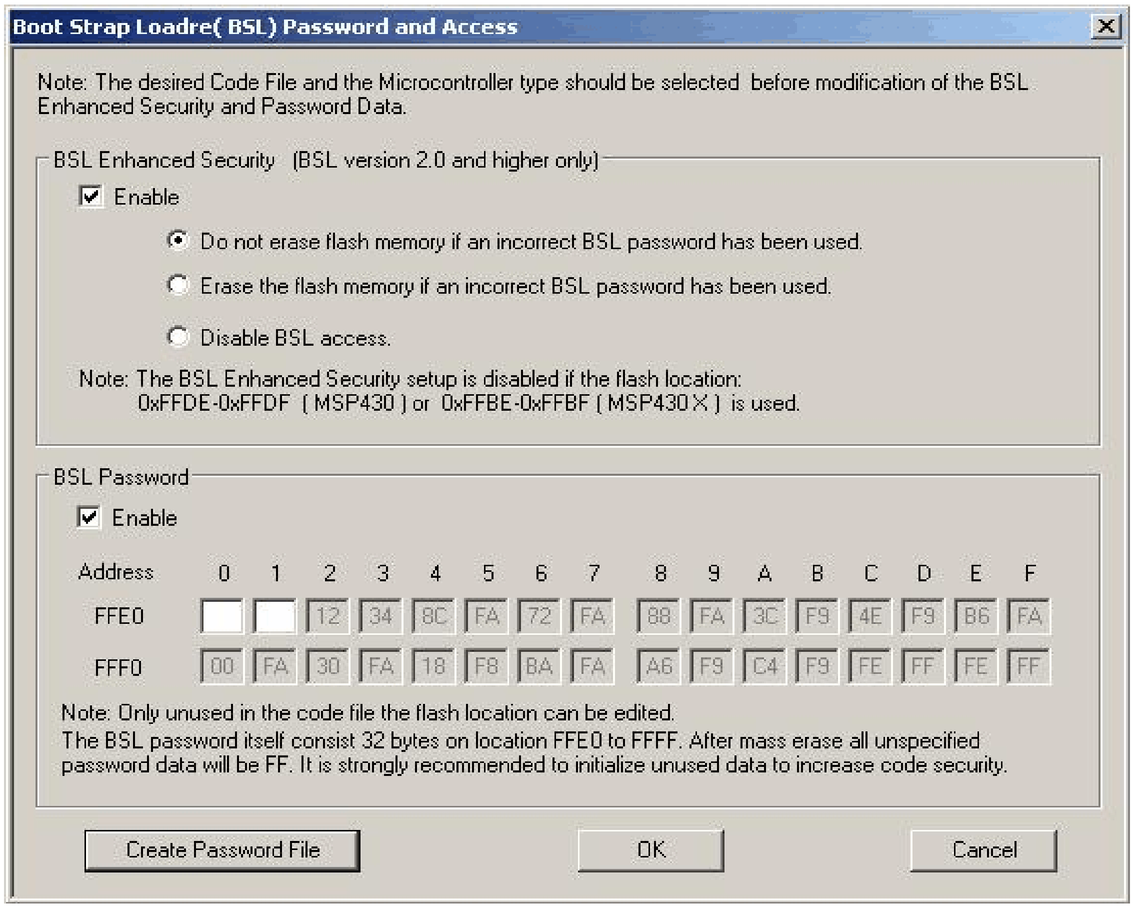
```
  CS_init_crc32_tab( CRC_def_POLY ) for CRC32 defined polynomial
```

where

```
void CS_init_crc32_tab( unsigned long poly_in )
{
  int n, k;
  unsigned long c, poly;

  poly = 0L;
  for (n = 0; n < 32; n++)
  {
      poly <<= 1;
      poly |= 1L & poly_in;
      poly_in >>= 1;
  }

  for (n = 0; n < 256; n++)
  {
    c = (unsigned long )n;
    for (k = 0; k < 8; k++)
      c = c & 1 ? poly ^ (c >> 1) : c >> 1;
    crc_tab32[n] = c;
  }
}
```

10. BSL Password and Access

The MSP430 bootstrap loader (BSL) enables users to communicate with the MSP430 even if the JTAG security fuse is blown. Access to the MSP430 memory via BSL interface is protected against unauthorized access by a user-defined password. The BSL password itself consist 32 bytes on location 0xFFE0 to 0xFFFF. This flash memory location is also used by the interrupt vector. If all interrupt location available in the MSP430 are used and specified, then the BSL password is used in fully and unauthorized access probability to the MSP430 is very low. But in a lot of application only part of the interrupt vector is defined. After mass erase all unspecified password data will be 0xFF and probability of the unauthorized access to the MSP430 becomes much higher. It is strongly recommended to initialize unspecified data in the interrupt vector to decrease probability of the unauthorized access to the MSP430.



Boot Strap Loadre(BSL) Password and Access

Note: The desired Code File and the Microcontroller type should be selected before modification of the BSL Enhanced Security and Password Data.

BSL Enhanced Security (BSL version 2.0 and higher only)

☑ Enable

[✓ Do not erase flash memory if an incorrect BSL password has been used.]

Erase the flash memory if an incorrect BSL password has been used.

Disable BSL access.

Note: The BSL Enhanced Security setup is disabled if the flash location:
0xFFDE-0xFFDF (MSP430) or 0xFFBE-0xFFBF (MSP430X) is used.

BSL Password

☑ Enable

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
FFE0			12	34	8C	FA	72	FA	88	FA	3C	F9	4E	F9	B6	FA
FFF0	00	FA	30	FA	18	F8	BA	FA	A6	F9	C4	F9	FE	FF	FE	FF

Note: Only unused in the code file the flash location can be edited.
The BSL password itself consist 32 bytes on location FFE0 to FFFF. After mass erase all unspecified password data will be FF. It is strongly recommended to initialize unused data to increase code security.

Figure 10.1

The ***BSL Password and Access*** dialogue (figure 10.1) allows to edit the undefined data located in the flash memory in location 0xFFE0 to 0xFFFF. In the ***BSL Password*** group all unused data can be specified. An access to particular flash location is disabled (grey field on the screen) if specified
data is defined in the code file. All unused in the code file locations between 0xFFE0 and 0xFFFF are enabled (white) and can be edited.

Note: The code contents always has a higher priority then an edited BSL data password. If the new code file is used and the same location is used in the code file and data specified in the ***BSL Password*** *dialogue screen, then the data specified in the* ***BSL Password*** *dialogue will be ignored.*

The ***Create Password File*** button allows to create BSL password file, then can be used in the future to unlock an access to programmed MSP430 devices.

The newest MSP430 microcontrollers with the BSL version 2.0 and higher have enhanced security features. These features are controlled by the Flash data word located below the interrupt vector e.g 0xFFDE for the MSP430 and 0xFFBE for the extended MSP430X . If this word contains:

0x0000:	The flash memory will not be erased if an incorrect BSL password has been received by the target. It is the same features like in all MSP430 with an older BSL version.
0xAA55:	The BSL is disabled. This means that the BSL communication can not be established.
All other values:	If an incorrect password is transmitted then the whole flash memory will be erased automatically, to protect unauthorized access to the MSP430 device.

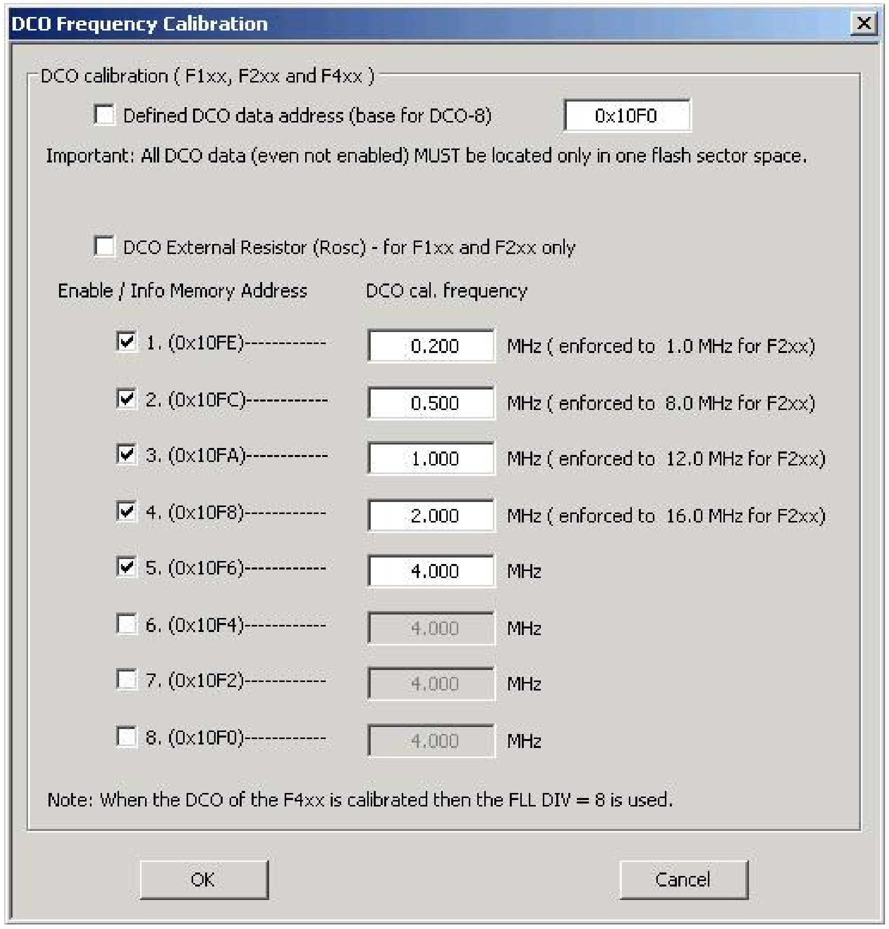
Desired option can be selected in the ***BSL Enhanced Security*** group of the ***BSL Password and Access*** dialogue. Option can be used only when the BSL version is 2.0 or higher.

11. DCO calibration

If an application software uses the DCO as a main clock in the MPS430Fxx microcontroller, then an execution time is related to the DCO frequency. When the DCO is not calibrated, then the DCO frequency variation from unit to unit can be +/-30% or more. See TI's data sheet for details. In the F2xx family the DCO frequencies have been calibrated in production process and calibration data saved in the Info Memory for the DCO frequencies 1MHz, 8MHz, 12MHz and 16 MHz. When the DCO calibrated data are used in the application software, then the DCO frequency tolerance is better then 1% from the nominal frequency.

FlashPro430 software allows to calibrate up to eight DCO frequencies in the MSP430F1xx, F2xx or F4xx to any value in the DCO range and saved calibrated DCO constants in the default INFO Memory locations 0x10F0 to 0x10FF or user defined memory location when the ***Defined DCO data address*** is selected. . The ***DCO Frequency Calibration*** dialogue screen (Figure 11.1) can be selected from pull down menu ***Setup->DCO calibration***.

When the selected DCO calibration data is enabled, then in the edited field should be specified the DCO frequency (in MHz) in the range 0.10 MHz up to 16 MHz. When the F1xx or F4xx MCU is used, then the maximum frequency can not be higher then 8 MHz.


DCO Frequency Calibration

DCO calibration (F1xx, F2xx and F4xx)

☐ Defined DCO data address (base for DCO-8) 0x10F0

Important: All DCO data (even not enabled) MUST be located only in one flash sector space.

☐ DCO External Resistor (Rosc) - for F1xx and F2xx only

Enable / Info Memory Address	DCO cal. frequency	
✓ 1. (0x10FE)-----------	0.200	MHz (enforced to 1.0 MHz for F2xx)
✓ 2. (0x10FC)-----------	0.500	MHz (enforced to 8.0 MHz for F2xx)
✓ 3. (0x10FA)-----------	1.000	MHz (enforced to 12.0 MHz for F2xx)
✓ 4. (0x10F8)-----------	2.000	MHz (enforced to 16.0 MHz for F2xx)
✓ 5. (0x10F6)-----------	4.000	MHz
☐ 6. (0x10F4)-----------	4.000	MHz
☐ 7. (0x10F2)-----------	4.000	MHz
☐ 8. (0x10F0)-----------	4.000	MHz

Note: When the DCO of the F4xx is calibrated then the FLL DIV = 8 is used.

