FET-Pro430

MSP430 Flash Programmer

User's Manual

Software version 2.8

PM014A01 Rev.9
April-05-*2010*

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 ASSOCIATED THE 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 a single computer;

B. make one copy of the software for archival purposes, or copy the software onto the hard disk of Your computer and retain the original for archival purposes;

C. use the software on the network, provided that You have a licensed copy of the software for each computer that can access the software over that network

YOU MAY NOT:

A. copy the printed documentation that accompanies this Product

B. 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;

C. redistribute, in whole or in part, any part of the software component of this Product.

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.

Table of Contents

1. Introduction

FET-Pro430 programming software is a software package designed to operate with existing programming adapters provided by Texas Instruments and other vendors. **FET-Pro430** requires device drivers and libraries (DLL) provided by adapter manufacturers, while supplying the software features you have come to expect and rely upon from Elprotronic, Incorporated. The **FET-Pro430** can program Texas Instruments MSP430Fxx family of microcontrollers via JTAG interface, using the parallel or USB ports. The programming speed and the size of the code that can be programmed are dependent only on the interface adapter and the target device.

To simplify production process, the programming software package can assign serial number, model type, and revision number. 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.

2. Features

FET-Pro430 programming software is designated to program the Texas Instruments MSP430Fxx microcontroller family via the JTAG interface using MSP430.dll driver and Texas Instrument's Flash Emulation Tool (FET) adapter.

Major features of the *FET-Pro430* programmer are:

* ***FET-Pro430*** programming software is a shell that uses the Texas Instruments' MSP430.dll driver to facilitates communication with the target device and TI's programming adapters - parallel port Flash Emulation Tool (FET) or TI-USB-FET. Communication speed is determined by the MSP430.dll driver and used FET
* Supports all MSP430Fxx microcontrollers from TI
* Blow the JTAG security fuse
* Full memory or sector memory erase
* Write Check Sum verification
* No code size limitations
* Target device can be powered from the programming adapter or from external source.
* Easy to use Windows™ based software.
* Programmer accept TI (*.txt), Motorola (*.s19) and Intel (*.hex) data files for programming.
* Combine code files
* 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 allows to review information about the flashed target devices.

3. Getting Started

FET-Pro430 programmer package contains:

1. One READ ME FIRST document.
2. One FET_Pro430 Flash Programmer CD ROM (Software + Manual).

3.1 Software Installation

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

1. Insert the ***FET-Pro430*** Programming Software CD into your CD-ROM drive.
2. ***FET-Pro430*** Programmer Setup wizard appears automatically. Click *Install FET-Pro430 Flash 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.

FET-Pro430 programming software uses standard TI's MSP430.dll library and TI's programming adapter (FET). Current version of the software package contains TI's MSP430.dll and HIL.dll supporting the MSP430-FET (parallel port version) and MSP-FET430UIF (USB port version). To connect to the programming adapter, select the parallel port (LPT1, LPT2, or LPT3) or USB port (TI-USB) as described in section 7.3.

3.2 Driver Installation

Parallel port FET requires DriverX to be installed. The DriverX should be installed with the Kickstart software. Follow instruction attached to your tool (FET) from TI. No additional action is requited to activate the driver for the parallel port FET. The USB driver for the MSP-FET430UIF can be installed using the latest KickStart software, or the TI's USB driver attached to the current ***FET-Pro430*** software package can be used. Follow instructions below to install TI's USB driver:

1. Plug in the MSP-FET430UIF to the PC USB Port, using provided cable extender (USB-A to USB-B)
2. The ***"New hardware has been found"*** should be displayed
3. Instruct the Wizard to install the hardware driver from a specific location
4. Point the Hardware Wizard to the according folder where the corresponding driver information files are located on your hard disc. Drivers in the previously installed software are located (on a default installation) in directory:

   C:\Program Files\Elprotronic\FET-Pro430 Flash Programmer\USB-Driver\WinXp
5. Driver installation process will start. Note, that Windows XP shows a warning that the driver is not certified by Microsoft. Ignore this warning and click "Continue Anyway"

Note that the driver installation wizard starts twice, as two drivers are installed. Reboot computer at the end.

3.3 Hardware Installation

Follow instruction attached to your hardware tool (FET) from TI.

3.4 Starting up "FET-Pro430" Flash Programmer

To start the ***FET-Pro430*** Flash Programmer click on the *FET-Pro430 Elprotronic* icon.

Figure 3.3-1

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

4. Programming Dialogue Screen

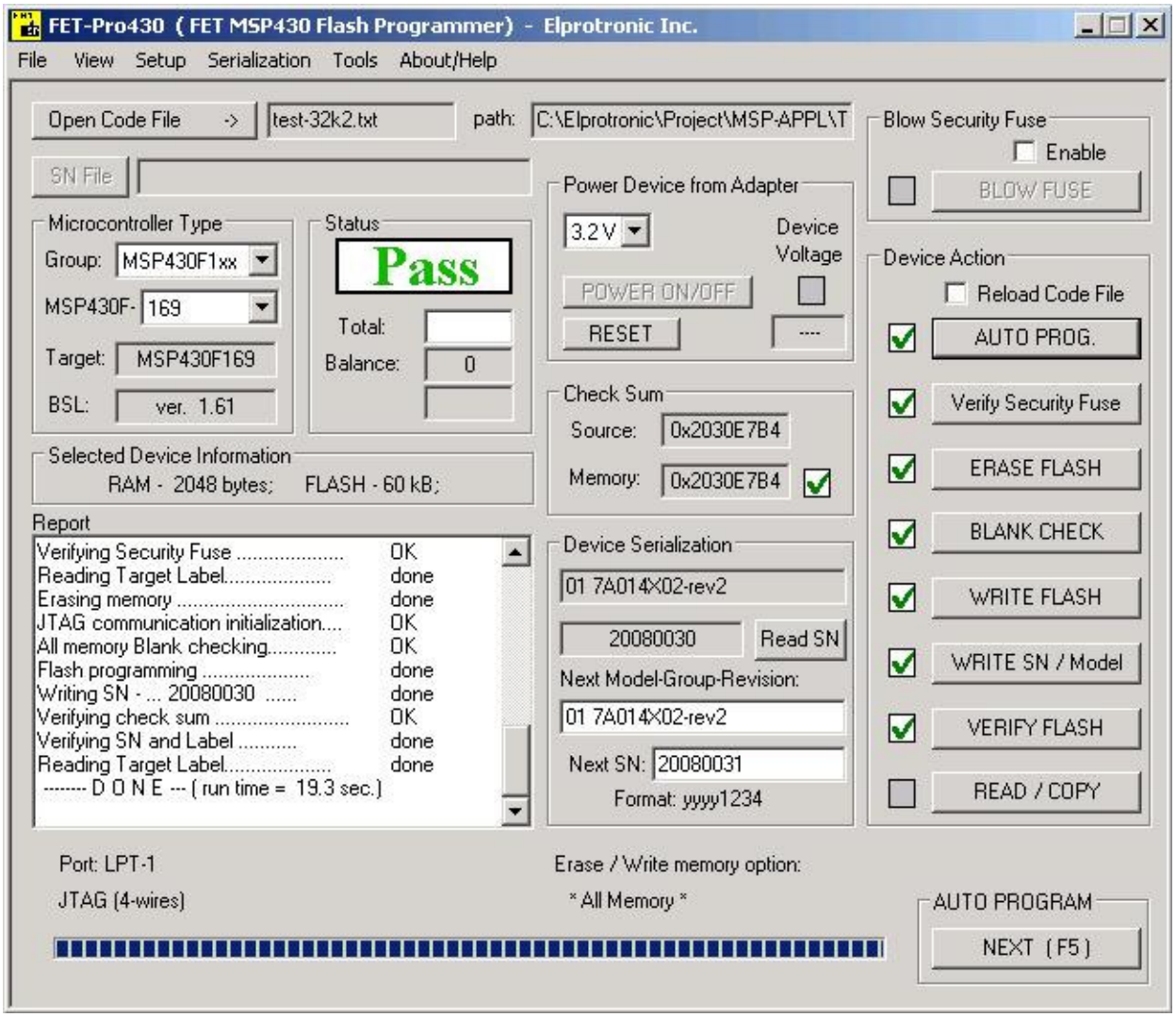

Figure 4-1. Programming dialogue box screen.

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 Microcontroller Type

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

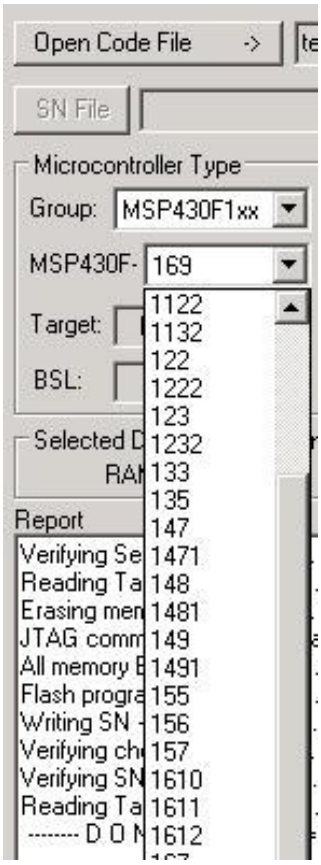

Figure 4.1-1

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

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

4.2 *Code File Management*

FET-Pro430 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.

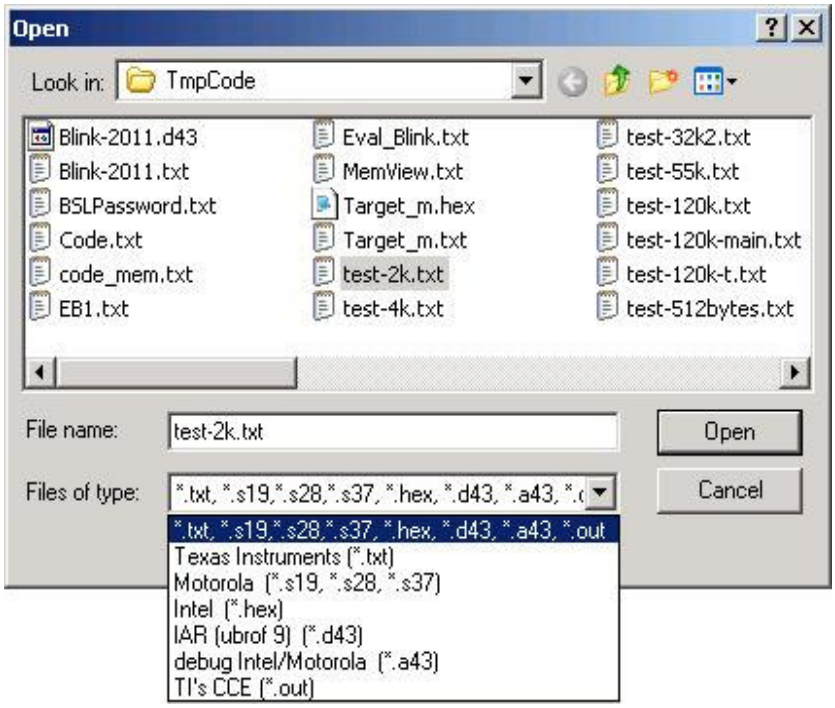

Figure 4.2-1

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 4.2-1. 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 4.2-2 will be displayed.

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

Figure 4.2-2

'View' menu (see chapter 5).

