FlashPro2000

Flash Programmer for Texas Instruments C2000 DSPs family

User's Manual

Software version 1.49

Elprotronic Inc.

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

Copyright © Elprotronic Inc. All rights reserved.

END USER LICENSE AGREEMENT

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

1. License.

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

YOU MAY:
A.   use this Product on many computers;
B.   make one copy of the software for archival purposes, or copy the software onto the hard disk of Your computer and retain the original for archival purposes;
C.   use the software on a network

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

2. Copyright

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

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

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

FC

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) this device may not cause harmful interference and

(2) this device must accept any interference received, including interference that may cause undesired operation.

NOTE:
This equipment has been tested and found to comply with the limits for a Class B digital devices, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of more of the following measures:
* Reorient or relocate the receiving antenna
* Increase the separation between the equipment and receiver
* Connect the equipment into an outlet on a circuit different from that to which the receiver is connected
* Consult the dealer or an experienced radio/TV technician for help.

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

C E

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

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

Table of Contents

1. Introduction

FlashPro2000 programmer is dedicated to programming the Texas InstrumentsC2000 DSP family (TMS320F24xx and TMS320F28xx). Using *FlashPro430* programmer the target device can be programmed via JTAG Interface (4-wires) or via SCI-BOOT Interface..

Each programmer package (Figure 1-1) consists of a microcontroller based adapter, Windows™ based software, cable to connect the adapter with the computer's USB port and two converters with flat ribbon cables - one for JTAG connection (with 14 wires ribbon cable) and the second for the SCI-BOOT connection (with 10 wires ribbon cable). Communication speed via JTAG is selectable - up to 3Mb/s or 1 Mb/s - and allows use of longer JTAG cables if required - useful in production. To facilitate high speed communication via SCI-BOOT Interface, a new Fast Boot Loader is temporarily downloaded to RAM of each programmed device, that allows to communicate with target device with speed up to 500 kb/s.

Figure 1-1

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

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

2. Features

Major features of the *FlashPro2000* programmer are:

* Support TMS320F24xx and TMS320F28xx flash-based devices.
* Programmer has a **unique feature** - two interfaces in one package allowing to program the target device via **JTAG** (one or more devices in the chain) or via **SCI-BOOT** Interface.
* The TI's target device specific Flash API are used to erase and program the DSP's flash. All API procedures are build-in to FlashPro2000 software and the required procedure is selected automatically when the desired DSP type is selected. That allow to simplify usage of flash programmer. User does not need to know what API should be selected. When the DSP is programmed the used API version is displayed in the report window. When the new DSP type would be released, then the new FlashPro2000 with new Flash API will be released. Upgraded software is available on our website
     http://www.elprotronic.com/download.html
* Programming speed via JTAG and SCI-BOOT are almost the same.
* Write CSM Security Password capability.
* No code size limitations.
* Full memory or sector memory erase capability.
* Write Check Sum verification.
* System Clock verification.
* Easy to use Windows™ based software.
* Programmer accept TI (*.txt), Motorola (*.s19), Intel (*.hex) and CCS (*.out) data files for programming.
* Lock setup capability with encryption feature, 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 or OTP memory in HEX, BCD or ASCII format. Log file capability allowing to review information about the flashed target devices.
* **DLL** software package can control programmer from other programs.
* Programmer has been fully tested to comply with the **FCC** and **CE** requirements.
* Our programmers are inexpensive - for users interested in basic features we provide limited, or lite software version.
* Using USB-1.1 (12Mbits/s) Port to communicate with the Programming Adapter.
* Communication with the target device via JTAG Interface using TI-standard 14-pins header connector and via SCI-BOOT Interface using customized 10-pins header connector.

2.1   Customized features

FlashPro2000 programmer can be controlled from external software or programming sequences can be customized. These features are very useful in production environment. Standard programming software FlashPro2000 has a lot of options described above, but of course it can not cover all customer's requirements.