Figure 11.1

See data sheets for details. When the F2xx is detected, then the DCO calibration data listed for the DCO 1 to 4 are hardcoded (as it is calibrated by TI) to frequencies 1, 8, 12 and 16 MHz. If other DCO frequencies are required, then the DCO 5 to 8 field can be used. The DCO frequency can be calibrated with an internal reference resistor (as default) or with an external resistor if the box ***"DCO external resistor Rosc"*** is selected.. The second option is available in the MSP430F1xx family.

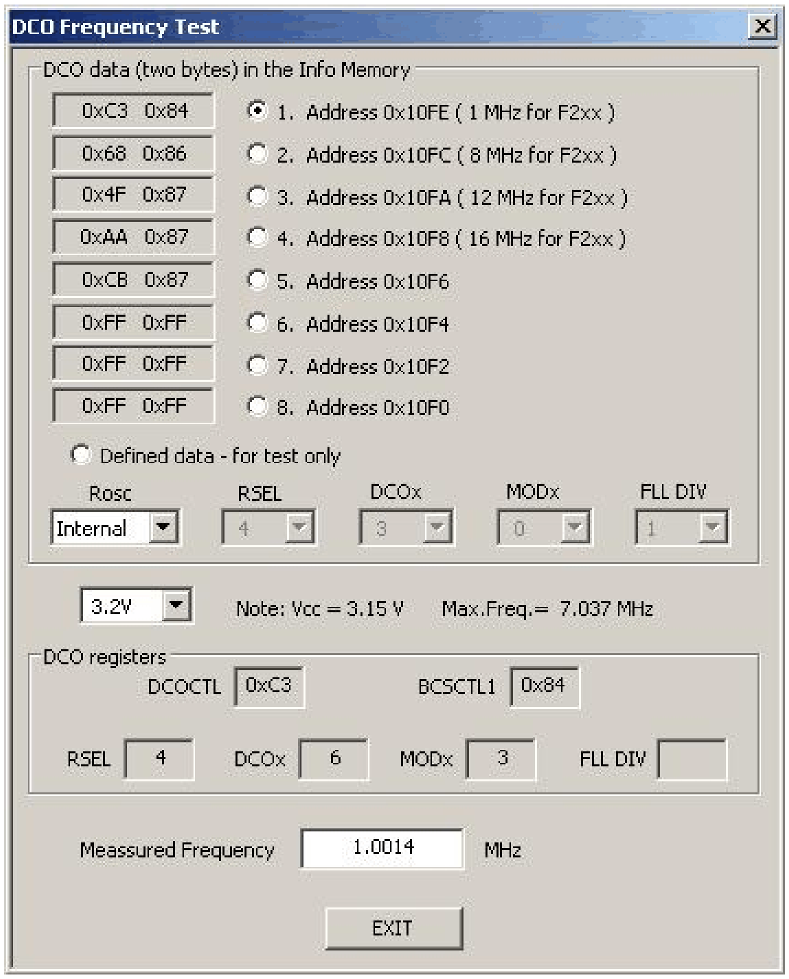
The DCO frequencies are calibrated with tolerance 1% when the ***Autoprogram*** or ***DCO Calibration*** button is pressed. Results is saved in the INFO memory.

Note: *The DCO frequency is calibrated for the Vcc used during the calibration procedure.*

The DCO calibration data can be verified using the tool from pull down menu ***Tools-> DCO Frequency Test*** (see figure 11.2). In the dialogue screen is possible to select the desired DCO calibration data and check the DCO frequency for selected DCO data from Info Memory. Current contents of the DCO registers (extracted from DCO calibrated Data) are displayed in the DCO registers fields.

The calibrated DCO frequency can be tested with Vcc from 2.2V to 3.6 V if target device is supplied from programming adapter, or for any Vcc if target device is supplied from an external Vcc.

When the ***Defined data - for test only*** is selected, then user can specify any DCO data manually and check the DCO frequency for a desired data.


DCO Frequency Test

DCO data (two bytes) in the Info Memory

0xC3 0x84	[✓ 1. Address 0x10FE (1 MHz for F2xx)]
0x68 0x86	2. Address 0x10FC (8 MHz for F2xx)
0x4F 0x87	3. Address 0x10FA (12 MHz for F2xx)
0xAA 0x87	4. Address 0x10F8 (16 MHz for F2xx)
0xCB 0x87	5. Address 0x10F6
0xFF 0xFF	6. Address 0x10F4
0xFF 0xFF	7. Address 0x10F2
0xFF 0xFF	8. Address 0x10F0

Defined data - for test only

Rosc	RSEL	DCOx	MODx	FLL DIV
Internal	4	3	0	1

3.2V Note: Vcc = 3.15 V Max.Freq.= 7.037 MHz

DCO registers
DCOCTL 0xC3 BCSCTL1 0x84
RSEL 4 DCOx 6 MODx 3 FLL DIV

Meassured Frequency 1.0014 MHz

Figure 11.2

12. Script File - defined programming sequence

Programming sequence can be customized when is using a script file. Script file prepared as a text file (using any editor like **notepad**) can contains customized programming sequences in any order. Generally, all buttons available on the main dialogue screen can be used in the script file. All other options available on others screens like memory options, serialization type etc. can not be modified from the script file directly, but can be reloaded in fully using configuration file. From the script file any configuration files can be called at any time that allows to modify programmer configuration. This method can simplify programming process using script file and allows to use full options available in the programmer. Programming sequence conditions can be taken from user defined procedures attached as an independent DLL if required.

Programmer has two entry for taking the sequence from the script file

1. By pressing the Script File button in the Main dialog
2. By using the -rf with the executable file

12.1 Script button

The ***'Script'*** button is the dynamically programmable device action button that allows to take a desired action taken from the script file. The ***Script*** button has a name ***Script File - none*** (Figure 12-1) if the script file is not defined or ***Script:*** with used file name when the script file is active (Figure 12-2). When the ***Script*** button is pressed and the current script file is not active, then the ***Open File*** dialog is displayed and the desired script file should be selected. When the ***Script file*** button is not empty and the new script file if required, then the new file can be selected from the pull down menu - ***File-> Open Script File***.

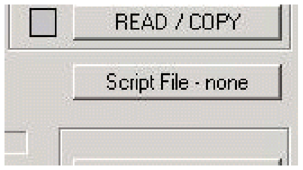


Figure 12-1

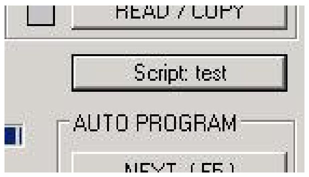


Figure 12-2

The ***Script*** button is very useful for implementing a short programming sequence not present directly in the ***Device Action*** group buttons Below is an easy script file used for downloading two independent codes to target device - first code used for

hardware test if possible, and when hardware is ok, then the second code is downloaded as the final code to target device. The same sequence can be used with other buttons, but sequence should be always repeated, that of course is not convenient.
Using the notepad editor create the script file and save it eg. as the file ***"test.sf"*** or any other file name. See this chapter below for all available instructions that can be used in the script file.

```
;---------------------------------------------------------------------------
; easy script file;
;---------------------------------------------------------------------------

 LOADCFGFILE C:\Program Files\Elprotronic\MSP430\USB FlashPro430\test.cfg
 LOADCODEFILE C:\Program Files\Elprotronic\MSP430\USB FlashPro430\test.cfg
 AUTOPROGRAM

; now the hardware is tested according to downloaded firmware
  MESSAGEBOX    YESNO
   "Press YES when the test finished successfully."
   "Press NO  when the test failed."

  IF BUTTONNO GOTO finish

 LOADCFGFILE C:\Program Files\Elprotronic\MSP430\USB FlashPro430\final.cfg
 LOADCODEFILE C:\Program Files\Elprotronic\MSP430\USB FlashPro430\final.cfg
 AUTOPROGRAM

>finish
 END
;---------------------------------------------------------------------------
```

When the script file above is used then the first configuration file and the fist code file is downloaded and Autoprogram function is executed. When finished then the MCU firmware started (make sure that the first configuration allows to start the code when the Autoprogram is finished). Final code is downloaded when the test has been finished successfully.

Before running the script file the configuration files named ***test.cfg*** and ***final.cfg*** required in the project should be created using the GUI software first. To do that connect target devices to programming adapter, select desired configuration and save the configuration file as ***test.cfg*** and create final configuration file in similar way.

12.2 Script file option

Programming sequence can be customized when using the -rf with the executable file (described in the ***"Project and Configuration Load/Save"*** chapter) .

When the executable file USB-MSP430-Prg.exe is called with a script path as an argument e.g.

USB-MSP430-Prg.exe -rf C:\Program Files\Elprotronic\USB FlashPro430\script.txt

or when the icon with the USB-MSP430-Prg.exe and script file path is executed then programmer starts automatically programming sequences according to procedure specified in the script file. Access to other buttons are blocked. When script file sequence is finished then program is terminated. There is not option to modify the running sequence when script sequence is used. This option is useful in production, because nobody can modify sequence that has been prepared for the production purpose.

12.3 Script commands

LIMITATIONS:

1. Up to 1000 script lines commands can be used. Empty lines and lines with comments only are ignored and not counted.
2. Up to 50 CALL's deep stack is used (CALL in CALL in CALL......).

SYNTAX:

white spaces before instructions, labels etc are ignored.

; comment - all contents after semicolon are ignored.

NOTE: Comment can not be used in the lines where the file name is specified.

>label - character '>' without spaces must be placed before label name.

NOTE: After label can not be specified any command in the same line. Line can contain label only.

LIST OF INSTRUCTIONS:

MESSAGE	- message declaration ,that can be saved to file if required
" message - line -1 "	- Text.
" message - line -2 "	- Each line contents must be located between characters " "

“ max up to 50 lines “	- Number of content lines - up to 50 lines.
SAVEMSG filename	- save message created in MESSAGE to specified file
APPENDMSG filename	- append message created in MESSAGE to specified file
SAVEREPORT filename	- save message from the GUI report window to specified file
APPENDREPORT filename	- append message from the GUI report window to specified file
GUIMSGBOX ENABLE	- display the message boxes (warning, errors etc) created by GUI enabled
GUIMSGBOX DISABLE	- display the message boxes (warning, errors etc) created by GUI disabled
IFGUIMSGBOXPRESS OK	- apply button OK / YES when the message box created by GUI is generated, but disabled to be displayed.
IFGUIMSGBOXPRESS CANCEL	- apply button CANCEL / NO when the message box created by GUI is generated, but disabled to be displayed.
MESSAGEBOX type **FCTEXT**	- pop-up message box with buttons.
	- message taken from the FCONTROL function (User’s DLL)
MESSAGEBOX type	- pop-up message box with buttons.
“ message - line -1 “	- Text displayed in message box.
“ message - line -2 “	- Each line contents must be located between characters “ ”
“ max up to 50 lines “	- Number of content lines - up to 50 lines.
Message box type list	
OK	- One button OK
OKCANCEL	- Two buttons OK , CANCEL
YESNO	- Two buttons YES , NO
YESNOCANCEL	- Three buttons YES , NO, CANCEL
GOTO label	
CALL label	- CALL procedure.
RETURN	- return from CALL.
IF condition **GOTO** label	
IF condition **CALL** label	
condition list:	
BUTTONOK	- if button **OK** pressed in the message box.
BUTTONYES	- if button **YES** pressed in the message box.
BUTTONNO	- if button **NO** pressed in the message box.
BUTTONCANCEL	- if button **CANCEL** pressed in the message box.
DONE	- if selected process e.g. **AUTOPROGRAM**, **Read File** etc. finished successfully.
FAILED	- if selected process e.g. **AUTOPROGRAM**, **Read File** etc. failed.
CONTROL = number	- if status from the FCONTROL function = NUMBER

FCONTROL type argument	- call the external function from FxControl DLL
PAUSE number	- pause in miliseconds - 1 to 100000 range (1ms to 100 s).
OPENDLLFILE filename	- FxControl DLL file - Full path and DLL File name.
LOADCFGFILE filename	- Configuration file - Full path and File name.
LOADCODEFILE filename	- Code file - Full path and File name.
LOADPASSWFILE filename	- Password file - Full path and File name.
LOADSNFILE filename	- File with Serial Number list - Full path and File name.
VCCOFF	- Turn OFF Vcc from programming adapter to target device.
VCCON	- Turn ON Vcc from programming adapter to target device.

Note: Vcc from FPA must be enabled first using configuration file.