The ***Combine Code Files*** option allows up to 40 code files to be loaded into the PC memory. When this option is selected the programmer will create a new data block, which will contain the combined data of the user selected files. In order to add a code file to the newly created data block, the user needs to press the ***ADD Code File*** button. The programmer will then prompt the user to specify the code file to be appended to the newly created memory block, using the window in Figure

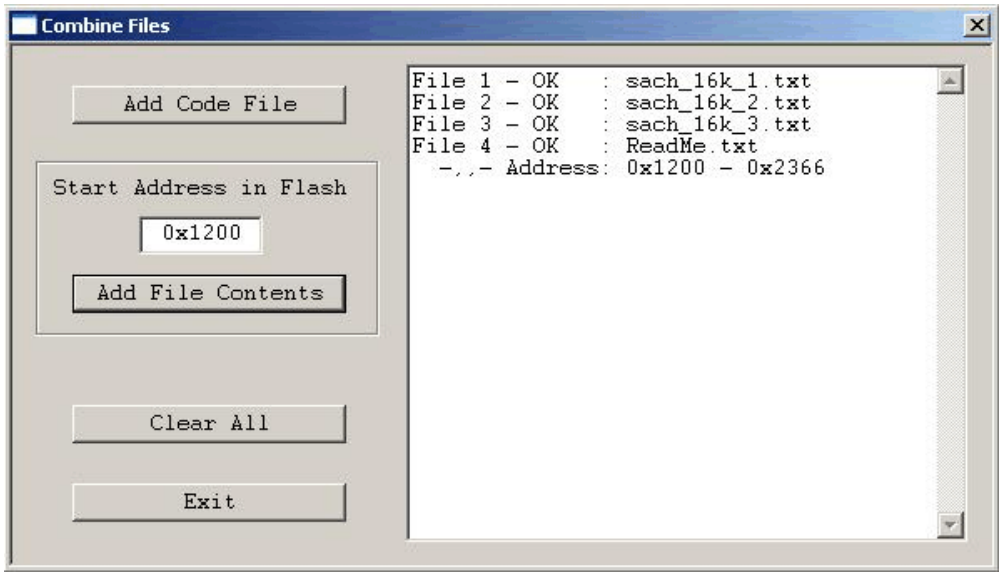

Figure 4.2-3

4.2-1. 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 4.2-3 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 4.2-1 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 4.2-3 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 4.2-4 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

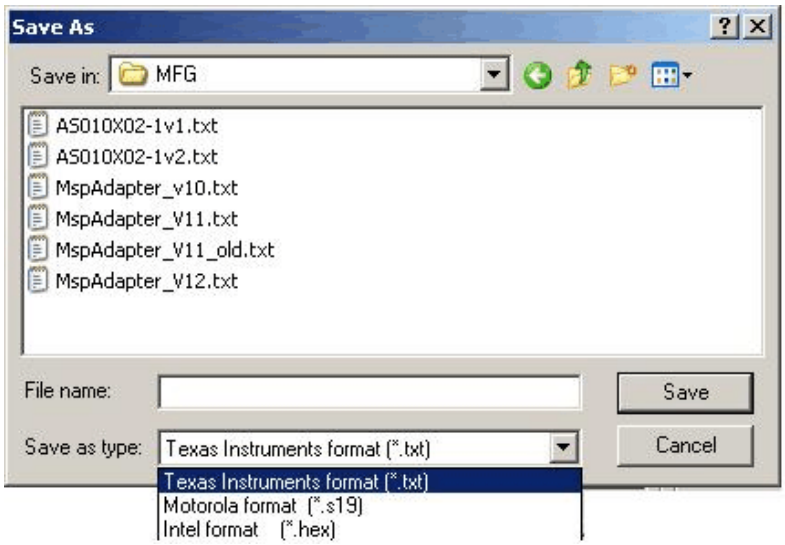

Figure 4.2-4

Code File and Save Code File options.

4.3 Blow Security Fuse and Open Password File

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

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

To blow the Security Fuse the check mark ***'Enable'*** must be selected first (see Figure 4.3-1).

Figure 4.3-1

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

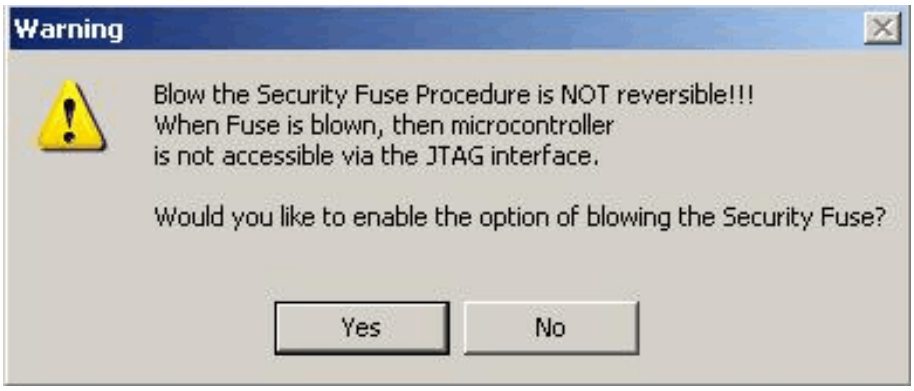

Figure 4.3-2

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

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

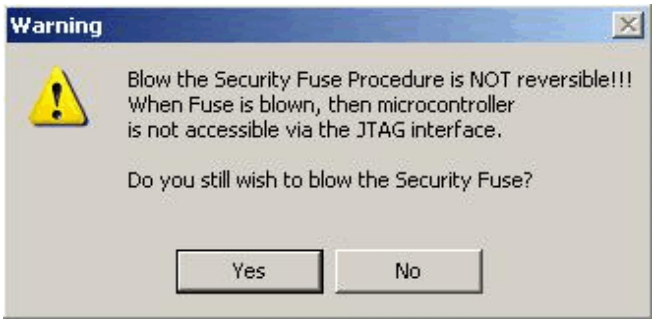

Figure 4.3-3

Figure 4.3-4

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

4.4 Power Device from Adapter

The programming adapter is powered from the USB Port interface. By clicking POWER ON/OFF button you can turn the power on or off on the target device. If programming adapter has capability to program the output DC level (like TI-USB-FET), then the desired Vcc can be selected between 2.2 to 3.6 V using selector box (figure 4.4-1). If the popular parallel port version of FET is used, then the setup of the Vcc is irrelevant.

Figure 4.4-1

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.

4.5 Device Action box

Device Action box contains 8 buttons (see Figure 4.5-1) and 8 status boxes. Each button allows a specific action to be executed. Software procedures related to each action allow you to fully execute the desired task, without the need to follow a specific sequence of actions. Every action starts by powering up the target device, if ***Power Device from the Adapter*** is enabled. The communication with the target device is initiated via JTAG. The security fuse is verified, if access to the microcontroller is available. 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.

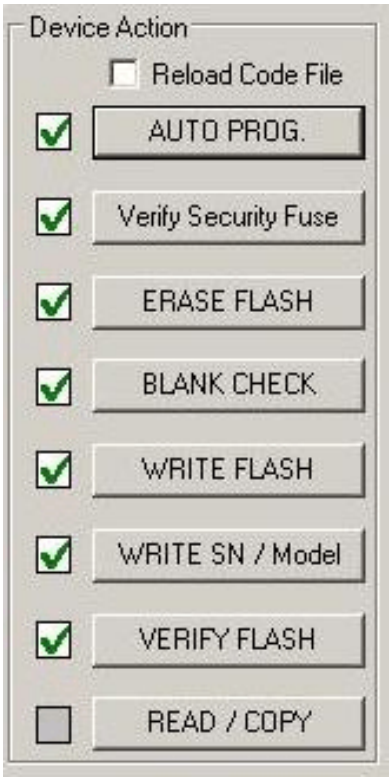

Figure 4.5-1

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.4.5.2). 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.4.5-3) 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.

4.5.1 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:

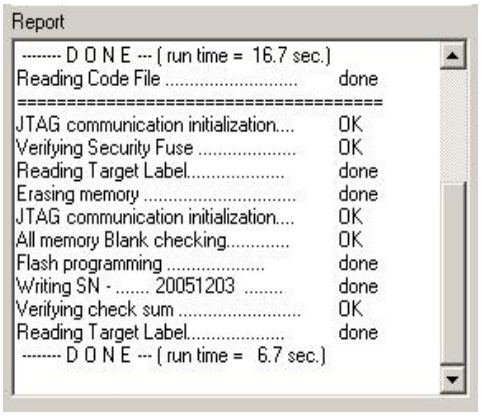

Figure 4.5-2

- reload code file when "***Reload Code File***" is selected

(useful for debugging when the code file is frequently modified)

- initialization
- read labelling information (Serial Number, Model,

Group, Revision) (optional)

- erase flash memory,
- confirm if memory has been erase,
- flash programming and verification,
- labelling information generation,
- flash memory check sum verification,
- retrieve labelling information,
- blowing the security fuse (if enabled).

In the report window you can see a typical report message during the Auto Program procedure (see Fig. 4.5-2).

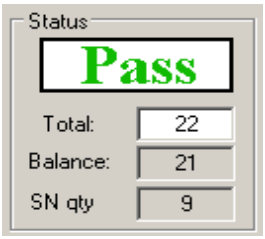

Figure 4.5-3

Status window (see fig. 4.5-3) has a counter that is useful in production process. The total number of programmed microcontrollers can be entered in the ***Total*** edit line. The ***Balance*** line shows the number of microcontrollers that have not been programmed yet. The Balance counter is initialized to the value entered in the ***Total*** edit line and is decremented every time *Auto Program* is completed successfully.

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

4.5.2 Verify Security Fuse

This button allows the security fuse to be verified. This is useful, if you try to check if the security fuse is blown. This procedure is used for test purposes only.

4.5.3 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:

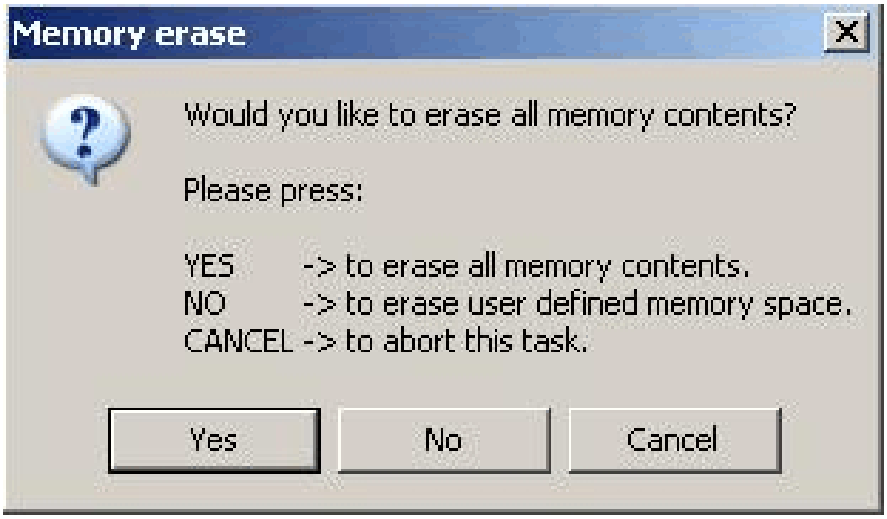

Figure 4.5.3-1

4.5.4 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.

4.5.5 Write Flash button

When write flash button is clicked, then contents from the code file 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.*

4.5.6 Verify Flash button

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

Check sum calculated from the code file data is displayed in the ***Source*** line of the ***Check Sum*** group (see Fig.4.5.6-1), and check sum calculated from the target microcontroller flash memory data is displayed in the ***Memory*** line of this group.

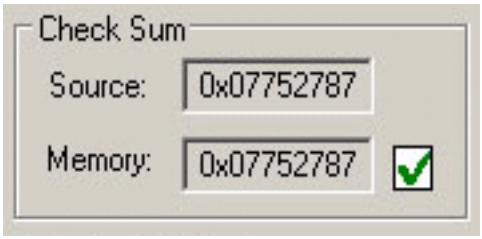

Figure 4.5.6-1

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.*

4.5.7 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.4.5.7-1). This window can also be selected from ***'Flash Memory Data'*** from the ***'View'*** menu. Flash memory data viewer,