2.1.1 Encrypted Project option

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

2.1.2 Script file

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

2.1.3 DLLs

When the customized programming sequence is not covering customer's requirements, then an attached to software package DLLs can be used. DLLs allows to fully control programming adapter from external software written in MS Visual C++, MS Visual Basic, LABView, DOS or other programming packages like Borland C++ etc. See *"FlashPro2000 - Remote Control Programming User's Guide"* for details.

2.1.4 Self Test Program

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

3. Getting Started

The *X-Pro430* programmer package contains (see Figure 1.1):

1.   One READ ME FIRST document.
2.   One Flash Programmer CD ROM (Software + Manual).
3.   One *FlashPro2000* (USB FPA 4.4 or higher
) Flash Programming Adapter

.
4.   One 6 feets length USB-A to USB-B cable
      extender.

5.   One C2000-JTAG (PE034X02) adapter with 14-
      wires flat ribbon cable.

6.   One C2000-SCI-BOOT (PE034X01) adapter
      with 10-wires flat ribbon cable.

3.1   Software Installation

The *FlashPro2000* Programming Software runs on PC under Windows ^{TM} ME, WinNT, 2000, XP
(32 or 64-b), VISTA (32 or 64-b) or Win-7 (32 or 64 b). Follow instructions below to install the
software:

1.   Insert Software CD into your CD-ROM drive.
2.   The Setup wizard appears automatically. Click *FlashPro2000* button to start the installation
      process (Figure 3.1-1).
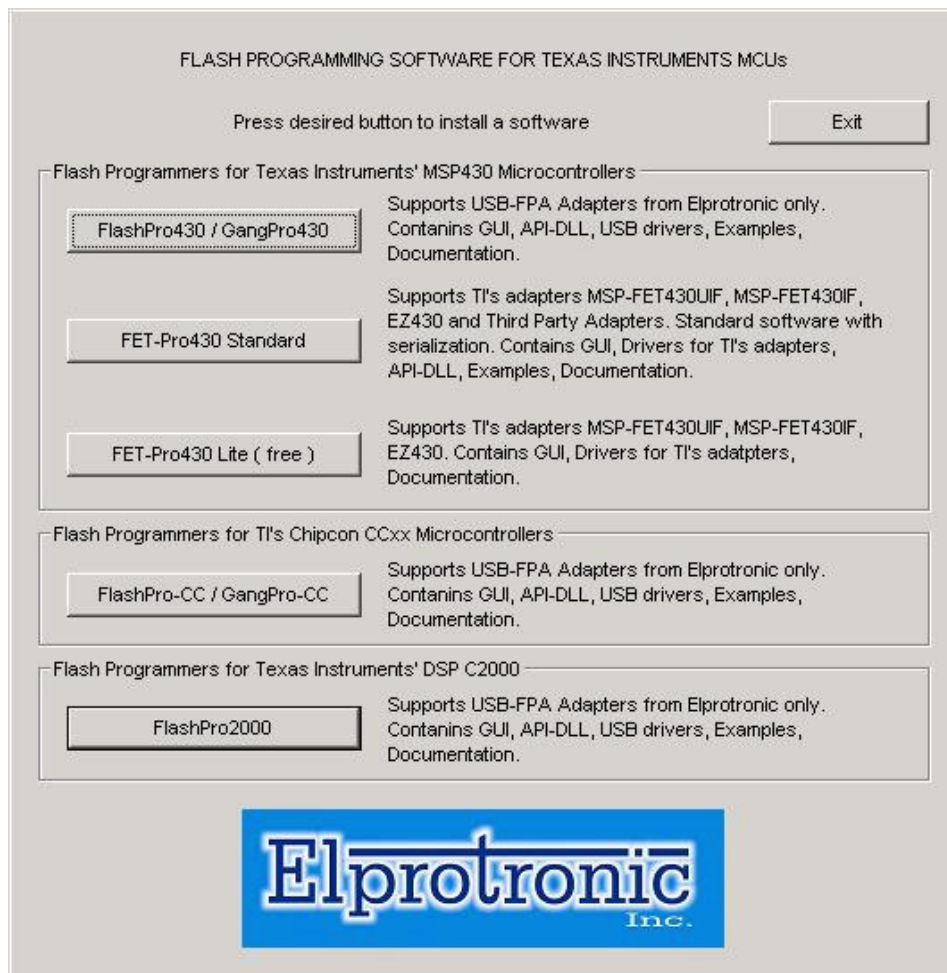
Figure 3.1-1