RESET	-equivalent to pressed button RESET on the main dialogue screen.
AUTOPROGRAM	-equivalent to pressed button AUTOPROGRAM on the main dialogue screen.
VERIFYFUSE	-equivalent to pressed button VERIFY SEC. FUSE on the main dialogue screen.
VERIFYPASSWORD	-equivalent to pressed button VERIFY PASSWORD on the main dialogue screen.
ERASEFLASH	-equivalent to pressed button ERASE FLASH on the main dialogue screen.
BLANKCHECK	-equivalent to pressed button BLANK CHECK on the main dialogue screen.
WRITEFLASH	-equivalent to pressed button WRITE FLASH on the main dialogue screen.
VERIFYFLASH	-equivalent to pressed button VERIFY FLASH on the main dialogue screen.
READFLASH	-equivalent to pressed button READ/COPY on the main dialogue screen.
READSN	-equivalent to pressed button READ SN on the main dialogue screen.
BLOWFUSE	-equivalent to pressed button BLOW FUSE on the main dialogue screen.

Note: If the **BLOWFUSE** command is used then the blow security fuse will be processed even if the "**Blow Security Fuse enable**" option is disabled. That allows to use command **AUTOPROGRAM** with disabled blow security fuse option and on the end call the function **BLOWFUSE** (if required) without modifying the configuration setup.

TRACEOFF	- trace OFF.
TRACEON	- trace ON and saved in the "Trace-Scr.txt" file in current working directory.

Option useful for debugging. Trace file contains sequence of all executed commands from script file in the run time. On the left side of all lines the current line numbers correspondent to the line number in the script file are printed. Line numbers are counted without empty lines and without lines contains comments only.

END	- end of script program.

Programming sequence conditions can be taken from user defined procedures attached as an independent DLL and called in the script as a function.

FCONTROL type argument	- call the external function from FxControl DLL

Function should be created using Visual C++ and attached to FlashPro430 software. When the DLL is created then the full path and name of the used DLL should be specified in the script file. In the script file the name of the desired DLL can be specified on-line few times. This means that more then one DLL can be used in the programming sequence, but only one DLL at the time. When the new DLL file is open, then the old DLL file is closed at the same time. One function is used in the user defined DLL

_int32 F_Control(**_int32** type, **_int32** argument, **char** * message);

Parameters type and argument are specified in the script file and are transferred from the programming software to DLL. Status from F_Control and message are transferred from DLL to programming software.
Programming software package contains the source code of the user defined DLL. Package has been prepared using MS Visual C++.net package. Source code is located in directory

C:\Program Files\Elprotronic\FxControl-DLL

User defined function should be inserted in empty place inside the FxControl.cpp file and recompiled. Recompiled file FxControl.dll ready to be used will be located in directory

C:\Program Files\Elprotronic\FxControl-DLL\release

DLL file can be renamed to any file name and name and specified in the script file via command
OPENDLLFILE filename

Below is an easy script file contents that allows to create following sequence;

1. Vcc supplied to target device is turn-OFF and first message box with buttons OK/CANCEL is displayed. Programmer is waiting until button OK or CANCEL is pressed.
2. When confirmed, then first configuration file **test-A.cfg** is downloaded to programmer. Configuration file **test-A.cfg** should be prepared first using programming software with desired configuration, selected desired code file etc. Programmer's configuration should be saved using **"Save setup us .."** option.

3. When test code is downloaded and processor started (if enabled in **test-A.cfg** file) then message box is displayed and software is waiting until button YES / NO is press. Meantime manual target's device test can be done. If test is positive, then button OK should be pressed. Or button NO if test failed.
4. When button OK has been pressed then programmer downloads **finalcode.cfg** configuration file to programmer. Current configuration can activate serialization if required, reload final code to be downloaded etc. When the new configuration is reloaded then final code is downloaded to target device, serialization is created etc.
5. On the end programmer returns to beginning and waiting for the next target device to be connected.

```
;=================================================================
;      Script file - demo program - without DLL file
;-----------------------------------------------------------------
>START
  VCCOFF
  MESSAGEBOX    OKCANCEL
    "VCC if OFF now. Connect the test board."
    "When ready press the button:"
    " "
    "OK     - to test the board"
    "CANCEL - to exit from program"

  IF BUTTONCANCEL GOTO finish
  LOADCFGFILE   C:\Elprotronic\Project\Cpp-Net\USB-MSP430Prg\test-A.cfg
  MESSAGEBOX    OK
   "Press OK to download the test program."
  AUTOPROGRAM

  MESSAGEBOX    YESNO
   "Press YES when the test finished successfully."
   "Press NO  when the test failed."

  IF BUTTONNO GOTO START

  LOADCFGFILE   C:\Elprotronic\Project\Cpp-Net\USB-MSP430Prg\finalcode.cfg
  AUTOPROGRAM
  GOTO START

>finish
  END
;=================================================================
```

Script below allows to start programmer, download configuration file, open the code file and program target device. When finished, then report (failed or pass) is saved into the file. File contents can be serviced by the external program. The GUI popup messages are disabled.

```
;=====================================================
;            Script file - demo program
;  Program MCU and exit. Save report in the file
;-----------------------------------------------------
  GUIMSGBOX     DISABLE   ;select Disable or Enable - remove or add comment ';'
 ; GUIMSGBOX     ENABLE

  IFGUIMSGBOXPRESS   CANCEL         ;press CANCEL if GUI box is generated,
                                    ;but disabled to be displayed,
 ; ifGuiMsgBoxPress   cancel        ;This is OK also. Commands are not case
                                    ;sensitive

  LOADCFGFILE   test-script.cfg     ;recommended full path and name
  IF FAILED GOTO fileerror

  LOADCODEFILE  test_1k.txt         ;recommended full path and name
  IF FAILED GOTO fileerror

  AUTOPROGRAM

  IF DONE CALL testOK
  IF FAILED CALL testFailed

            ;extra message taken from report window added to file
            ;Can be saved on the same file or other file
  APPENDREPORT tmp_file.txt        ;recommended full path and name
  GOTO finish

>testOK
  MESSAGE
    "1  Test OK"              ;min 1 line, max 50 lines
    "2-nd line"               ; -optional
  GOTO      saveMsg

>testFailed
  MESSAGE
    "0  Test Failed"          ;min 1 line, max 50 lines
    "2-nd  line"              ; -optional
    "etc "                    ; -optional

>saveMsg
```

```
  SAVEMSG  tmp_file.txt       ;recommended - full path and name
  RETURN

>fileerror
  MESSAGE
    "Config or Code file open error."    ;min 1 line, max 50 lines
    "Program terminated."
  SAVEMSG  tmp_file.txt       ;recommended - full path and name
    ;end exit

>finish
  END
;==========================================================================
```

Below is the next script file examples uses DLL file that allows to control testing process via function written in the DLL. Functionality is the same as in the example above, but instead manually confirmation of the test result the result is taken automatically from the DLL function. Two functions has bee used for this purpouse

`FCONTROL` - calls external user defined function in the DLL

`IF CONTROL = 0 GOTO START` - test status from the `FCONTROL` and if result is 0 (FALSE) then procedure returns to start.

Required DLL file should be created first.

```
;=========================================================
;     Script file - demo program - with DLL file
;---------------------------------------------------------
 OPENDLLFILE  C:\Program Files\Elprotronic\FxControl-DLL\release\FxControl.dll
>START
  VCCOFF
  MESSAGEBOX    OKCANCEL
    "VCC if OFF now. Connect the test board."
    "When ready press the button:"
    " "
    "OK     - to test the board"
    "CANCEL - to exit from program"

  IF BUTTONCANCEL GOTO finish
  LOADCFGFILE   C:\Elprotronic\Project\Cpp-Net\USB-MSP430Prg\test-A.cfg
  MESSAGEBOX    OK
   "Press OK to download the test program."

  AUTOPROGRAM
  FCONTROL 1 0                ;type 1, argument 0, but can be any
  IF CONTROL = 0 GOTO START   ;when false (0), return to start
  IF BUTTONNO GOTO START
  LOADCFGFILE   C:\Elprotronic\Project\Cpp-Net\USB-MSP430Prg\finalcode.cfg
  AUTOPROGRAM
  GOTO START
>finish
  END
;==================================================================
```

13. Project and Configuration Load / Save

Programming software can save configuration settings in the configuration files or save the whole project configuration with used code contents and save it in the encrypted project file . This allows the user to create several configuration or project fils, 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.

13.1 Load / Save Setup

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.

13.2 Load / Save Project

The Project option (Save/Load) contains more then the programmer configuration only, but can also the code and the BSL password used in the project. Contents of the project file is encrypted, so it is not possible to read the contents of the used code downloaded to target device. When the project is opened then the same decryption key must be used as it was used in the encryption process, otherwise decryption will not succeed. Encryption key depends from the used type of software (FlashPro430, GangPro430, etc.) used password or destination's PC "hardware fingerprint" number. So - the project file created with the FlashPro430 software cannot be used with the FET-Pro430 or GangPro430 and vice-verse. Each project file should be create in the same type of software. Project file is CRC protected and CRC check is performed when the file is loaded .

Project can be unprotected or protected with the destination PC “hardware fingerprint” number or password protected. This allows to create the project that can be used only on the specific PC when the project is encrypted with the destination PC “hardware fingerprint” number (useful in production) or create the project that can be used only when the correct password is entered every time when the project is open. Project can be unlocked or locked with almost all blocked buttons and pull down menu items. When the project is locked, then only major buttons like ***Autoprogram*** or ***Verify*** are active - and only a few pull-down menu items are accessible. All options that allows to read the code contents are blocked.

When the new project is create then it is recommended to select the ***New Setup*** from pull down menu and set the default option of all parameters and names used in the programmer. As the next - the desired processor, code file, password file if required and all desired option (see all available options described in this manual) should be selected. When it is done, it should be verified if programmers works as expected. When all works, then the current setup can be saved as the project file. Select the ***Save Project as..*** from **File** pull down menu. Following dialogue will be displayed (Figure 13.2-1) that allows to select desired project option

Following options can be selected:

Project protection:

Any PC - not protected.

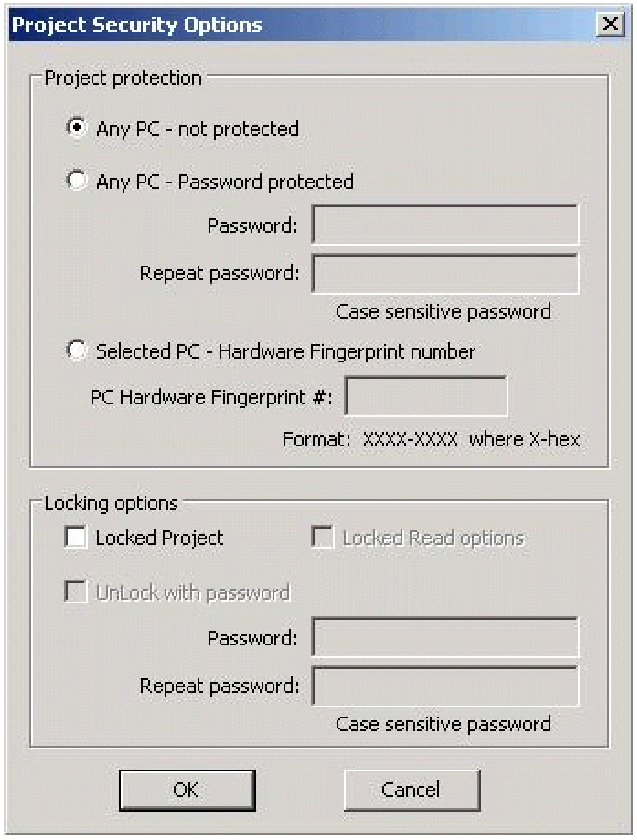
When this option is selected then project is not protected and can be opened on any PC without restrictions.

Any PC - Password protected.

When this option is selected then project can be opened when the password is correct. The desired password should be entered in two edit lines. Password is case sensitive and takes up to 16 characters - space including.