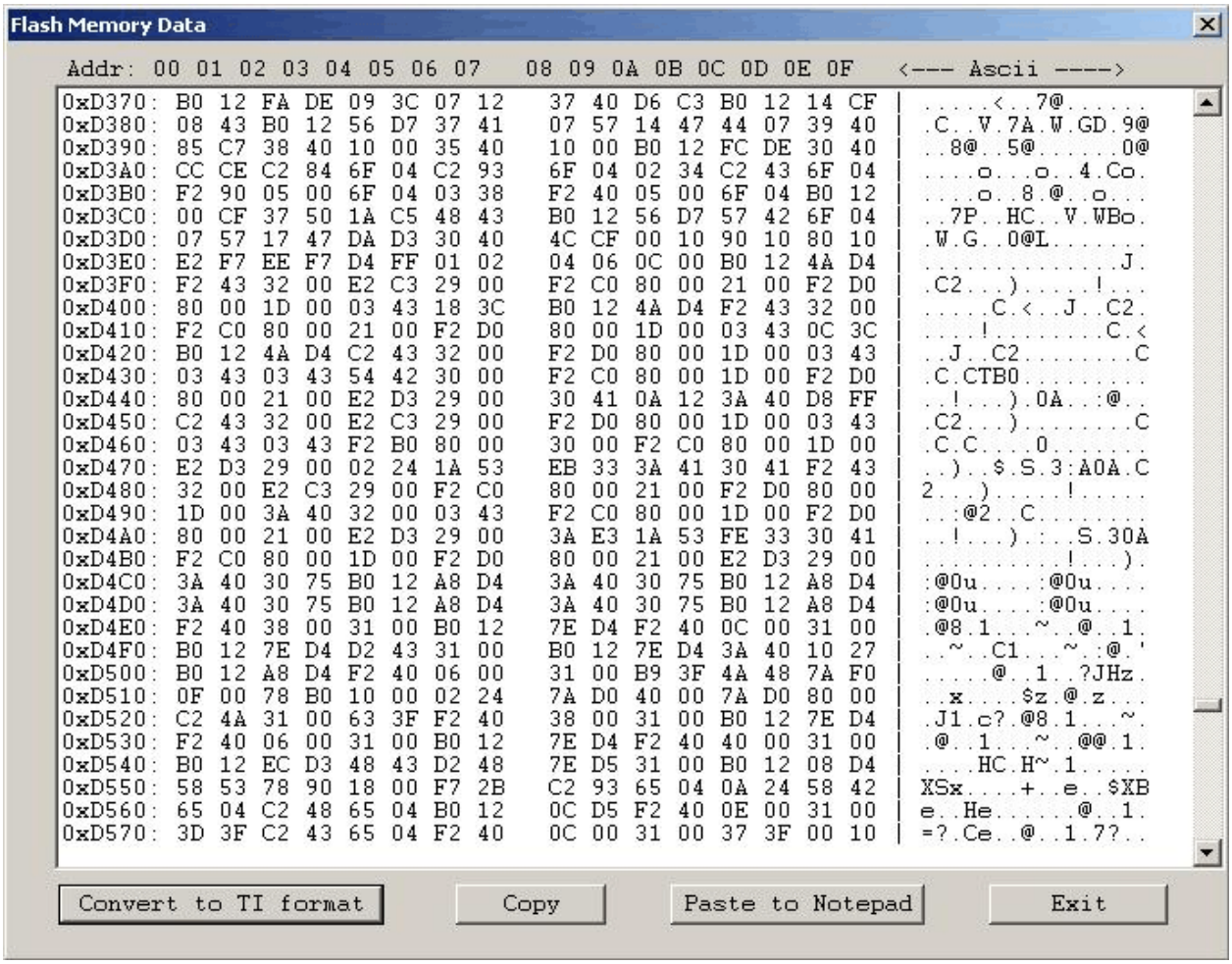

Figure 4.5.7-1

shown in figure 4.5.7-1, 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***'.

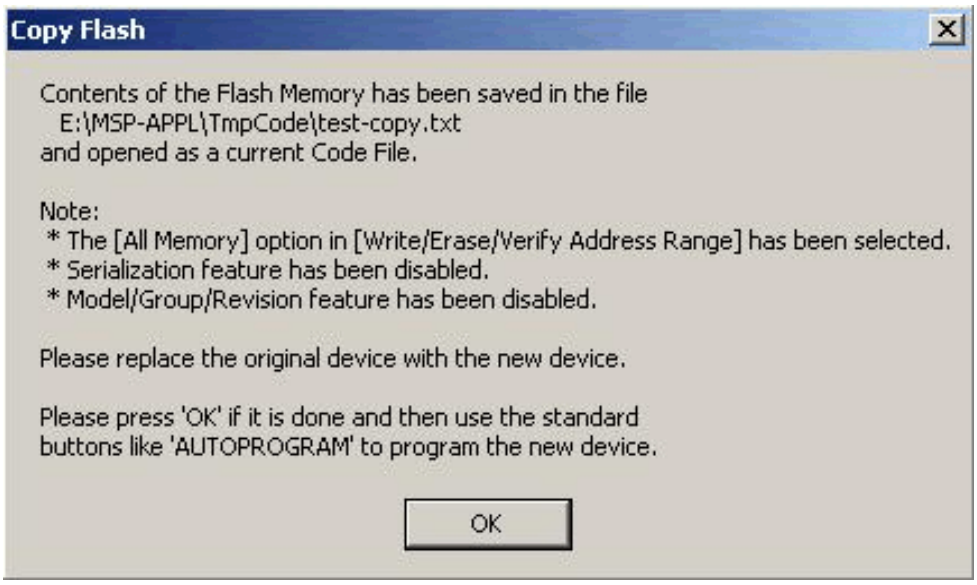

Figure 4.5.7-2

Following message will be displayed. When the button ***'OK'*** is pressed then programmer is ready to program the destination microcontrollers.

Figure 4.6-1

4.6 Next button

Figure 4.6-2

Figure 4.6-3

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.4.6-1). 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.4.6-2). 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.4.6-3). The read flash procedure will be called, if button ***'NEXT'*** or function key ***F5*** is pressed.

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

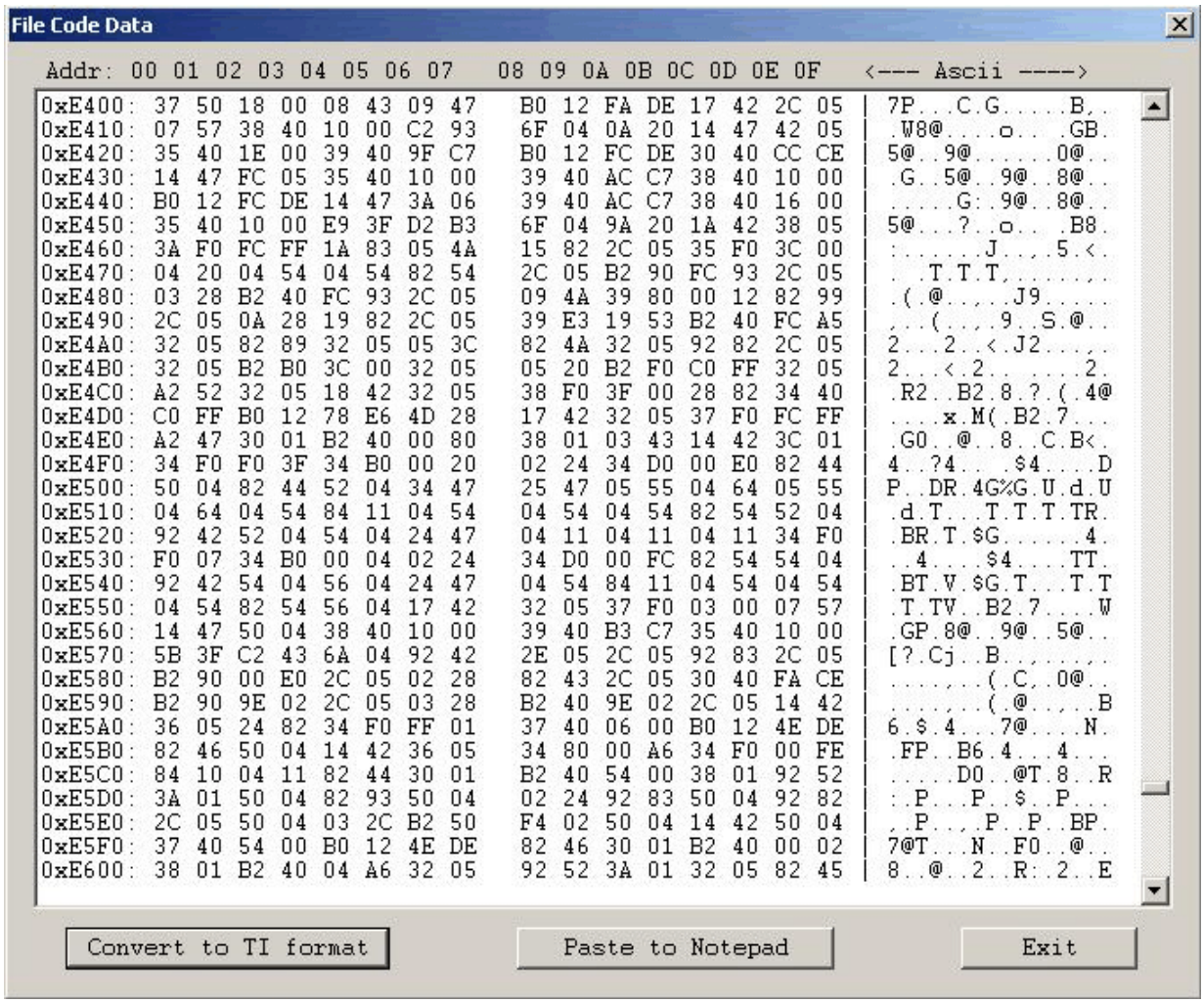

Figure 5-1

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

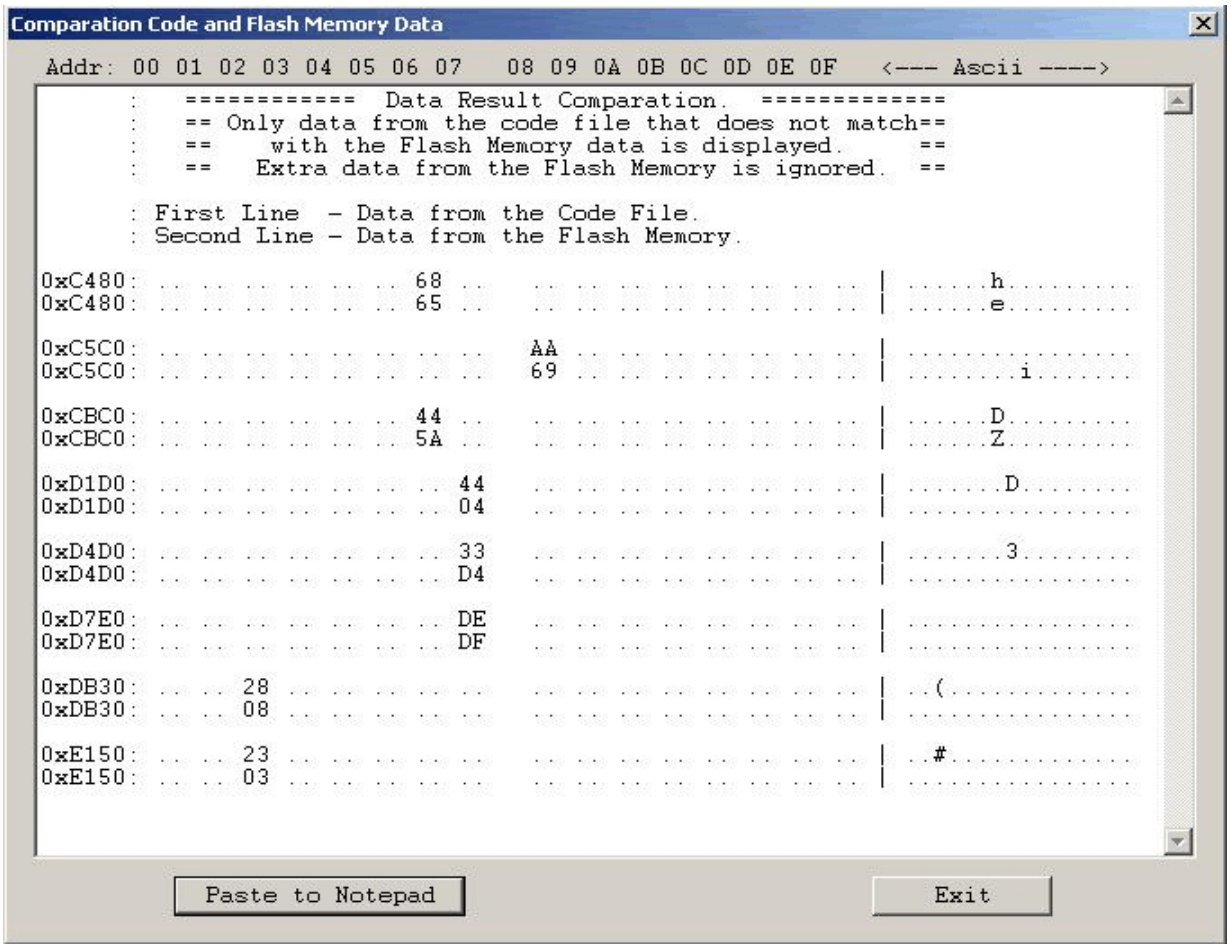

Figure 5-2

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.