3.   If the Setup wizard does not start automatically, click the Start button and choose the Run dialog box. Type "D:\CD-entry.exe", where D represents the drive letter of your CD-ROM drive. Then click the OK button.

4.   Once the installation program starts, on-screen instructions will guide you through the remainder of the installation. You **must** accept licence agreement before using software.

3.1.1 Driver Installation

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

Windows\inf

and

Windows\system32\drivers

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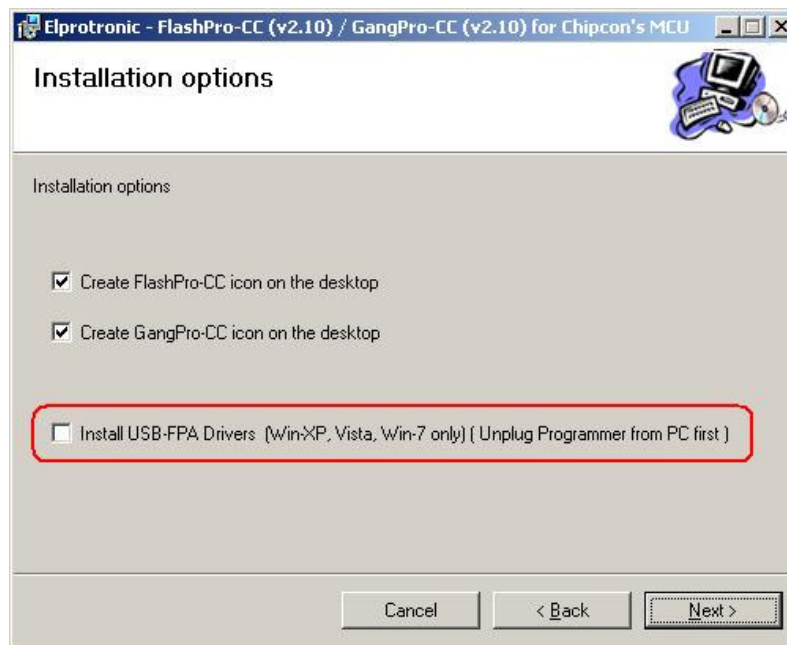
Figure 3.1-1

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

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

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

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

C:\Program Files\Elprotronic\Drivers USB-FPA

directory if you find any problems.

3.2   Hardware Setup

FlashPro2000 software support the USB-FPA versions 4.4 adapter and can communicate with target device via JTAG interface, or via SCI-BOOT interface. When the JTAG interface is used, then it should be plug-in the C2000-JTAG adapter between standard TI's JTAG connector and Programming Adapter FPA (Figure 3.2-1) .

1.   Connect USB-FPA Flash Programming Adapter to the PC USB Port, using provided cable extender (USB-A to USB-B).
2.   Plug in C2000-JTAG adapter to FPA. Plug-in 14-wires ribbon cable between C2000-JTAG Adapter and header connector on your device board (Figure 3.2-1). Make sure that pin 1 on your device board's header is connected to pin 1 of the socket connector. Pin 1 is marked as a red cable on the ribbon cable.