Selected PC - Hardware Fingerprint

When this option is selected then project can be opened only on one desired PC where the ***PC's "hardware fingerprint"*** number taken from the destination PC is the


Project Security Options

Project protection

(● Any PC - not protected)

○ Any PC - Password protected

Password:

Repeat password:

Case sensitive password

○ Selected PC - Hardware Fingerprint number

PC Hardware Fingerprint #:

Format: XXXX-XXXX where X-hex

Locking options

☐ Locked Project ☐ Locked Read options

☐ UnLock with password

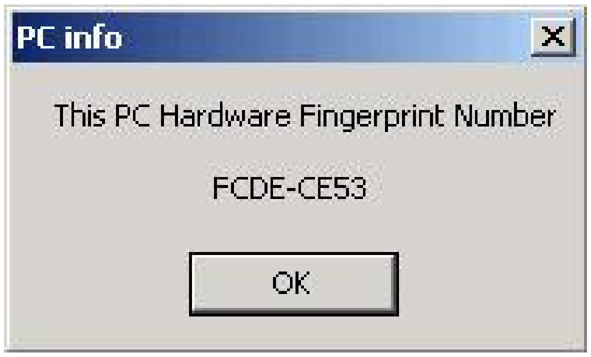
Password:

Repeat password:

Case sensitive password

Figure 13.2-1

same as the number used when the project has been created. This option is useful in


PC info

This PC Hardware Fingerprint Number

FCDE-CE53

Figure 13.2-2

production because project can be opened automatically without password on the desired PC. The same project file cannot work on other computers. When the project is created for particular PC, then the ***PC "hardware fingerprint"*** number should be taken from the desired PC and entered in the edit line in dialogue screen (figure 13.2-1). This number has hardcoded format and contains eight hex characters with dash between 4th and 5th character eg.

6FA4-E397

Notice, that the project created with the desired ***PC's "hardware fingerprint"*** number will not work on the PC where the project has been created, because ***"hardware fingerprint"*** numbers on the destination PC and the PC used for creating a projet are not the same. It is possible to create the project with the ***PC's "hardware fingerprint"*** number taken from his own PC, create a project and check if work as expected. When all is OK, then project should be saved again with the desired ***PC's "hardware fingerprint"*** number.

PC's "Hardware fingerprint" number used with the project can be read by selecting the ***"PC Hardware fingerprint number"*** option from pull down menu

About/Help -> PC Hardware fingerprint number

Following message box is displayed when the option above is selected (figure 13.2-2)

Locking option:

Locked Project

1. When not selected, then project is not locked. All contents can be modified and all buttons are accessible.
2. When selected then project is locked. Almost all buttons are disabled (grayed) and almost all items in the pull down menu are disabled.
   When the project is locked, then it is possible to select - permanently lock project, or select an option that it is possible to unlock the project under password. The unlock password can be not the same as the password used for opening the project.

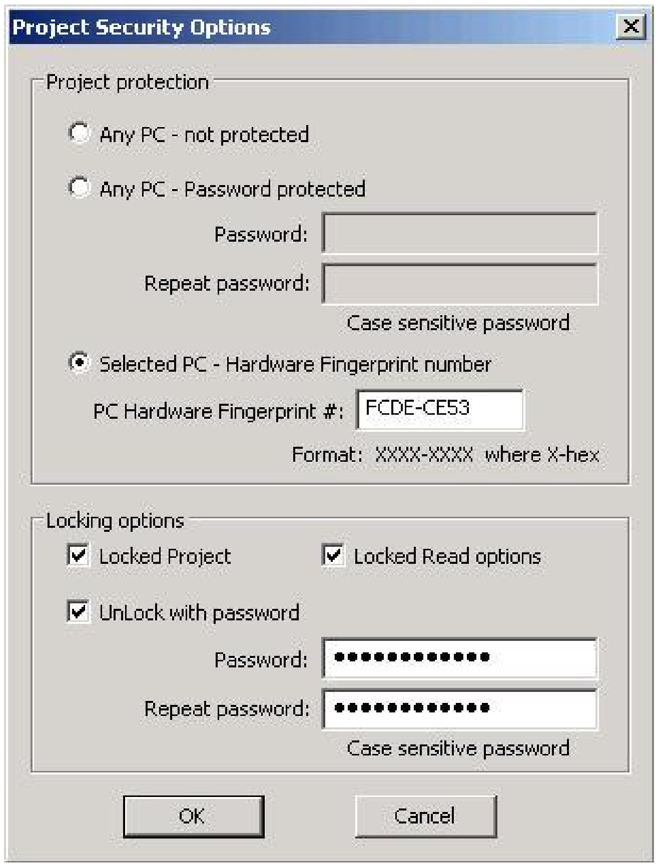
Locked Read options

When selected then the code viewers and READ button are blocked and not allows to read the code contents downloaded to target device. If the security fuse is blown after programming the target device, then code cannot be seen by the staff downloading code to target devices.

Unlock with password

When project is locked then it is possible to select option "unlock with password" and specify up to 16 characters unlocking password. Password is case sensitive.
On the figure 13.2-3 is a "Project Security Options" dialogue screen with selected options

Project protected with ***PC's "hardware fringerprint"*** number, locked and unlocked with password.


Project Security Options

Project protection

( ) Any PC - not protected

( ) Any PC - Password protected

Password:

Repeat password:

Case sensitive password

(●) Selected PC - Hardware Fingerprint number

PC Hardware Fingerprint #: FCDE-CE53

Format: XXXX-XXXX where X-hex

Locking options

☑ Locked Project ☑ Locked Read options

☑ UnLock with password

Password: •••••••••••••

Repeat password: •••••••••••••

Case sensitive password

Figure 13.2-3

By default, project is not protected and not locked. This allows to create unprotected project and open it at any time on any PC without restrictions. All buttons and items on the dialogue screen are not blocked.

13.3 Commands combined with the executable file

Programming executable file can be opened with project file having extension **FP430prj** eg.

USB-MSP430-Prg.exe test.FP430prj

The Folder option in Windows can also be configured to open the programming executable file (USB-MSP430-Prg.exe) when the file with extension **FP430prj** is selected. That allows to easy and fast opening the Flash Programmer with configuration taken from the *.FP430prj file.

Project file or configuration setup file (or Code / Password file) can be opened using ***Load Setup*** (***Load Code / Password File***) option from ***File*** menu or can also be opened using command line combined with the executable file name. Following command line switches are available

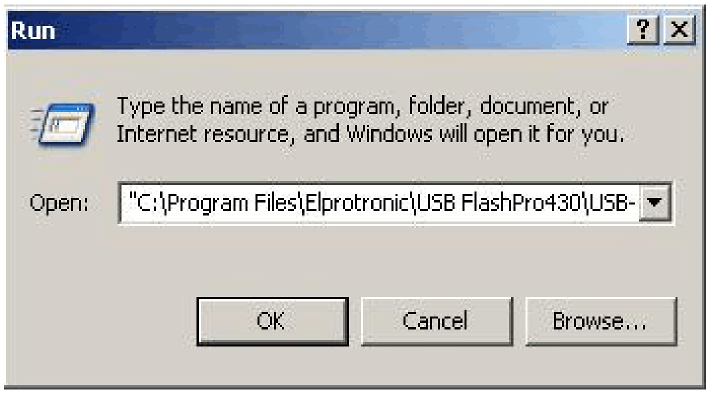
-prj Project file name (Open Project file)
-sf Setup_file_name (Open Setup file)
-cf Code_file_name (Open Code file)
-pf Password_file_name (Open Password file)
-nf SN_ file_name (Open Serial number list file)
-rf Script_file_name (Run programming sequence from the Script File)
-lock

Note: *When the **-cf** option is used, then code file name saved in the setup file (configuration file) is ignored and code file name specified with key **-cf** is used. Also when the **-pf** option is used, then password file name saved in the setup file (configuration file) is ignored and password file name specified with key **-cf** is used.*
*When the **-prj** option is used, then the **-sf, -cf, -pf, -rf** options are ignored.*

Using Windows ***START*** button (left bottom) select ***Run..*** Using ***Browse..*** find and select executable file (see Figure 13.3-1)

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

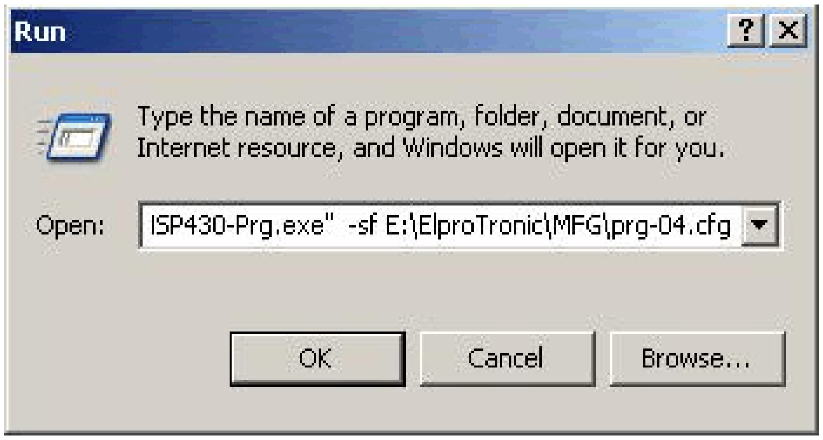
and at the end enter the required key with name of the setup file eg.


Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
OK
Cancel
Browse...

"C:\Program Files\Elprotronic\USB FlashPro430\USB-

Figure13.3-1

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


Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: ISP430-Prg.exe" -sf E:\ElproTronic\MFG\prg-04.cfg

Figure 13.3-2

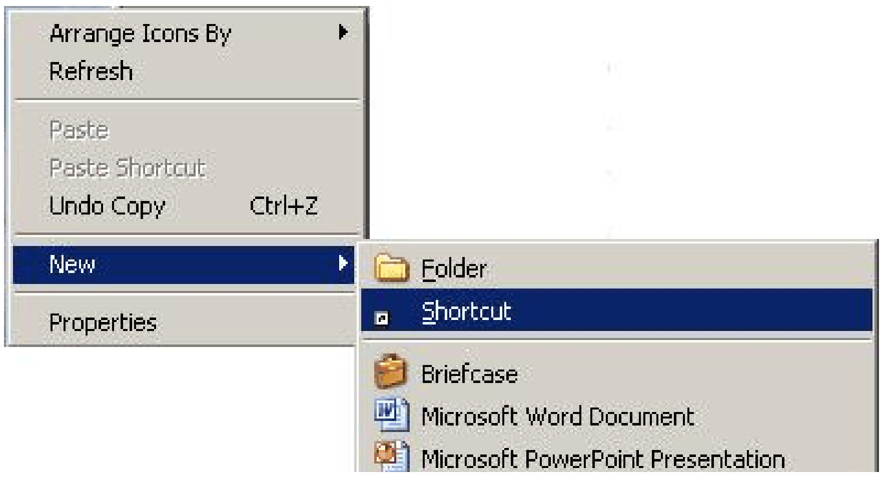
To fully lock the configuration setup the extra key “-lock” can be added in the command line eg.

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

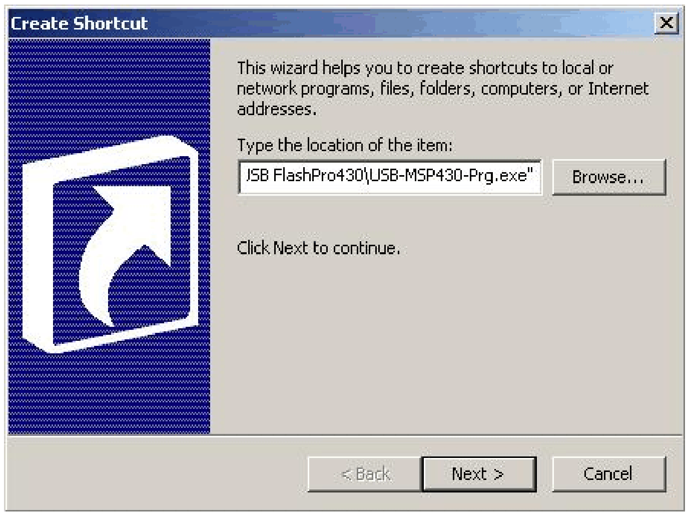
or

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

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 13.3-3) Using Browse.. in the Create Shortcut dialogue box select the following executable file


Arrange Icons By
Refresh
Paste
Paste Shortcut
Undo Copy Ctrl+Z
New
Properties
Folder
Shortcut
Briefcase
Microsoft Word Document
Microsoft PowerPoint Presentation
Figure 13.3-3

"C:\Program Files\Elprotronic\USB FlashPro430\USB-MSP430-Prg.exe"
(see Figure 13.4) and at the and add the required command keys (see Figure 13.5) eg.


Create Shortcut

This wizard helps you to create shortcuts to local or network programs, files, folders, computers, or Internet addresses.