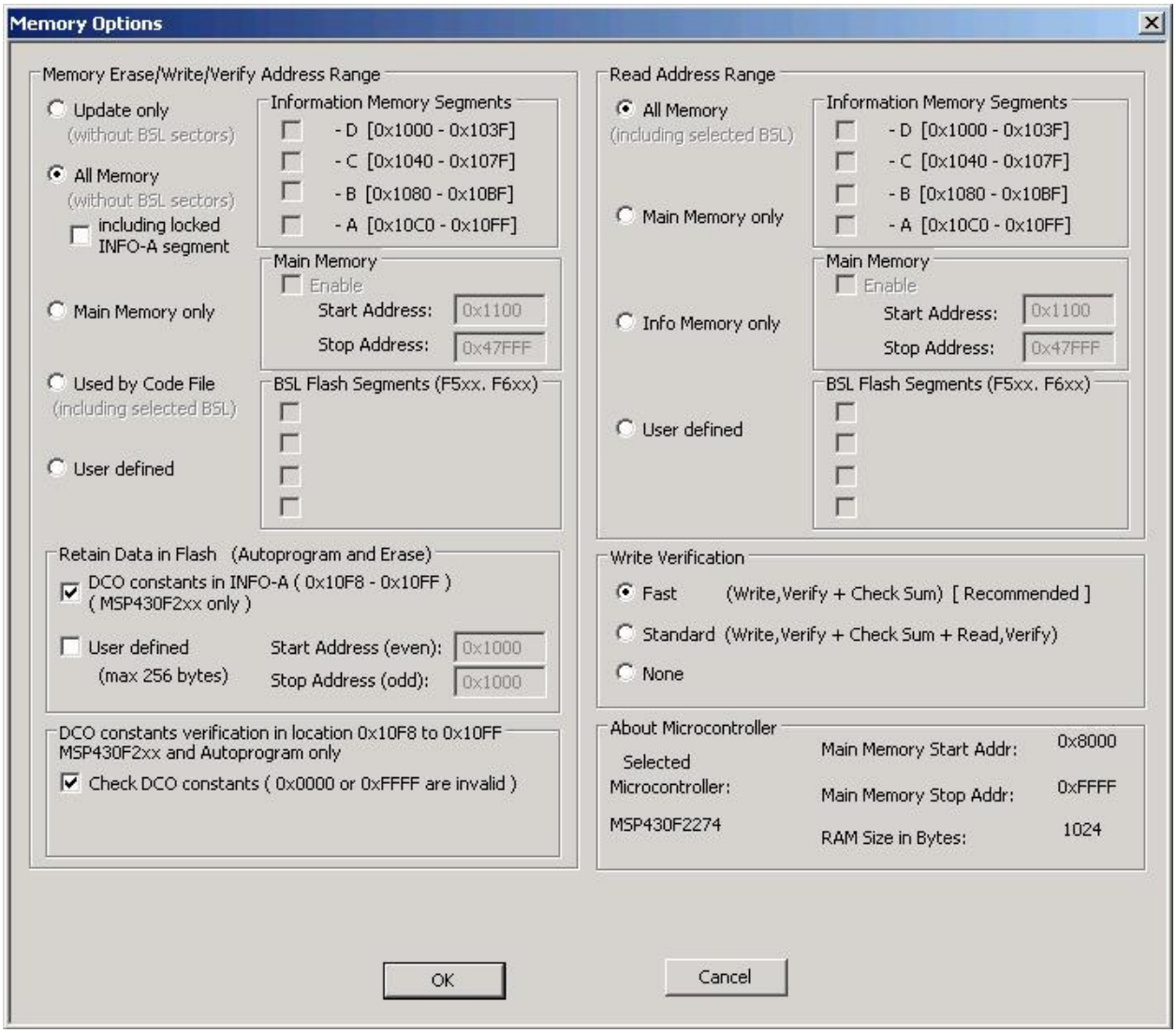

Figure 6-1

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.

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. ***Retain Data in Flash Group:***

The MSP430F2xx series has the DCO calibration data saved in the INFO memory at addresses 0x10F8 to 0x10FF. However, when the info segment is erased, then the DCO calibration data can be erased also. When the ***DCO Calibration Data*** box is selected in the ***Retain Data in Flash*** group, Autoprogram button is pressed and the MSP430F2xx microcontroller is selected, then contents of the info memory at location 0x10F8 to 0x10FF is read, whole action is performed (erase, blank check, program) and contents of the original DCO data (info at location 0x10F8 to 0x10FF) are restored.

User defined option in the ***Retain Data in Flash*** group allows to specify other region to be restored after erase, program and verification. This option can be used with any MSP430

microcontroller type. Location of the retain data block is not limited and can be used any part of flash - info or main memory. Maximum size of the retain data block is limited to 256 bytes only.

6.2 Read Group

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

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

The meaning of each option is the same as for the erase/write/verify procedure. The *Info Memory only* option works the same way as *Main memory only* option described above, except that only information memory is modified.

6.3 Verification Group

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

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

***Fast Verification*:**
Fast verification method is performed using a pseudo signature analysis (PSA) algorithm.

***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.

6.4 *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

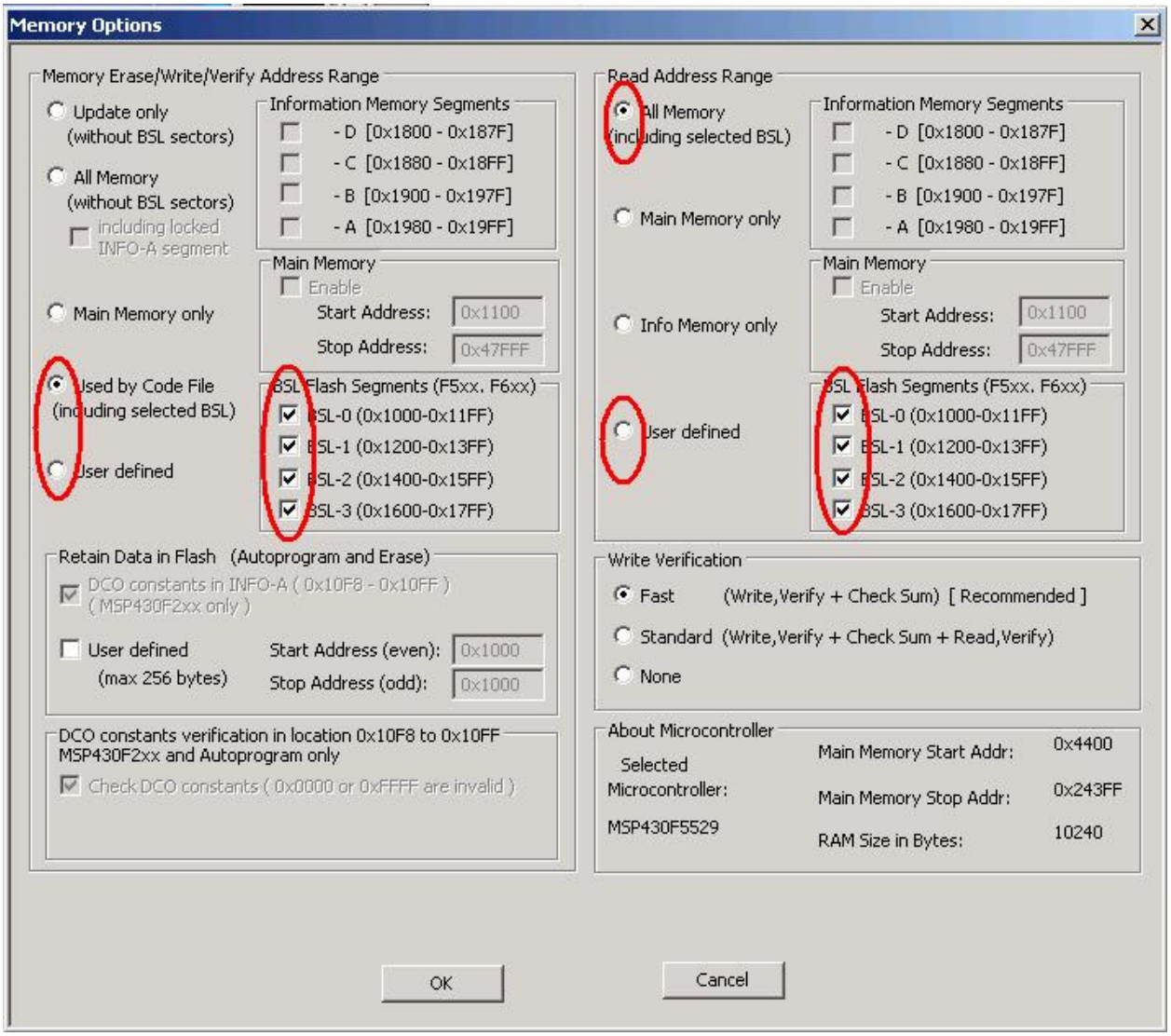

Figure 6.3

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).

Figure 6.4

7. Target's Connection - Reset Options

7.1 Communication with Target Device

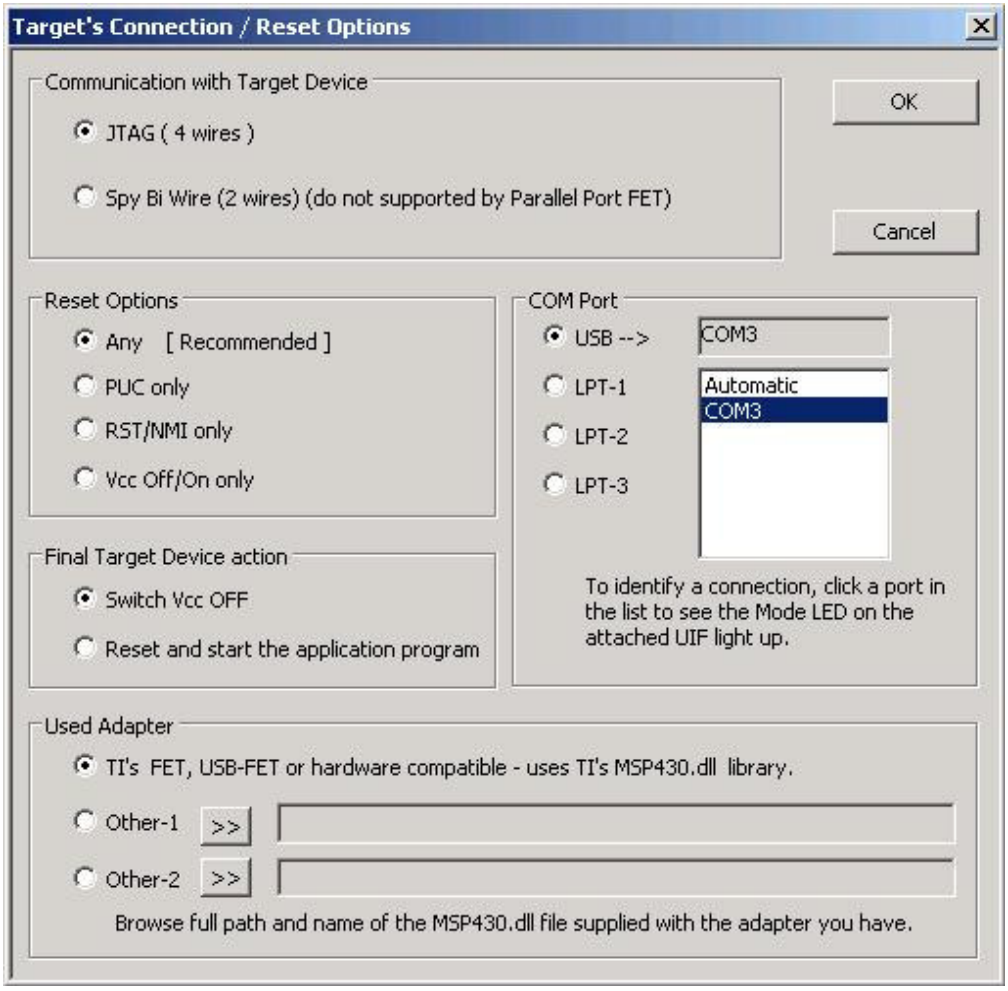

Figure 7-1

Communication with the target devices can be selected in the **Target's Connection** dialogue screen (Figure 7.1). Most of the microcontrollers MSP430Fxx have only standard JTAG communication interface. In this case the **"Standard JTAG"** selection should be used. The latest MSP430F2xx microcontrollers with small packages have the **Spy Bi Wire** (2 wires only) interface, or 4 wires JTAG combined with Spy Bi wire Interface. In this case the **2/4 wire JTAG** or **Spy Bi Wire** can be selected. Before selecting non standard JTAG communication interface make sure that your FET is

supporting selected communication interface. The non standard communication interface is by the Texas Instruments USB-FET (MSP-FET430UIF) with the latest firmware (released in Oct.2005 and later) . Ask TI for firmware upgrade if the USB-FET you have has an older firmware.