When the SCI-BOOT interface is used, then the C2000-SCI-BOOT Adapter should be plug-in to FPA and 10-wires ribbon cable used for communication with target device. See "Adapter description" chapter in this manual for detailed information how to connect the SCI-BOOT interface with target device.
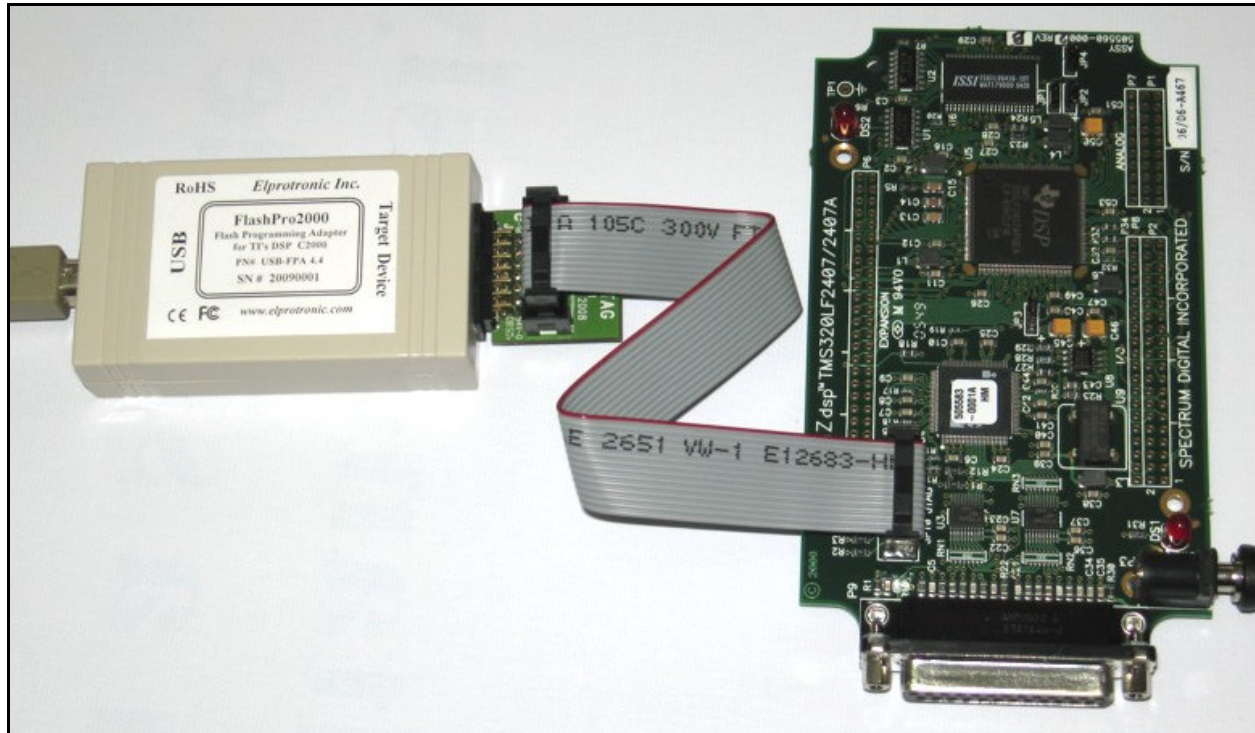
Figure 3.2-1

3.3 Starting up "FlashPro2000" Flash Programmer

To start the *FlashPro2000* Flash Programmer click on the Elprotronic FlashPro2000 icon.

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

3.4 X-Pro430 Selector

The *Flash Programming Adapter (FPA)* has Multi-USB feature. Up to 16 Flash Programming Adapters can be connected to one PC. Each adapter can be controlled by one opened software application. Up to sixteen application software can be opened at the same time. Each application software can have independent setup from the other application software setup (code file, controlled microcontroller type etc.)