Type the location of the item:

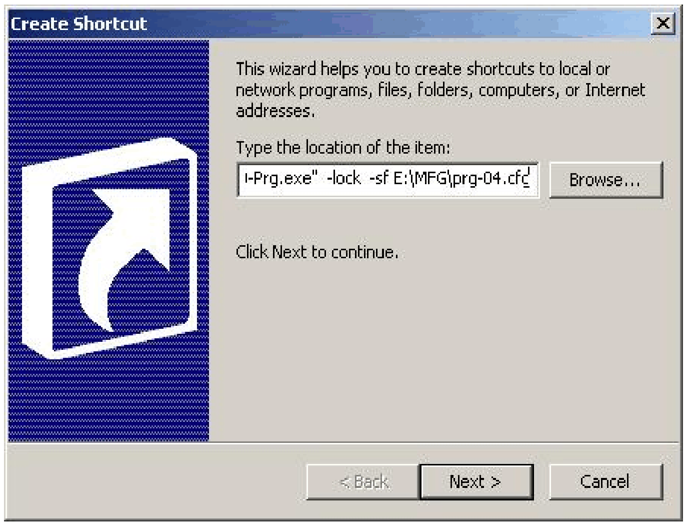
JSB FlashPro430\USB-MSP430-Prg.exe"

Click Next to continue.

Figure 13.3-4

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

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.


Create Shortcut

This wizard helps you to create shortcuts to local or network programs, files, folders, computers, or Internet addresses.

Type the location of the item:

I-Prg.exe" -lock -sf E:\MFG\prg-04.cfc

Click Next to continue.

Figure 13.3-5

14. 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/SBW/BSL Interface and DB-25 male connector, as shown in figure 14-1.

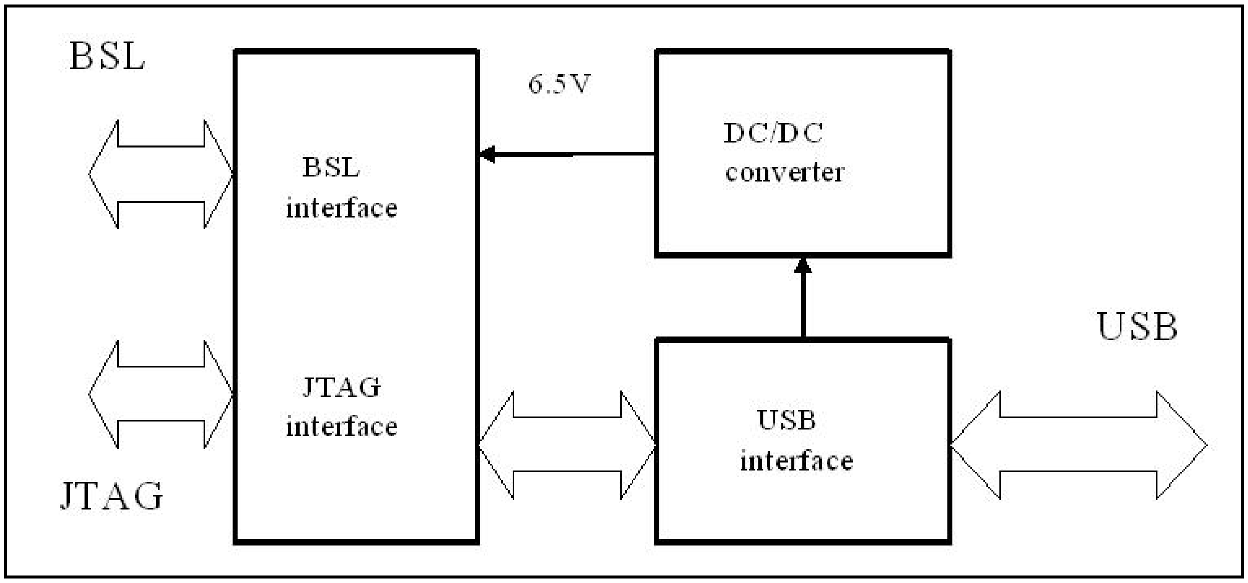


Figure 14-1

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 to100mA 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 it's 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 100 ohms to 470 ohms. Input impedance of the BLTx-In and TDO-In is higher then 100 kΩ. Target board input resistance of the RST, BLRX, TEST, TDI, TMS and TCK inputs should be at least 10 kΩ. Output impedance of the BLTX and TDO output should not exceed 1kΩ.

I/O schematic of the JTAG/SBW/BSL connections in the FlashPro430 is shown on Figure 12-2 (USB-MSP430-FPA-2.0) and 12-3 (USB-FPA-4.x).

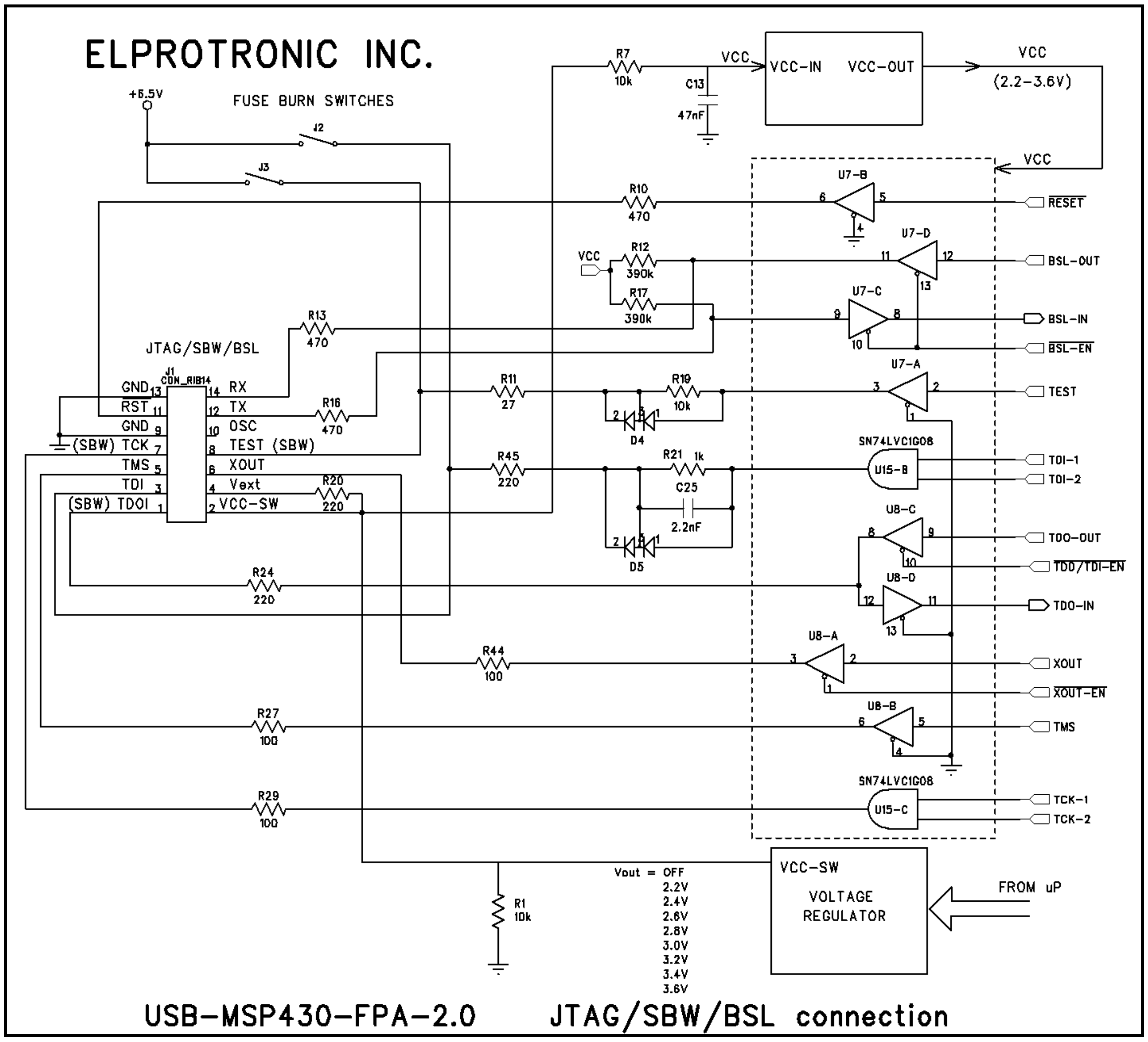
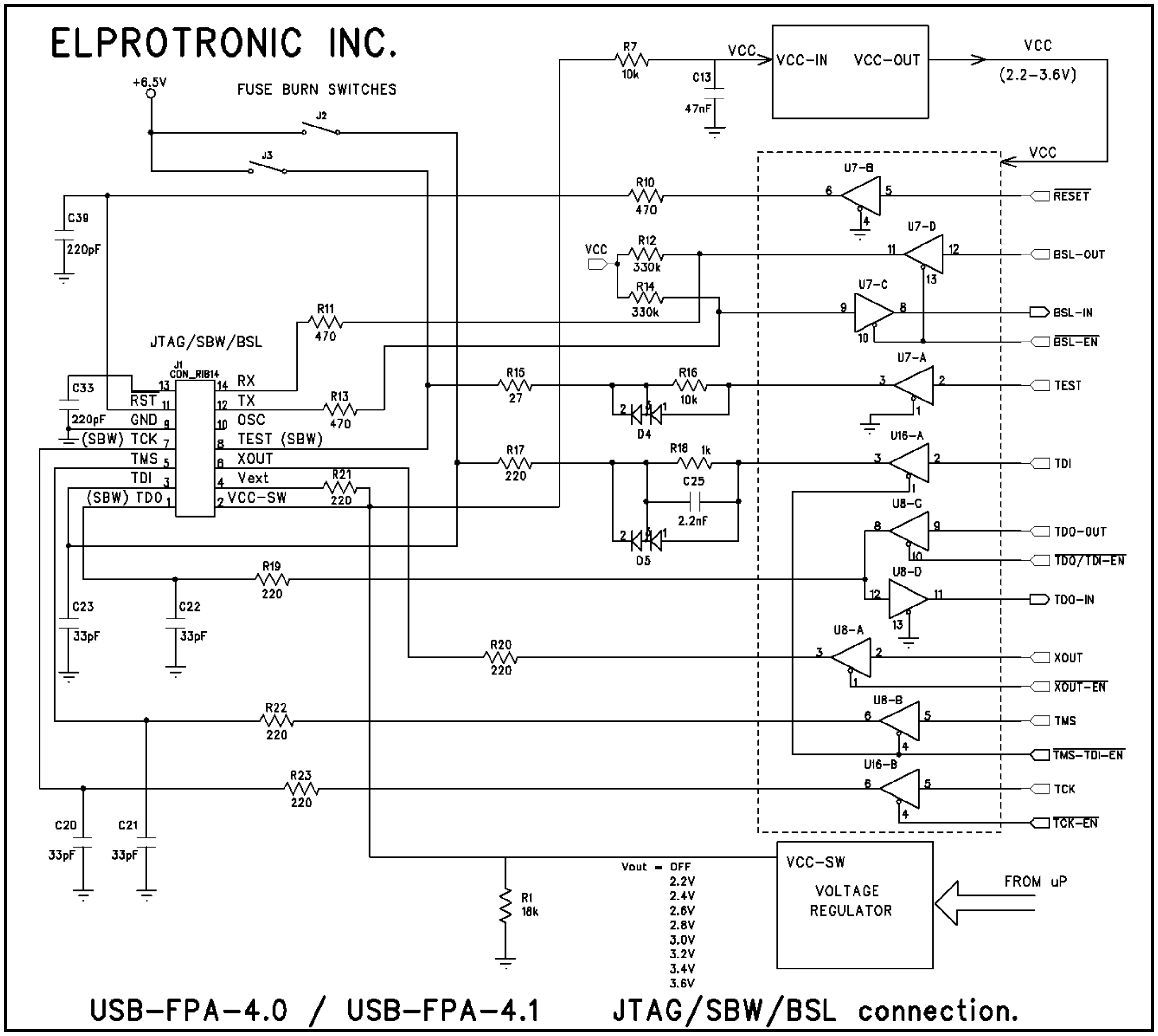


Figure 14-2



ELPROTRONIC INC.
FUSE BURN SWITCHES
JTAG/SBW/BSL
VCC-IN
VCC-OUT
VCC (2.2-3.6V)
RESET
BSL-OUT
BSL-IN
BSL-EN
TEST
TDI
TDO-OUT
TDO/TDI-EN
TDO-IN
XOUT
XOUT-EN
TMS
TMS-TDI-EN
TCK
TCK-EN
VCC-SW VOLTAGE REGULATOR
FROM uP
Vout = OFF 2.2V 2.4V 2.6V 2.8V 3.0V 3.2V 3.4V 3.6V
USB-FPA-4.0 / USB-FPA-4.1 JTAG/SBW/BSL connection.