7.2 Reset Options

The Target's Reset Options screen (figure 7-1) enables the user to select the following Reset method.

- PUC Reset - The device is reset using PUC (i.e. a "soft" reset)
- RST Reset - The device is reset using RST/NMI pin ((i.e. a "hard" reset)
- Vcc Reset - The device is reset by cycling power to the device.

Reset Option selector allows to use one of described above reset method (***PUC only***, ***RST only***, ***Vcc only***), or ***Any.*** If ***Any*** option is selected, then at the first the PUC reset is executed. If failed, then RST medhod is used. If still failed then cycling Vcc reset option is executed.

7.3 Final Target Device action

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

The target device can be set to run an application immediately after the target device programmed. In order to do this check the ***'Reset and start the application program'*** option in the Reset Options window, shown in Figure 7-1.

7.4 Connection

Connection selector allows to select desired communication port with programming adapter (FET). Communication port is selected by MSP430.dll driver during initialization process and parameters passed to the MSP430.dll. Following string is passed to initialization MSP430 procedure.

"LPT1" when LPT 1 is selected,
"LPT2" when LPT 2 is selected,
"LPT3" when LPT 3 is selected,
"USB" when TI USB is selected.

7.5 Used Adapter

The **FET-Pro430** software is using TI's MSP430.dll library that allows communication with Texas Instruments FET (parallel port FET or USB-FET). However the ***FET-Pro430*** can also be used with the Third's Party Tool adapters. If other then the TI's adapter is used then the correct MSP430.dll library between FET-Pro430 software and used adapter must be used. In the ***Used Adapter*** group (Figure 7.1) should be selected option

TI's FET

- when the TI's FET (parallel port FET or USB-FET) is used , or parallel port FET with hardware compatible to the TI's FET.

or **Other-1**

or **Other-2**

- when the Third Party Tool adapter is used (usually USB model). In this case the MSP430.dll file location with full path should be selected using browse **(>>)** button.

Note: *Usually the FET drivers supplied by Third Party Tool supplier has name - MSP430.dll Do not copy the Third Party Tool MSP430.dll file to the location where the current TI's MSP430.dll file is located and by mistake do not overwrite this file. Using browse* ***(>>)*** *button select full path of the desired MSP430.dll file without moving it from original location to location where the TI's MSP430.dll file is saved.*

7.6 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

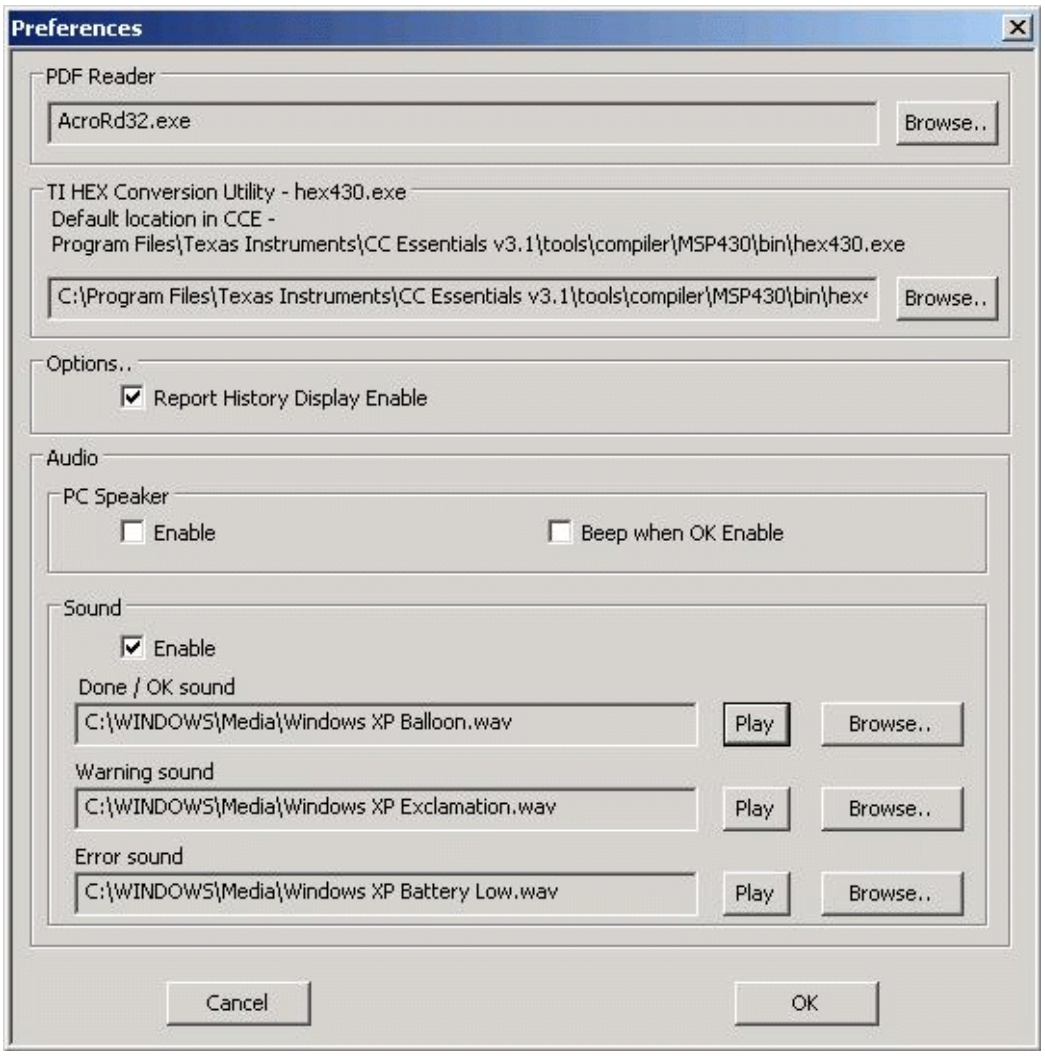

Figure 7.2

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 7.3).

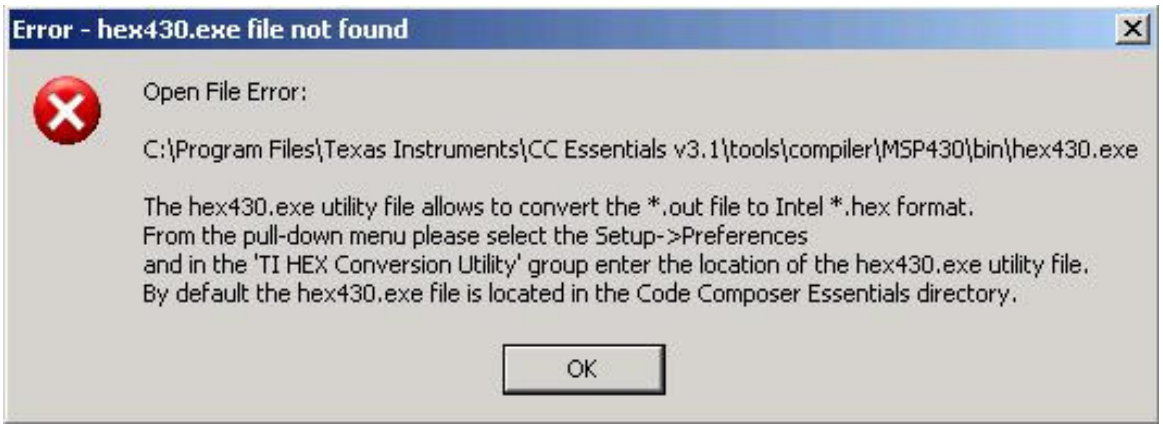

Figure 7.3

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 7.2).

8. Serialization

8.1 Introduction

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

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

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

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

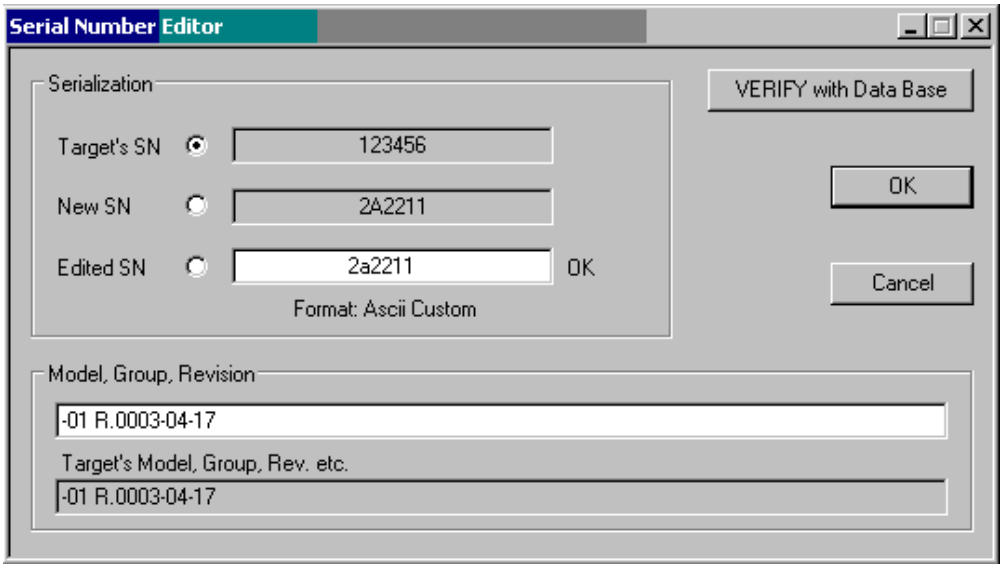

Figure 8.1-1

8.2 *Serialization Dialogue Screen*

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

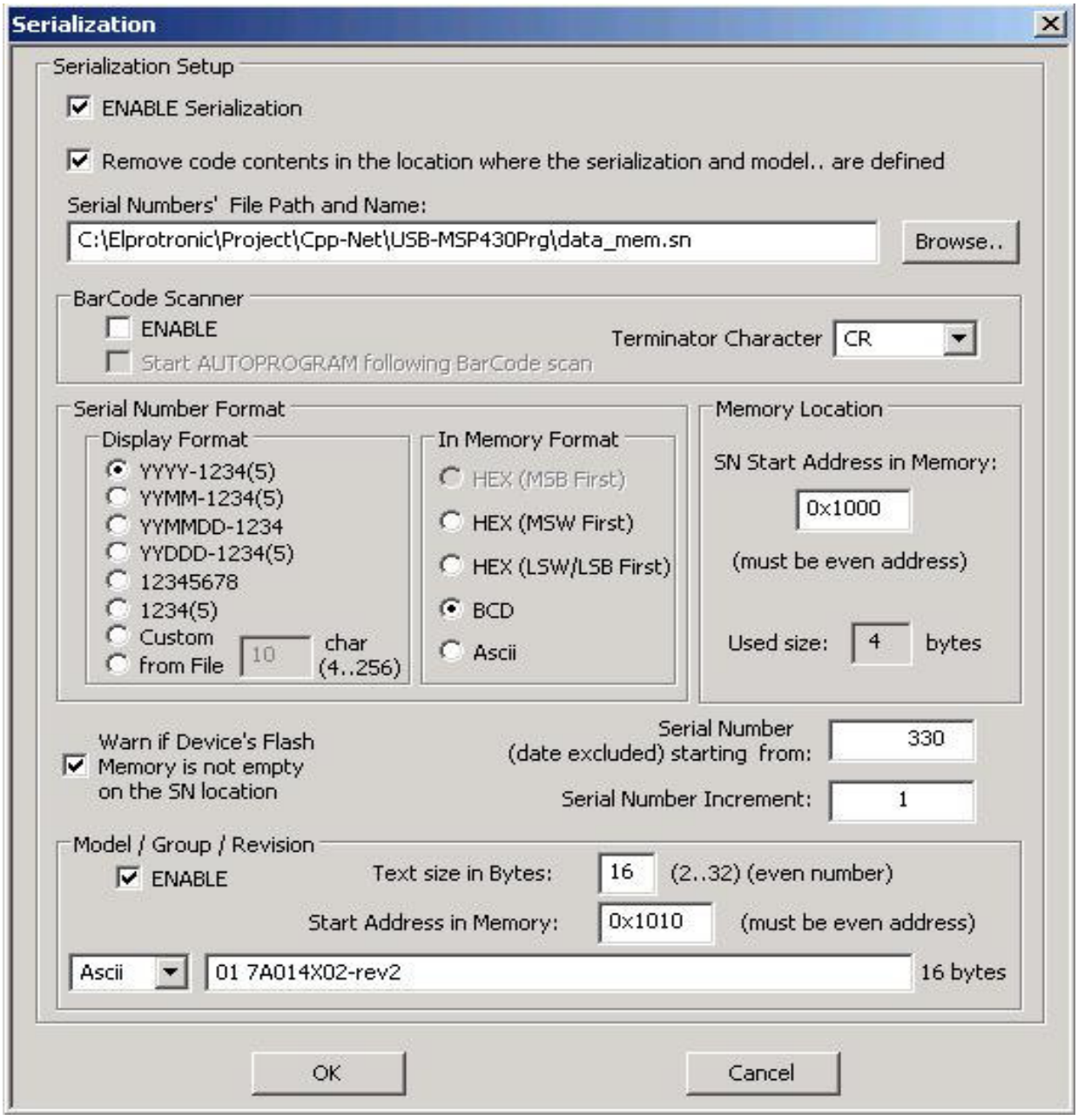

Figure 8-2

8.2.1 Serial number File

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

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

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

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

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

8.2.2 Serial number formats

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

Display Format:

1. YYYY-1234(5) -(SN Format - **F0**) Serial number has 8 or 9 characters. First four characters contain current year, and remaining 4 or 5 characters contain the serial number, eg. SN 20030123 or 200300123 has a number 0123 (or 00123) created in the 2003 year.
2. YYMM-1234(5) - (SN Format - **F1**) Serial number has 8 or 9 characters. First two

characters contain last two digits of current year, next two characters contains current month, and remaining 4 or 5 characters contain a number, eg. SN 03030123.

3. YYMMDD-1234 - (SN Format - **F5**) Serial number has 10. First six characters contain date (year, month, day of month) and remaining 4 characters contain a number, eg. 0405120123.

4. YYDDD-1234(5) - (SN Format - **F4**) Serial number has 9 or 10. First five characters contain date (year, day of year from 1 to 366) and remaining 4 or 5 characters contain a number, eg. 041230123.

5. 123456768 - (SN Format - **F2**) 8 digits serial number without date stamp.

6. 1234(5) - (SN Format - **F3**) 4 or 5 digits serial number without date stamp.

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 (MSW first) and HEX (LSW first) format

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

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

SN 200300123 will be saved as

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

In flash memory this number can be seen as

07D3 007B -> ***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 4E 61 -> ***HEX(MSW first)***

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

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

SN 12345 will be saved as

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

In flash memory this number can be seen as

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

or

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

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

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

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

E.g. entered hexadecimal number

02A569C1

will be seen as

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

or

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

8.2.2.2 BCD format

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

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

SN 20030123 will be saved as

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

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

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

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

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

SN 03030123 will be saved as

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

In flash memory this number can be seen as

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

Display Format: YYMMDD-1234 - size in FLASH - 5 bytes

SN 0405110123 will be saved as

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

In flash memory this number can be seen as

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

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

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

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

SN 041110123 will be saved as

YYDDD - 04111 (Decy) -> 0x41 0x11 (hex)

1234 - 0123 -> 0x01 0x23 (hex)

Display Format: 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 - size in FLASH - defined size in bytes

Entered manually or read via Bar Code Scanner ascii sring will be saved in flash memory **"as is".** E.g. entered hexadecimal number

02WX24S234

will be seen as

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

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

or ***HEX(LSW 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:

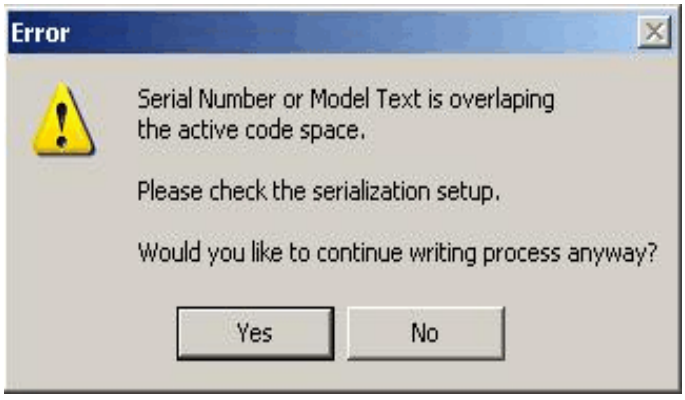

Figure 8.2.1-1

8.2.3 Model, Group, Revision

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

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

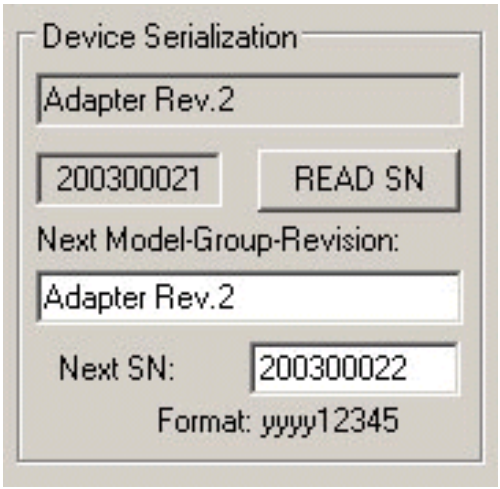

Figure 8-2

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

8.2.4 Device Serialization box

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

The '***Next Model-Group_Revision***' and *'**Next SN**'* edit lines can contain any SN and text.

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

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

8.2.5 Bar Code Scanner setup

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

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

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

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

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.