Figure 14-3

15. Target connection

The Fast USB MSP430 Flash Programmers with the JTAG/SBW and the BSL Interfaces use the **STANDARD 14-pin TI-JTAG** connector's pinout to facilitate the JTAG/SBW 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/SBW interface connector and for the Bootstrap Loader (BSL) connector. The JTAG/SBW 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/SBW connector. We can notice that the TI-JTAG/SBW 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

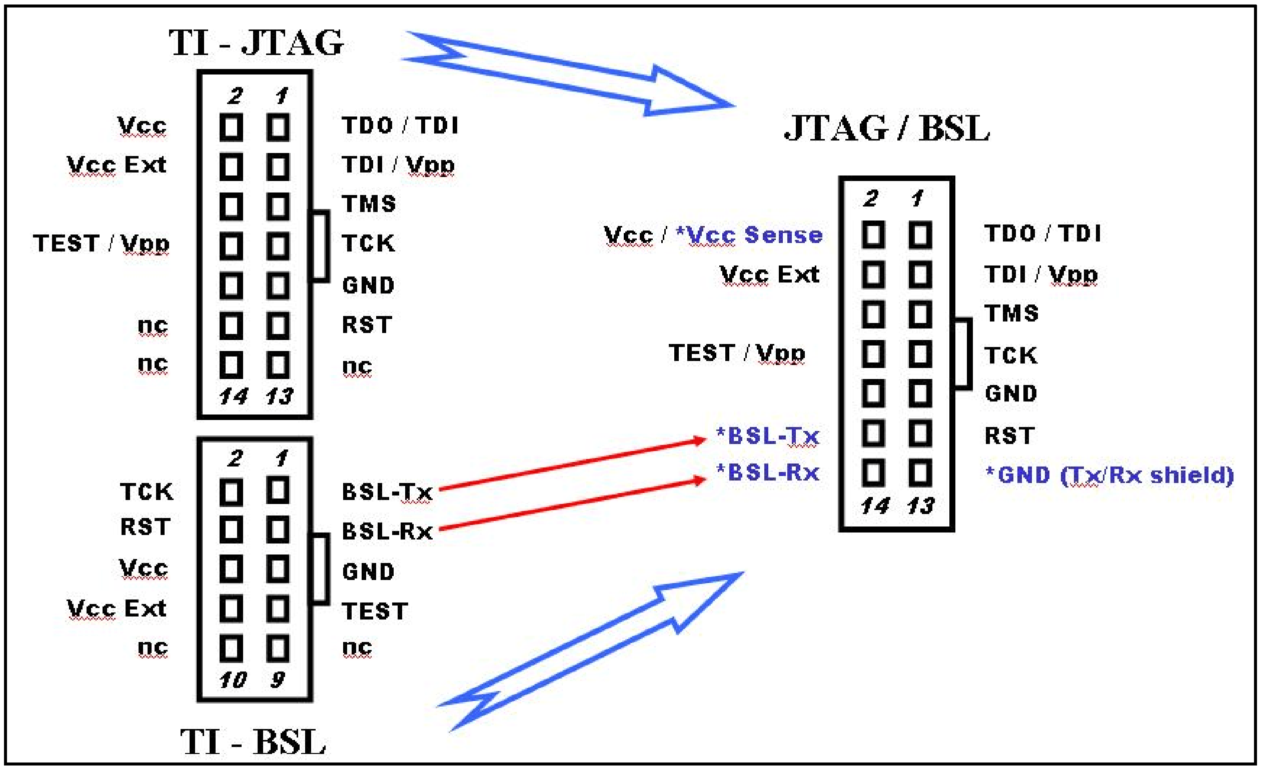


Figure 15-1. JTAG/BSL header connector - solder side.

utilizing these unused pins enables us to facilitate both the JTAG/SBW and the BSL communication interfaces on a SINGLE 14-pin JTAG/SBW connector.

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

The pinout for the standard JTAG/SBW connector with added BSL-Tx and BSL-Rx signals is shown in figure 15-1. 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/SBW signal lines are connected to pins number 1 through 11 in compliance with the standard JTAG/SBW specification provided by Texas Instruments.
The definition of all the pins is given in the tables 15-1.

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.*

Table 15.1 JTAG/BSL Interface connector

Pin #	Name	Description
1 (Red)	TDO/TDI - **SBWTDIO**	Data output / input, /**Spy-Bi-Wire** data I/O
2	VCC-In / Sense	Vcc supplied to the target (3.3V/ max 100 mA) and the target's Vcc voltage sense. This pis 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. **Note**, that in the old verison of the USB-MSP430-FPA- rev. 1.0 the pin 4 is not connected. 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 (used in the USB-MSP430-FPA - rev.1.1 and up) (not used in the USB-MSP430-FPA rev.1.0)
5	TMS-In	TMS Input
6	NC	Not used.
7	TCK-In - **SBWTCK**	Boot Strap Loader / JTAG TCK Input pin / **Spy-Bi-Wire** TCK (note-3)
8	TEST-Vpp - **SBW-Vpp**	Boot Strap Loader / JTAG TEST Input pin, Blow Fuse voltage Vpp (+6.5V) / **Spy-Bi-Wire** Blow Fuse voltage (note-4)
9	GND	Ground
10	NC	Not used.
11	\RST	Microcontroller Reset Input pin.
12	BLTX-Out (*)	Boot Strap Loader Tx Output from the target (note-1)
13	GND (*)	Ground

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.*

Figure15-2 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.

Figure15-3 show interconnection between modified JTAG connector and MSP430F1122 microcontroller.

Figure15-4 and 15-5 show interconnection between JTAG/SBW connector and the MSP430Fxx microcontroller using **Spy-Bi-Wire** interface combined with RESET circuit.

Figure15-6 show interconnection between JTAG/SBW connector and the MSP430F2031 microcontroller using the **Spy-Bi-Wire** interface (for simplicity - RESET circuit is not present).

Figure15-7 to 153-10 shows interconnection between JTAG/SBW/BSL connector and the MSP430F22x4 microcontroller using the **BSL only, Spy-Bi-Wire and BSL** and **JTAG and BSL** interfaces (for simplicity - RESET circuits are not present).

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

Important: The old adapters - **USB-MSP430-FPA-1.0, 1.1** and **1.2** do not support in fully the **Spy-Bi-Wire** interface. The TEST pin from the FPA (see figure 15-4, 15-5) should not be connected, and in this case the **Spy-Bi-Wire** becomes fully functional, but without blow security fuse feature.

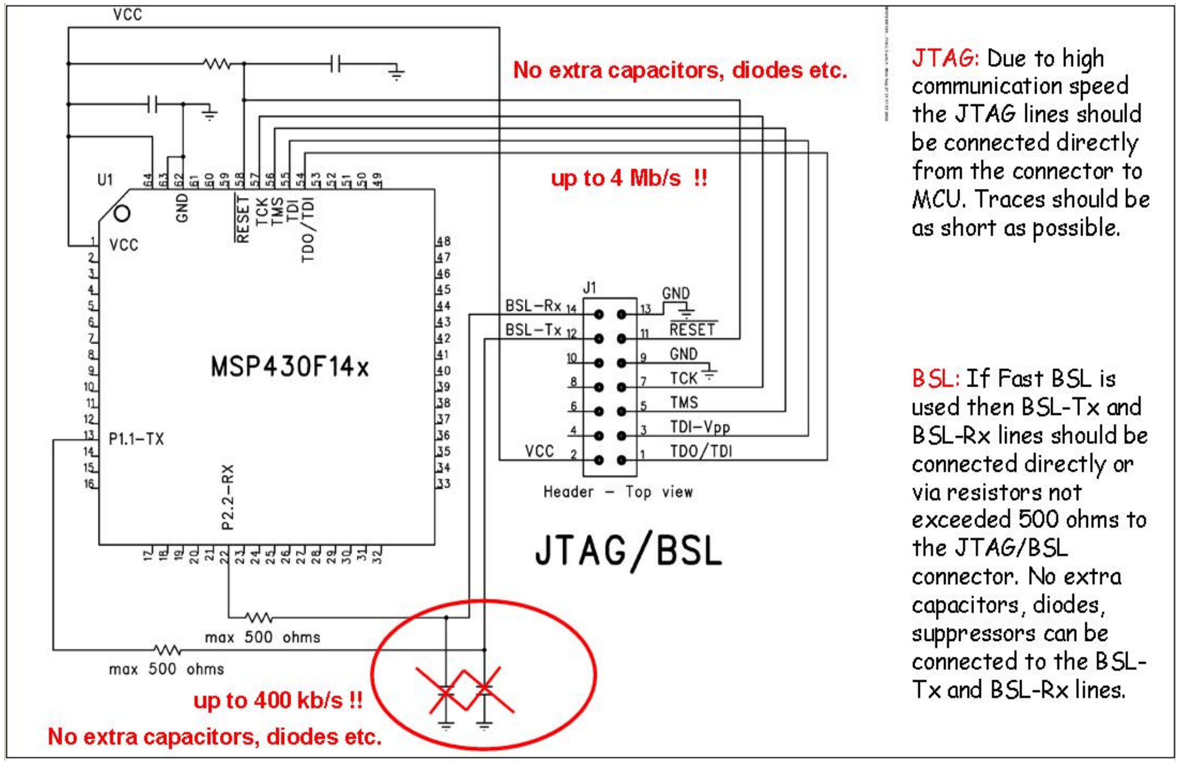

VCC
No extra capacitors, diodes etc.
up to 4 Mb/s !!
U1
GND
RESET
TCK
TMS
TDI
TDO/TDI
VCC
MSP430F14x
P1.1-TX
P2.2-RX
J1
BSL-Rx
BSL-Tx
GND
RESET
GND
TCK
TMS
TDI-Vpp
TDO/TDI
VCC
Header - Top view
JTAG/BSL
max 500 ohms
max 500 ohms
up to 400 kb/s !!
No extra capacitors, diodes etc.
JTAG: Due to high communication speed the JTAG lines should be connected directly from the connector to MCU. Traces should be as short as possible.
BSL: If Fast BSL is used then BSL-Tx and BSL-Rx lines should be connected directly or via resistors not exceeded 500 ohms to the JTAG/BSL connector. No extra capacitors, diodes, suppressors can be connected to the BSL-Tx and BSL-Rx lines.