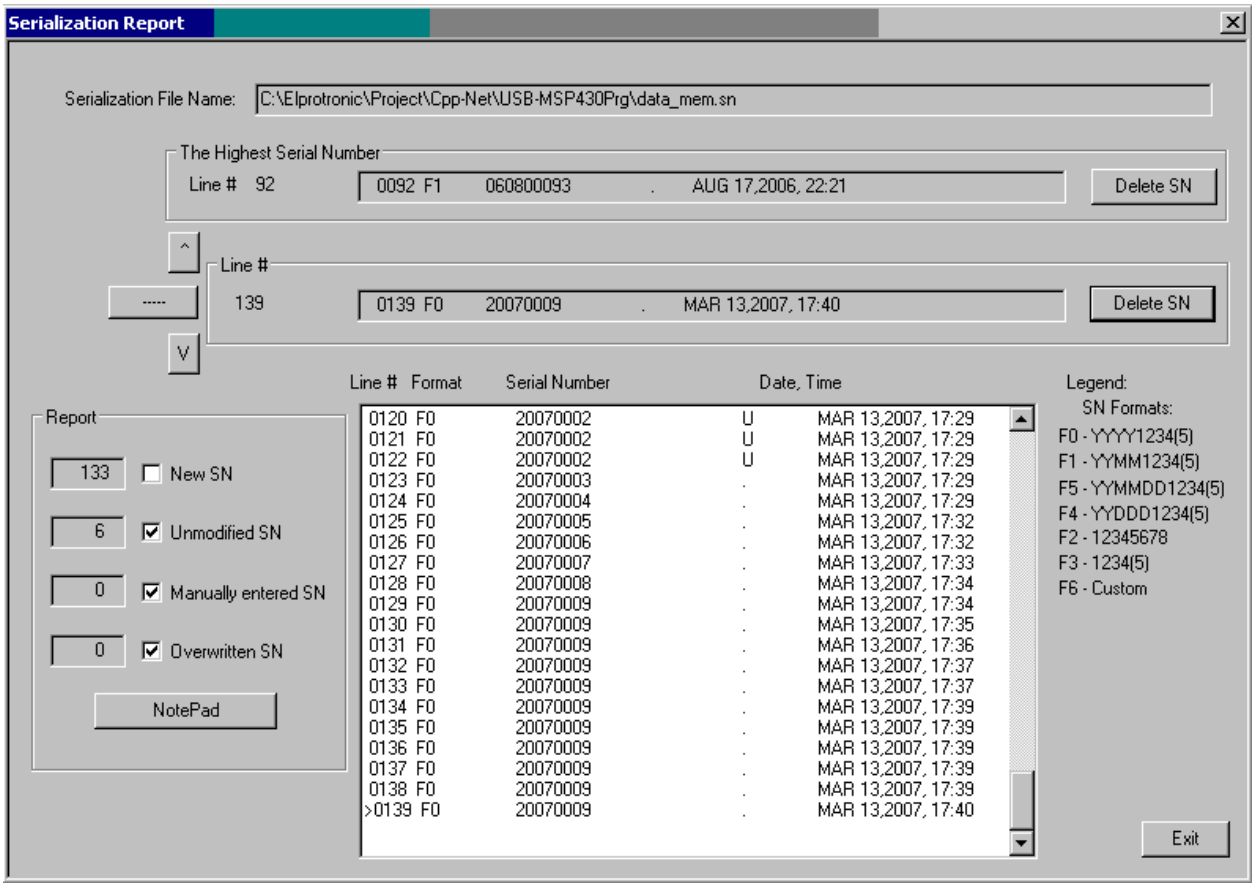

Figure 8.3-1 Serialization Report Dialog screen

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_PREFIX        string            ;optional
#SN_SUFFIX        string            ;optional
```

Serial number can contains up to 16 characters. If part of characters are the same in specified serial number list, then the repetable 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

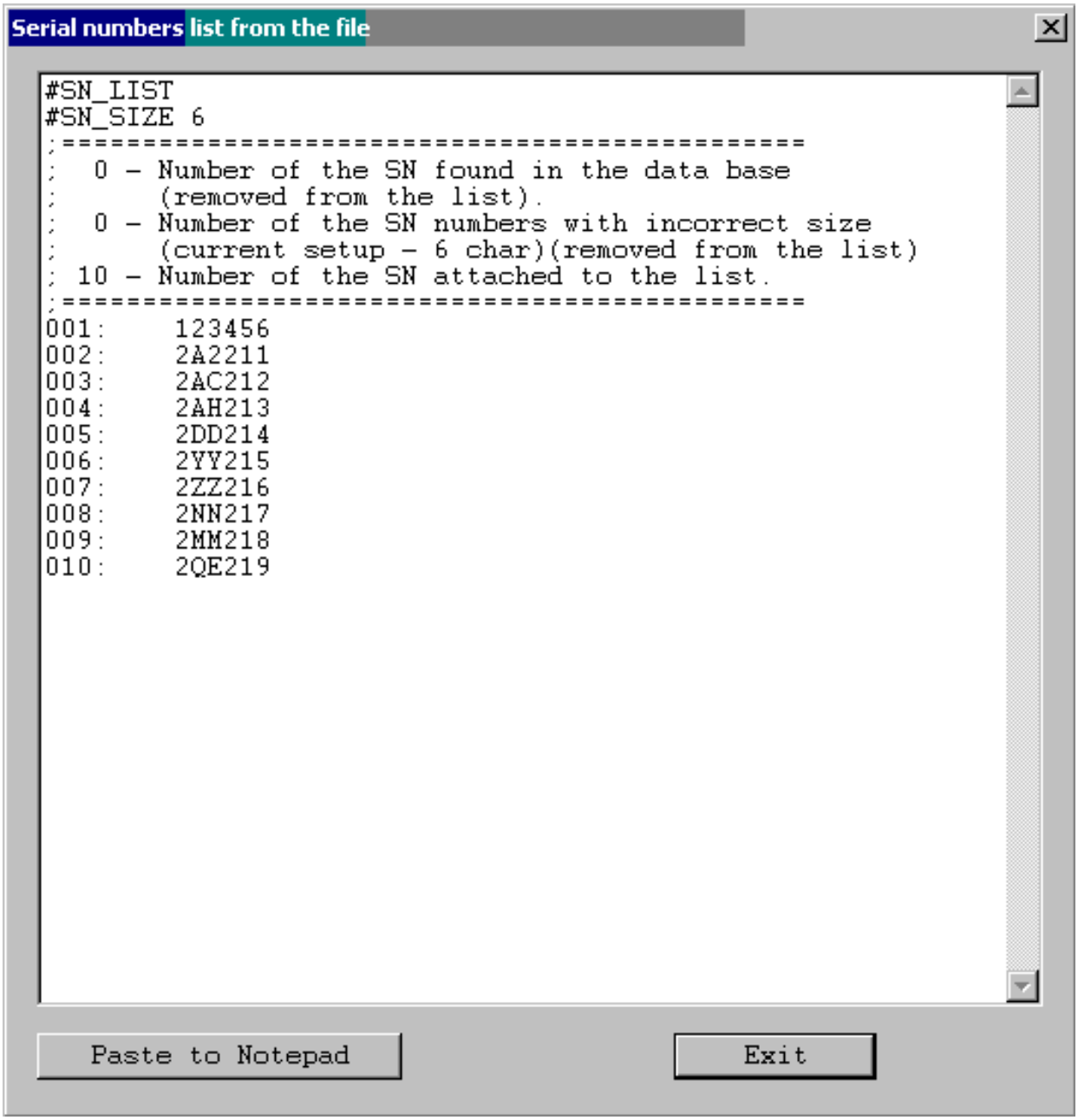

Figure 9.1

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.

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)

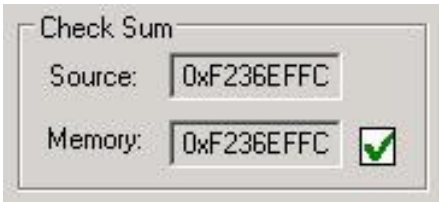

Figure 9.1

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

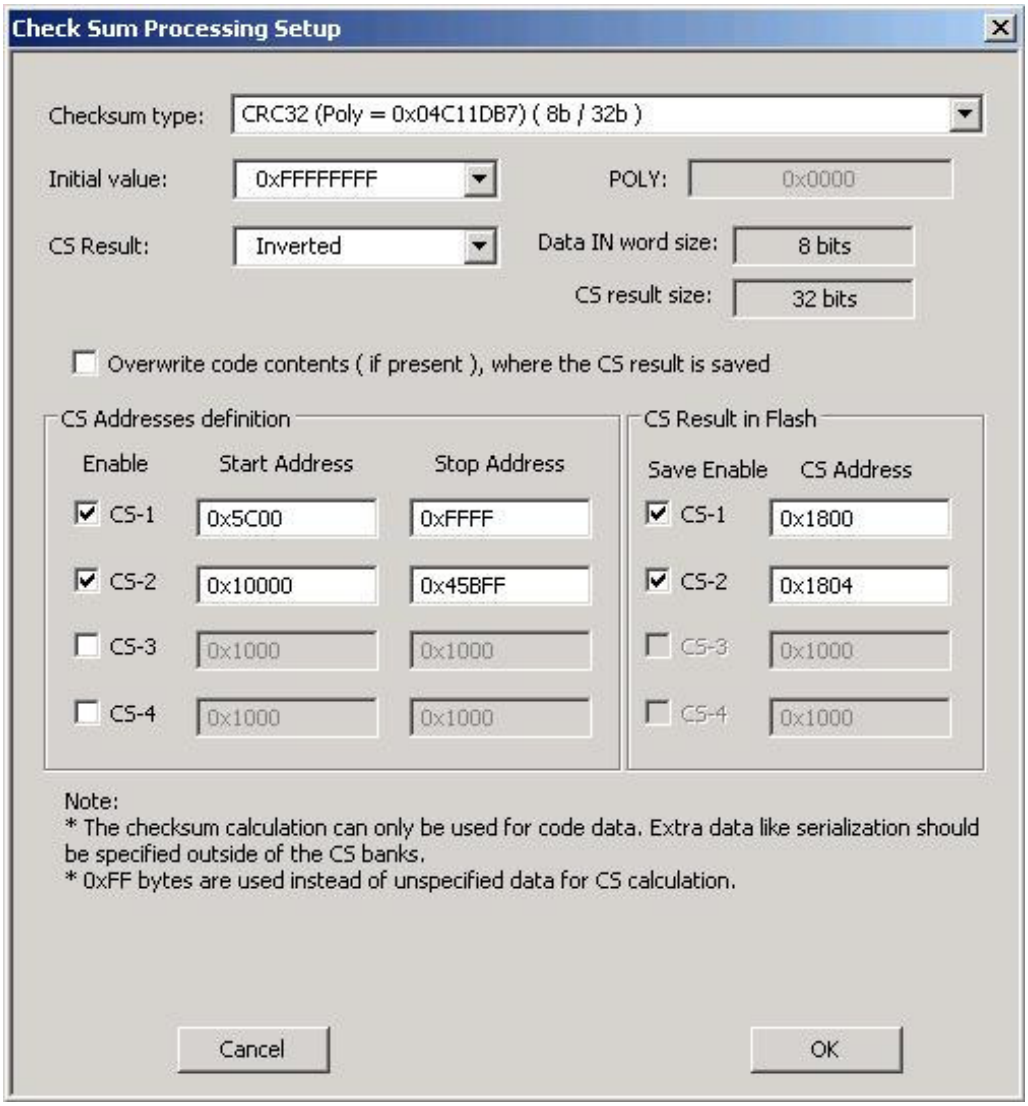

Figure 9.2

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.

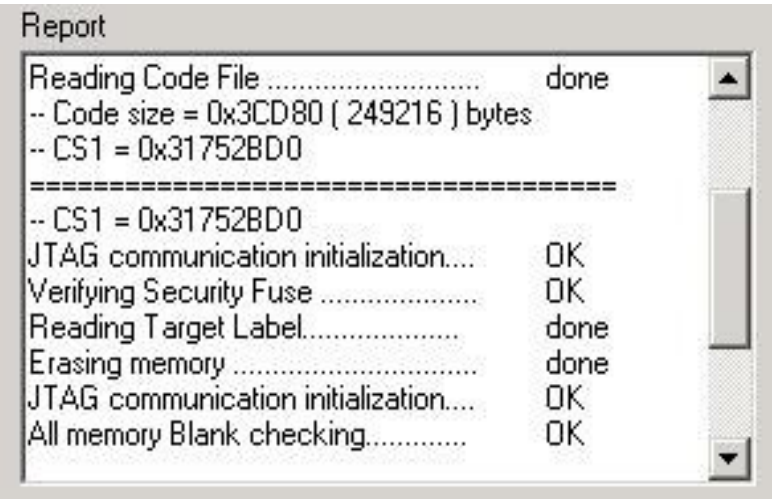

Figure 9.3

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

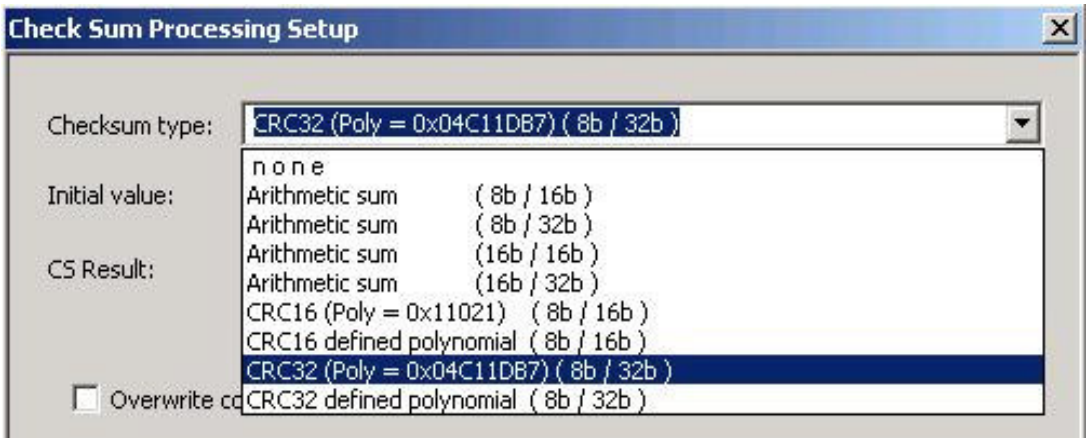

Figure 9.4

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

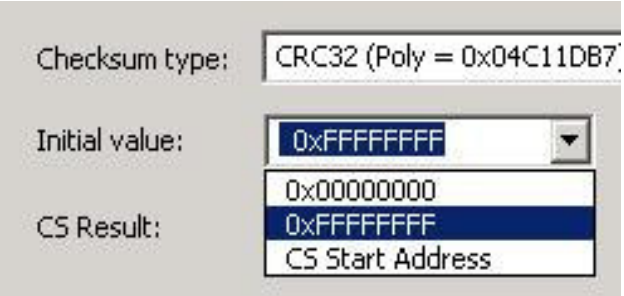

Figure 9.5

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

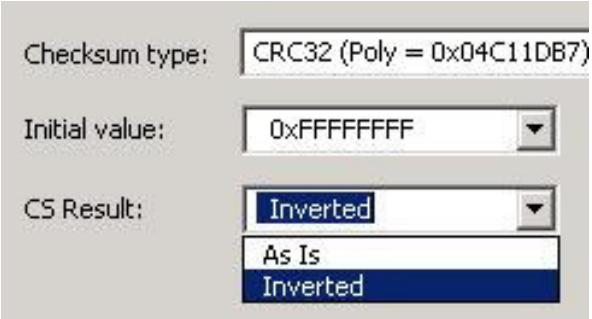

Figure 9.6

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 Start to 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.

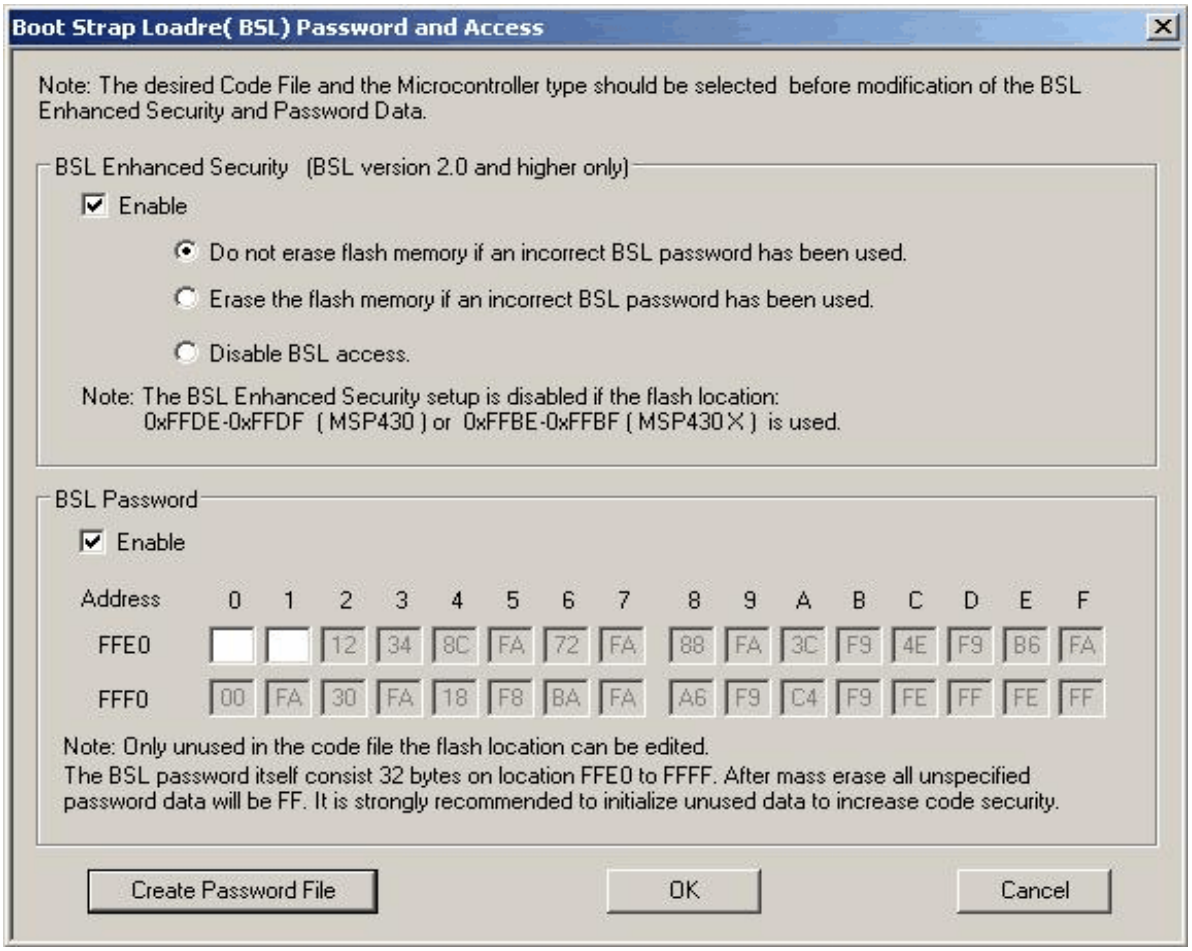

Figure 10.1

T

he ***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 passwor 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. Load/Save Setup

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.

11.1 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.

11.2 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 FET-Pro430 software cannot be used with the FlashPro430 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 10.2-1) that allows to select desired project option

Following options can be selected:

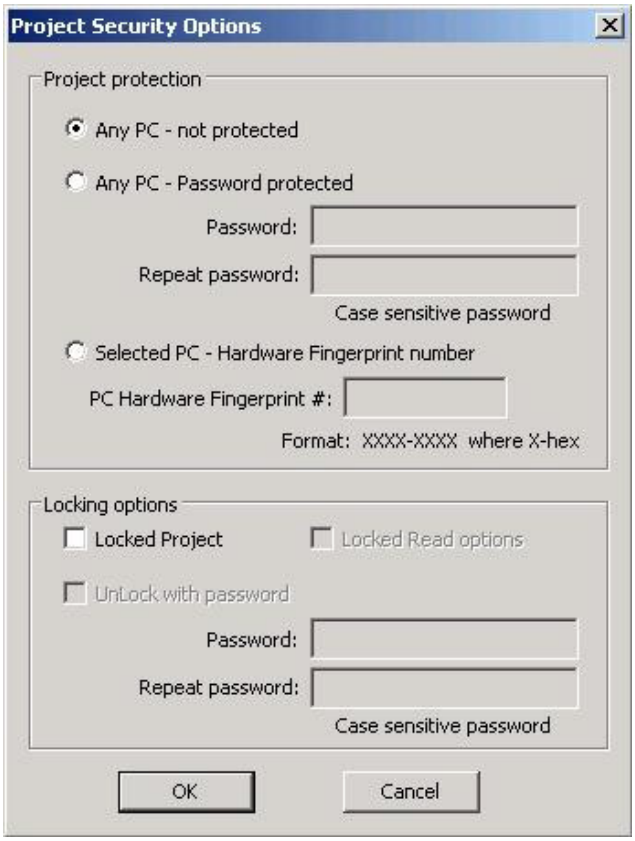

Figure 11.2-1

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 same as the number used when the project has been created. This option is useful in 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 11.2-1). 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 11.2-2)

Figure 11.2-2

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 11.2-3 is a “Project Security Options” dialogue screen with selected options

Project protected with ***PC’s “hardware fringerprint”*** number, locked and unlocked with password.

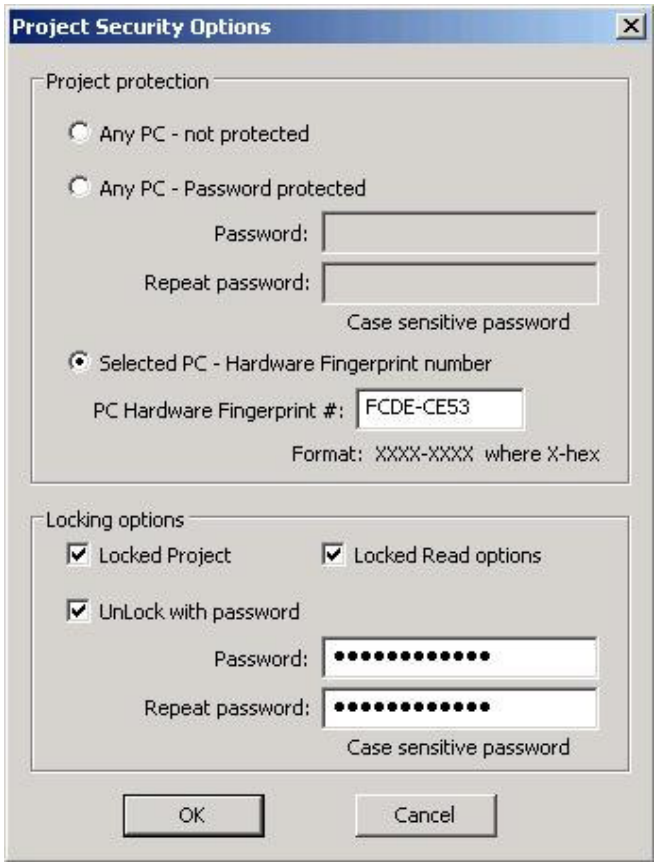

Figure 11.2-3

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.

11.3 Commands combined with the executable file

Programming executable file can be opened with project file having extension **FET430prj** eg.

FET-Pro430.exe test.FET430prj

The Folder option in Windows can also be configured to open the programming executable file (FET-Pro430.exe) when the file with extension **FET430prj** is selected. That allows to easy and fast opening the Flash Programmer with configuration taken from the *.FET430prj 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)
-nf SN_ file_name (Open Serial number list 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.*
*When the **-prj** option is used, then the **-sf, -cf,** options are ignored.*

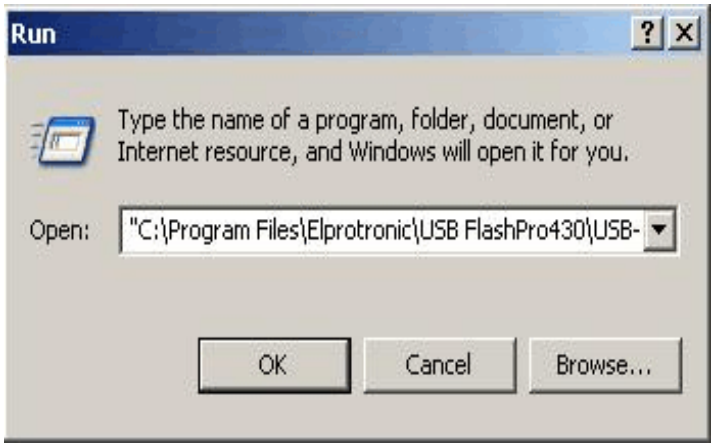

Figure11.3-1

Using Windows ***START*** button (left bottom) select ***Run..*** Using ***Browse..*** find and select executable file (see Figure 11.3-1)

"C:\Program Files\Elprotronic\FET-Pro430 Flash Programmer\FET-Pro430.exe"

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

C:\Program Files\Elprotronic\FET-Pro430 Flash Programmer\FET-Pro430.exe" -sf E:\ElproTronic\MFG\prg-04.cfg

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

“C:\Program Files\Elprotronic\FET-Pro430 Flash Programmer\FET-Pro430.exe" -lock -sf E:\ElproTronic\MFG\prg-04.cfg
or
“C:\Program Files\Elprotronic\FET-Pro430 Flash Programmer\FET-Pro430.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 11.3-3)

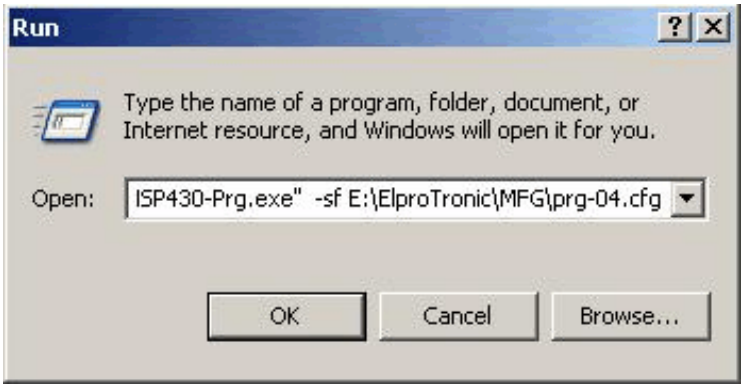

Figure 11.2

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

"C:\Program Files\Elprotronic\FET-Pro430 Flash Programmer\FET-Pro430.exe"

(see Figure 11.3-4) and at the and add the required command keys (see Figure 11.-35) eg.

"C:\Program Files\Elprotronic\FET-Pro430 Flash Programmer\FET-Pro430.exe" -lock -sf E:\ElproTronic\MFG\prg-04.cfg

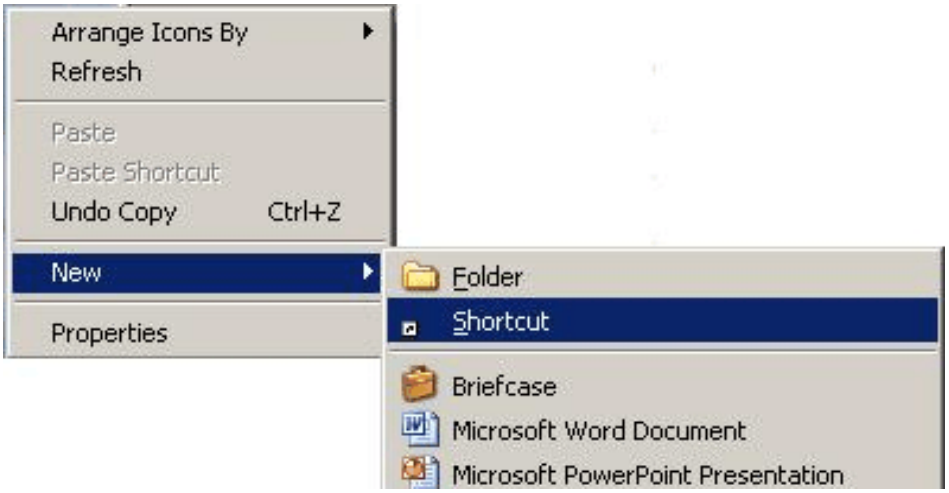

Figure 11.3-3

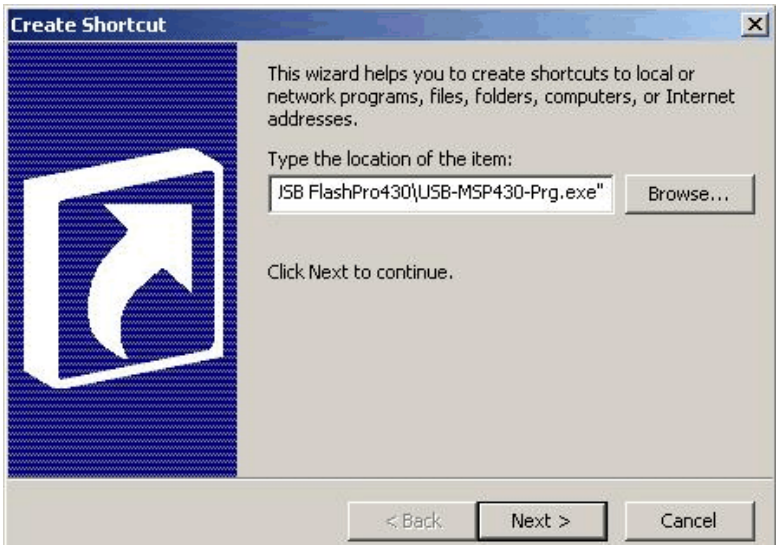

Figure 11.3-4

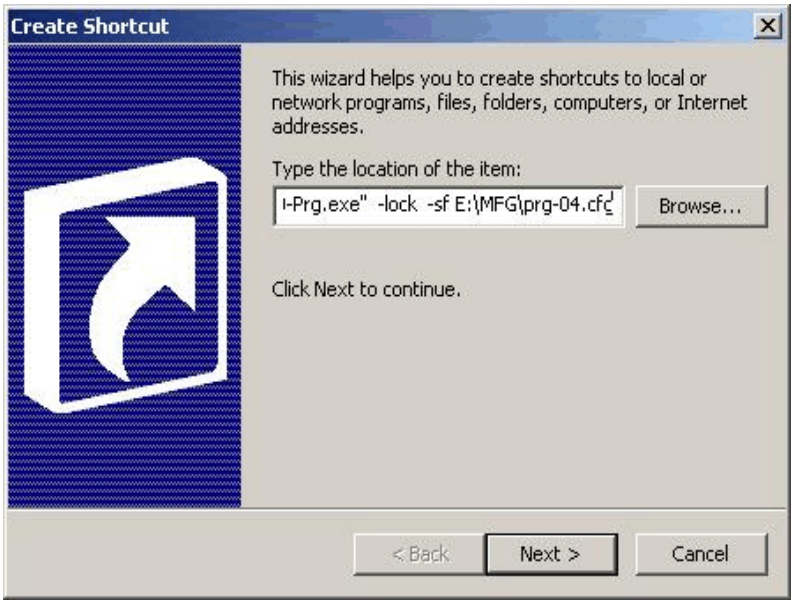

Figure 11.3-5

Click button ***Next*** and follow instruction to create icon. Using ***Copy*** and ***Paste*** and modify required configuration file names a lot of icons can be created with independent configuration setups. Clicking on the selected icon FET-Pro430 programming software will start with the selected configuration setup, and locked if required.