Figure 15-2

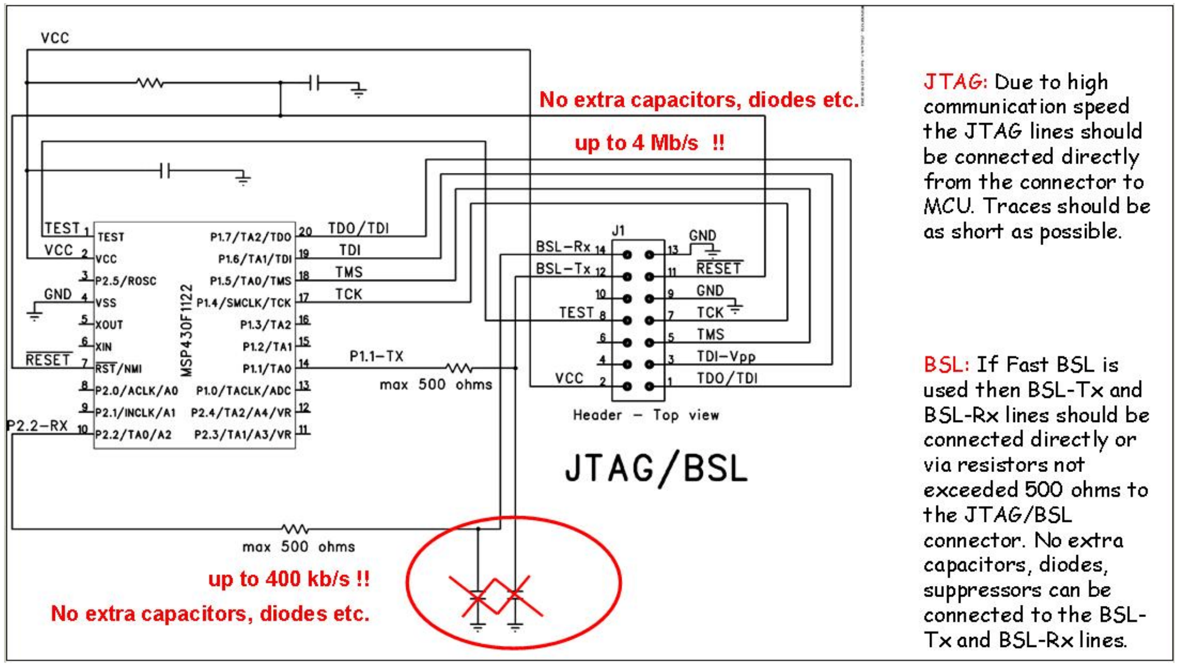

VCC
No extra capacitors, diodes etc.
up to 4 Mb/s !!
TEST
VCC
GND
RESET
P2.2-RX
TEST
VCC
P2.5/ROSC
VSS
XOUT
XIN
RST/NMI
P2.0/ACLK/A0
P2.1/INCLK/A1
P2.2/TA0/A2
MSP430F1122
P1.7/TA2/TDO
P1.6/TA1/TDI
P1.5/TA0/TMS
P1.4/SMCLK/TCK
P1.3/TA2
P1.2/TA1
P1.1/TA0
P1.0/TACLK/ADC
P2.4/TA2/A4/VR
P2.3/TA1/A3/VR
TDO/TDI
TDI
TMS
TCK
P1.1-TX
max 500 ohms
J1
BSL-Rx
BSL-Tx
TEST
VCC
GND
RESET
GND
TCK
TMS
TDI-Vpp
TDO/TDI
Header - Top view
JTAG/BSL
max 500 ohms
up to 400 kb/s !!
No extra capacitors, diodes etc.
JTAG: Due to high communication speed the JTAG lines should be connected directly from the connector to MCU. Traces should be as short as possible.
BSL: If Fast BSL is used then BSL-Tx and BSL-Rx lines should be connected directly or via resistors not exceeded 500 ohms to the JTAG/BSL connector. No extra capacitors, diodes, suppressors can be connected to the BSL-Tx and BSL-Rx lines.

Figure 15-3

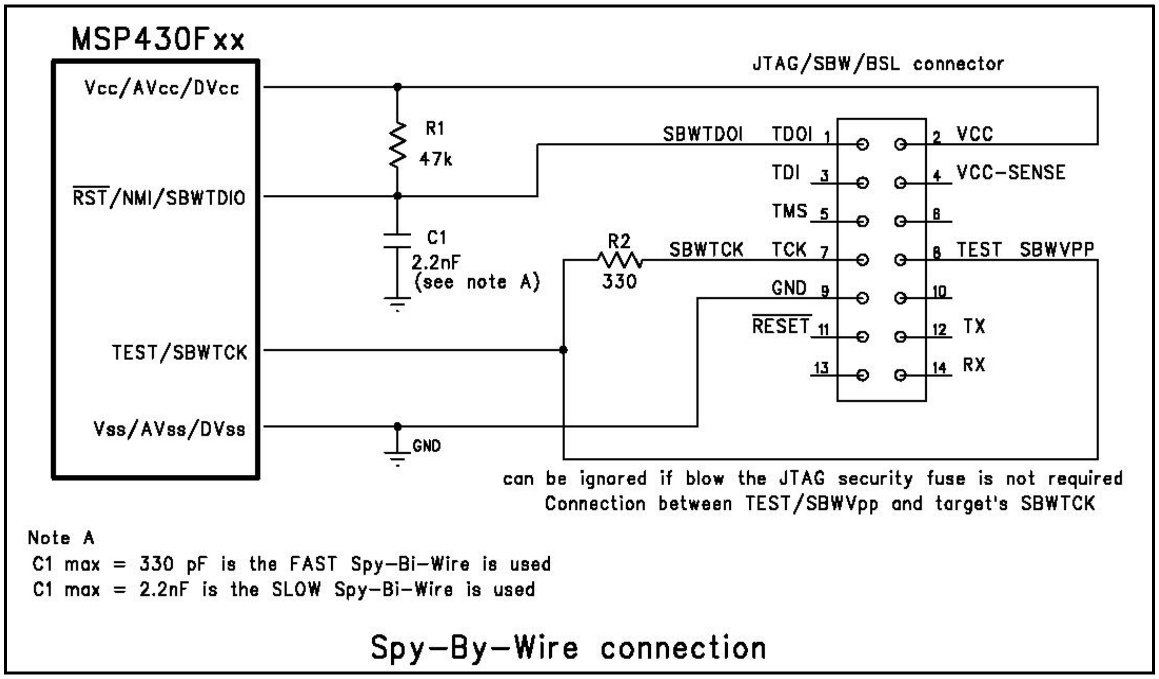


Figure 15-4

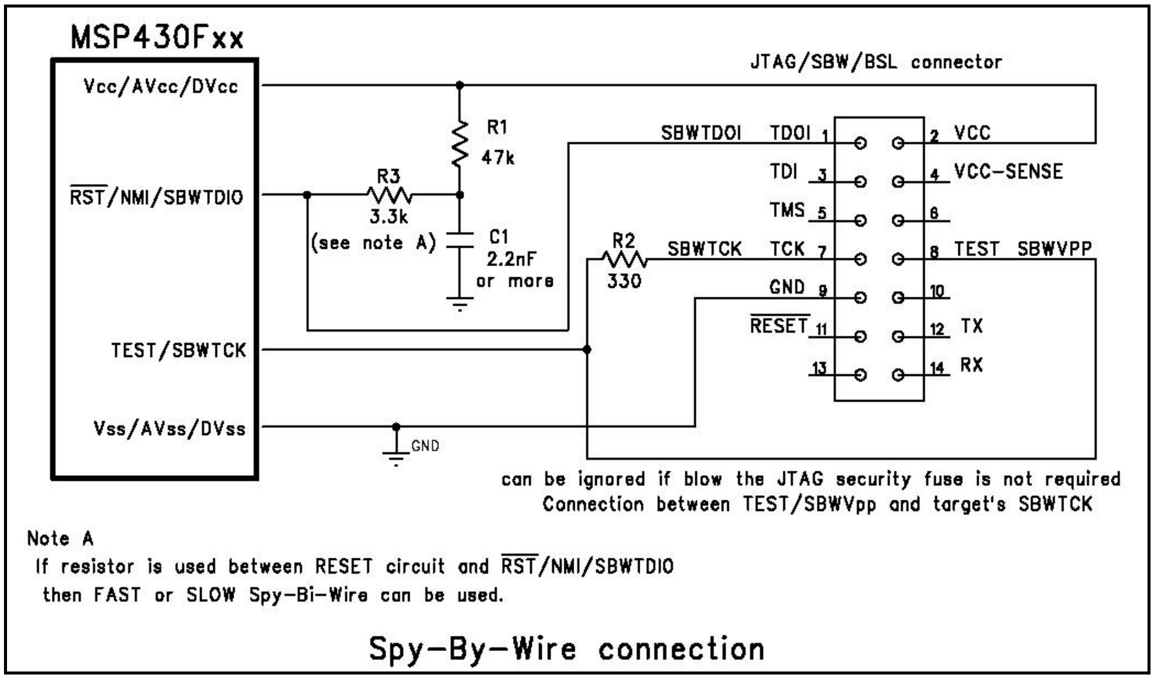


Figure 15-5

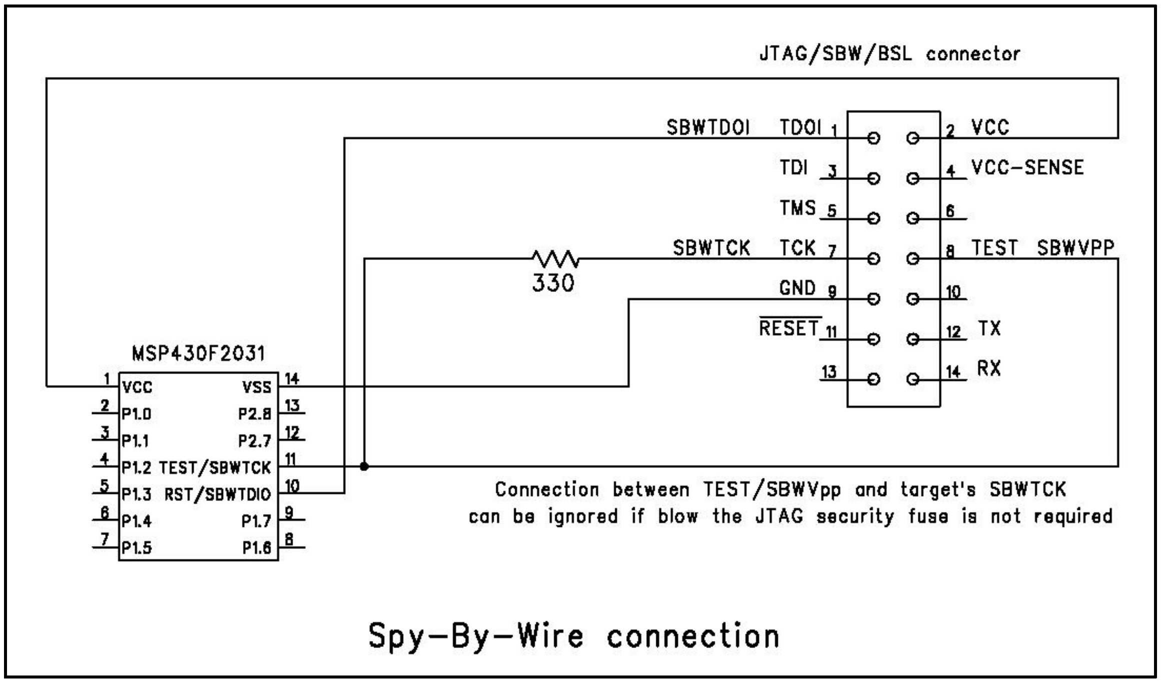


Figure 15-6

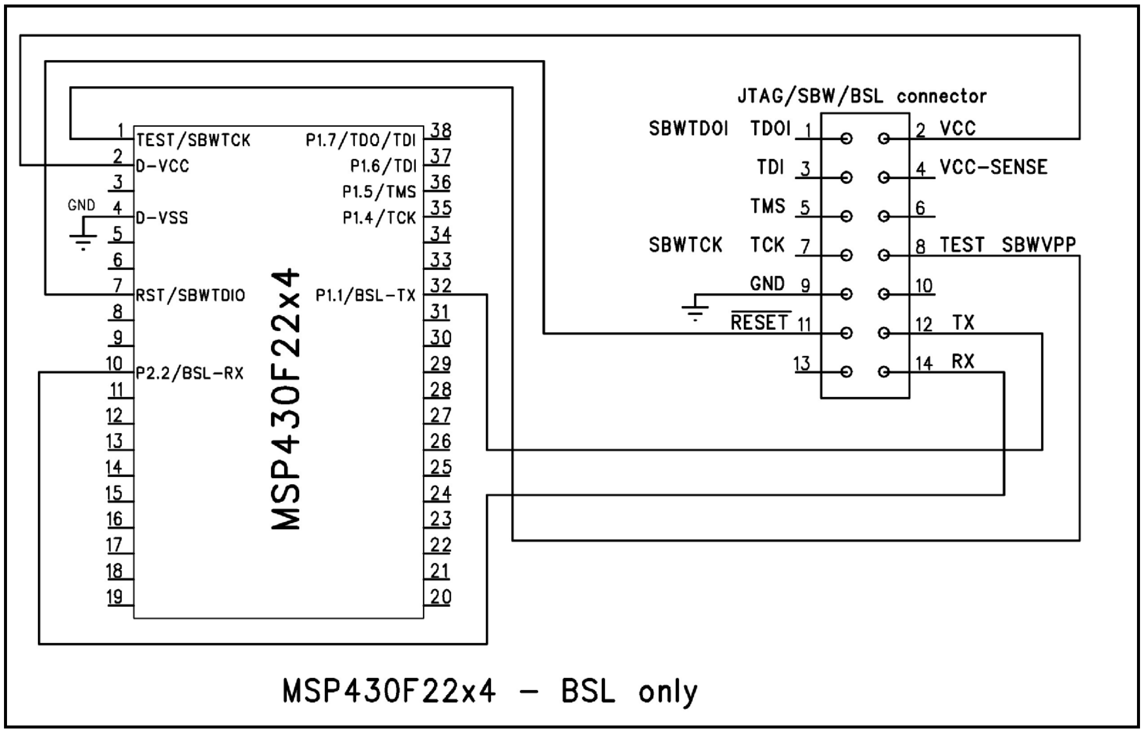


Figure 15-7

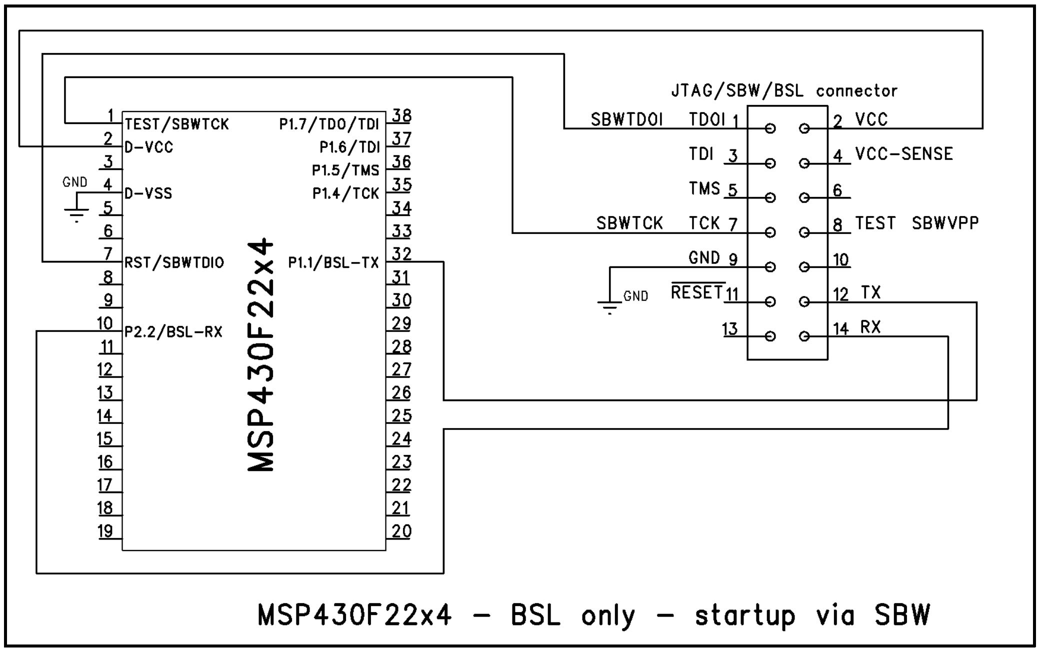


Figure 15-8

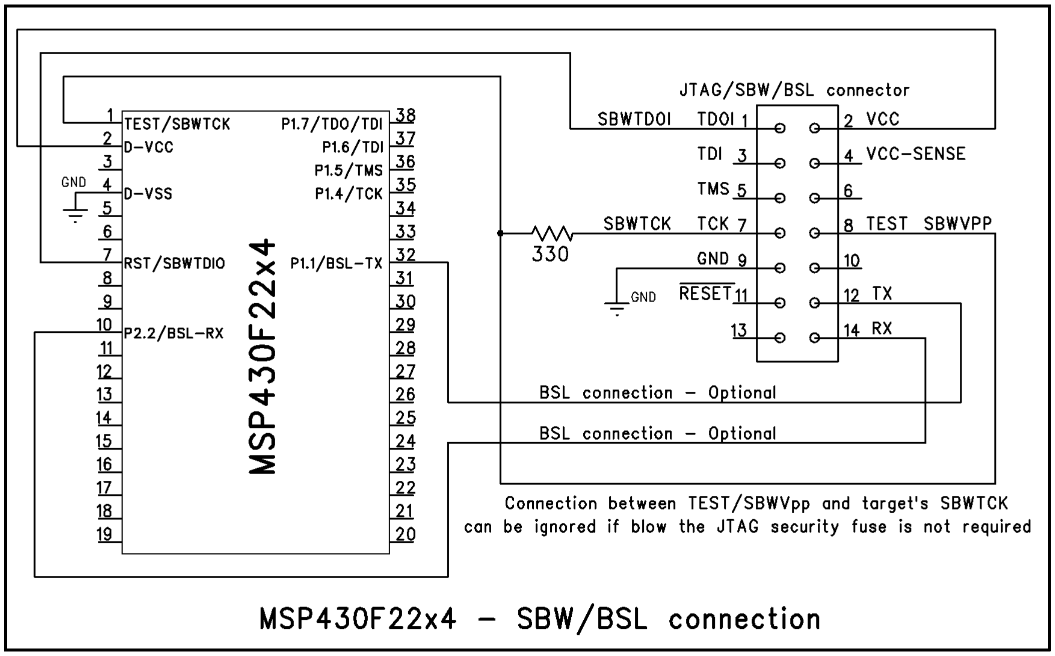


Figure 15-9

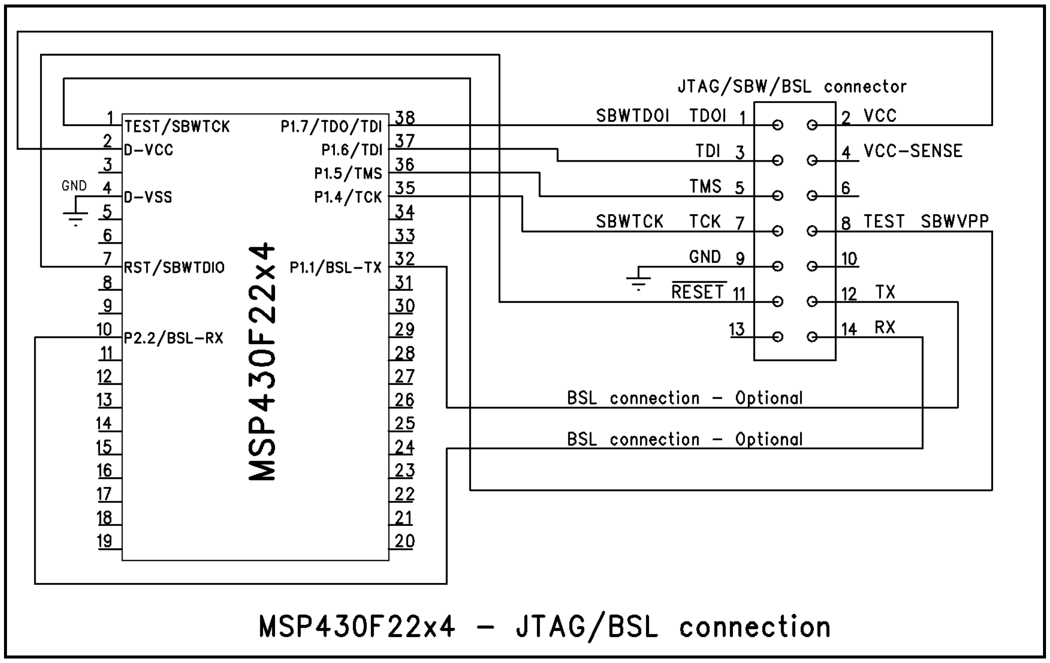


Figure 15-10

In a typical design all lines are connected directly between BSL connector and the dedicated microcontoller'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 Ω. 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.

Specification:

FPA adapters - **USB-FPA-6.x**	
PC Communication Interface:	- Full Speed USB-1.1 (12Mbits/s)
USB connector	- Adpater site: USB-type B, Computer site: USB-type A
Target connector	- 14 pins header connector - standard JTAG / Spy-Bi-Wire pinhead with added BSL connection to unused pins.
DC Power - from USB Interface	- 5V +/- 20%, 70mA + target's current (0-200mA)
Target Device DC supply	
- external	- 1.8 V to 3.6 V
- from programming adapter	- Fixed 3.3 V / 200 mA max (USB-FPA-6.x) Programmable 1.8 to 3.6 V step 0.1 V / 200 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.
Operating temperature	- +5 to +55 C deg.
Storage temperature	- - 40 to +85 C deg.
Humidity	- 30 to 80 %

Flash Programming specification - JTAG		**4 Mb/s**	**1 Mb/s**	**400 kb/s**
**** Block Words Write (64 bytes) ****				
1. MCLK clock frequency (From FPA to MSP430 via TDI pin)	-	6 MHz	1.2 MHz	750 kHz
2. Flash Timing Gen. Frequency (f FTG)	-	428.6 kHz	400 kHz	375 kHz
3. Cumulative Programming Time (t CPT)				
MSP430F1xx, F4xx	-	1.72 ms	2.14 ms	2.48 ms
MSP430F2xx, MSP430X	-	1.50 ms	1.87 ms	2.16 ms

**** Single Word Write ****

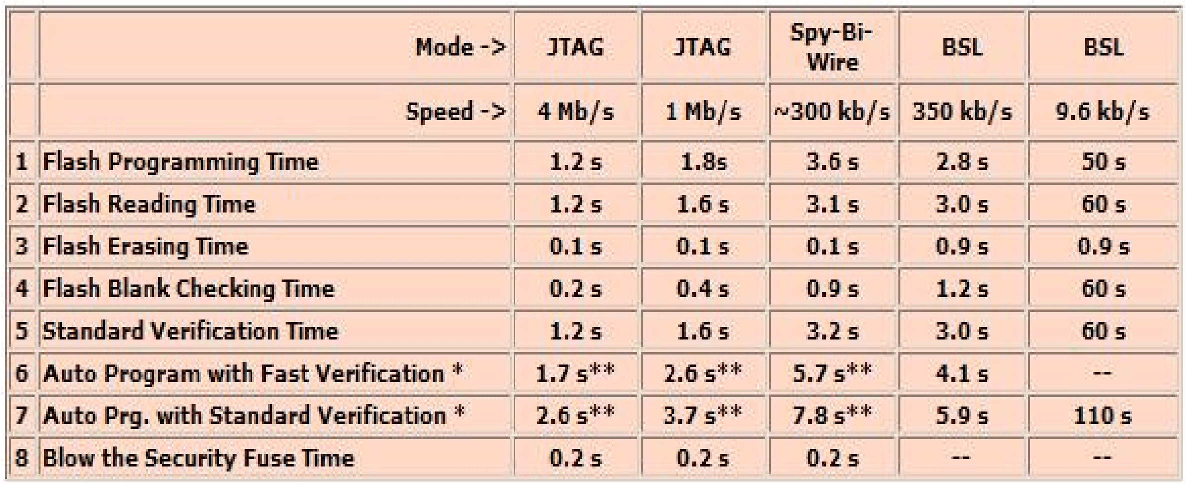
1. Flash Timing Gen. Frequency (f FTG)	-	428.6 kHz	428.6 kHz	428.6 kHz
2. Cumulative Programming Time (t CPT) (32 words)				
MSP430F1xx, F4xx	-	2.62 ms	2.62 ms	2.62 ms
MSP430F2xx, MSP430X	-	2.40 ms	2.40 ms	2.40 ms

Flash Erasing specification - JTAG - any speed

Flash Timing Gen. Frequency (f FTG)	-	428.6 kHz
Mass or Main Memory Erase Time		
MSP430F1xx, F4xx	-	248 ms

Table B-1 - Performance

Programming / reading times for a target microcontroller with 32 kB of Flash Memory



	Mode ->	JTAG	JTAG	Spy-Bi-Wire	BSL	BSL
	Speed ->	4 Mb/s	1 Mb/s	~300 kb/s	350 kb/s	9.6 kb/s
1	Flash Programming Time	1.2 s	1.8s	3.6 s	2.8 s	50 s
2	Flash Reading Time	1.2 s	1.6 s	3.1 s	3.0 s	60 s
3	Flash Erasing Time	0.1 s	0.1 s	0.1 s	0.9 s	0.9 s
4	Flash Blank Checking Time	0.2 s	0.4 s	0.9 s	1.2 s	60 s
5	Standard Verification Time	1.2 s	1.6 s	3.2 s	3.0 s	60 s
6	Auto Program with Fast Verification *	1.7 s**	2.6 s**	5.7 s**	4.1 s	--
7	Auto Prg. with Standard Verification *	2.6 s**	3.7 s**	7.8 s**	5.9 s	110 s
8	Blow the Security Fuse Time	0.2 s	0.2 s	0.2 s	--	--

*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).

MSP430F2xx, MSP430X	-	24.8 ms