Figure 3.3-1

When more then one *X-Pro430* adapter is connected to PC then following *X-Pro430* selector dialog screen will be displayed on the PC screen (see Figure 3.4-1). Using available buttons the one desired Flash Programming Adapter should be selected. Make a sure, that selected *X-Pro430* programming adapter is not used by other opened application.
Selected *X-Pro430*'s serial number will be displayed on the left bottom side of the programming dialog screen. The *FlashPro2000* programming software supports all *X-Pro430* programming adapters. *FlashPro2000* programming software can be used to access a single target device, regardless type of the used FPA. Using the *FlashPro2000 FPA* (model USB-FPA-4.4) the target device can be connected to FPA via JTAG or SCI-BOOT.
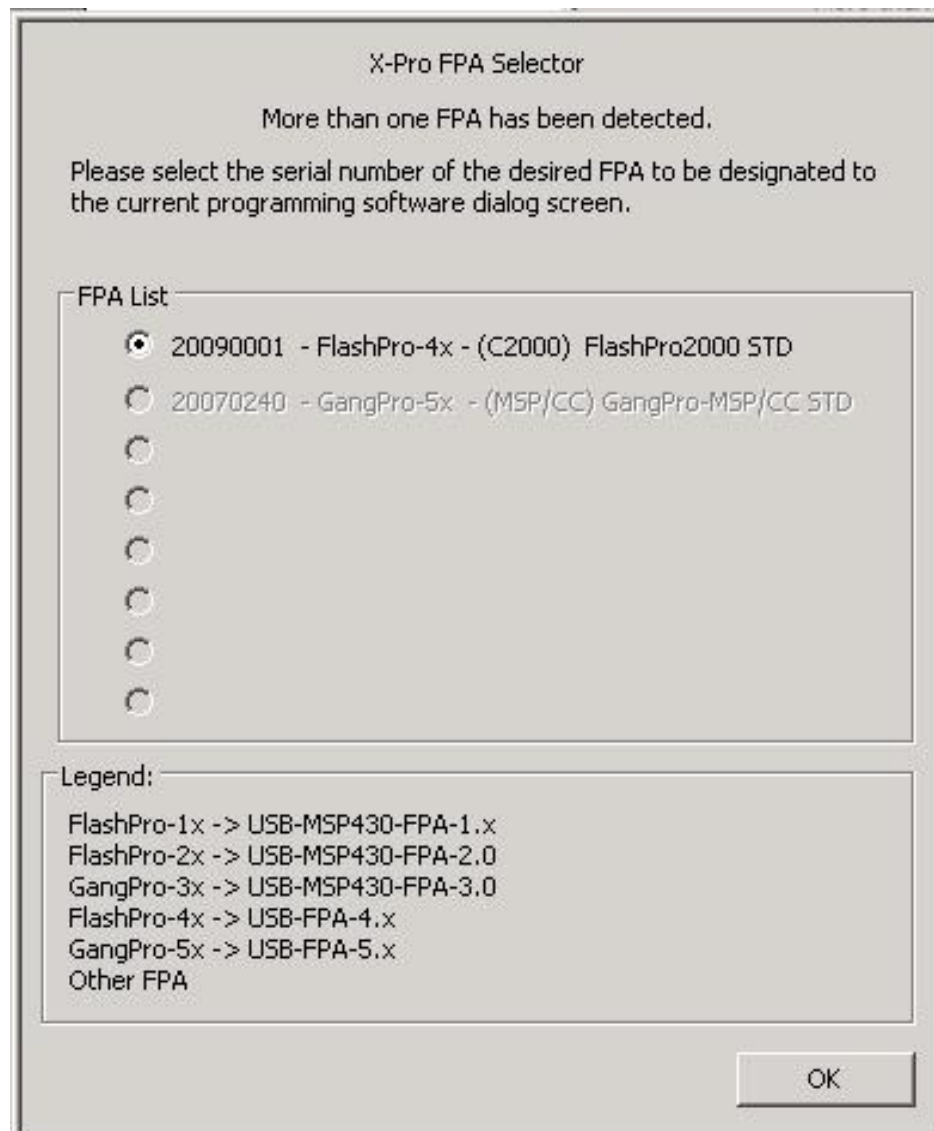
Figure 3.4-1

4. Programming Dialog Screen
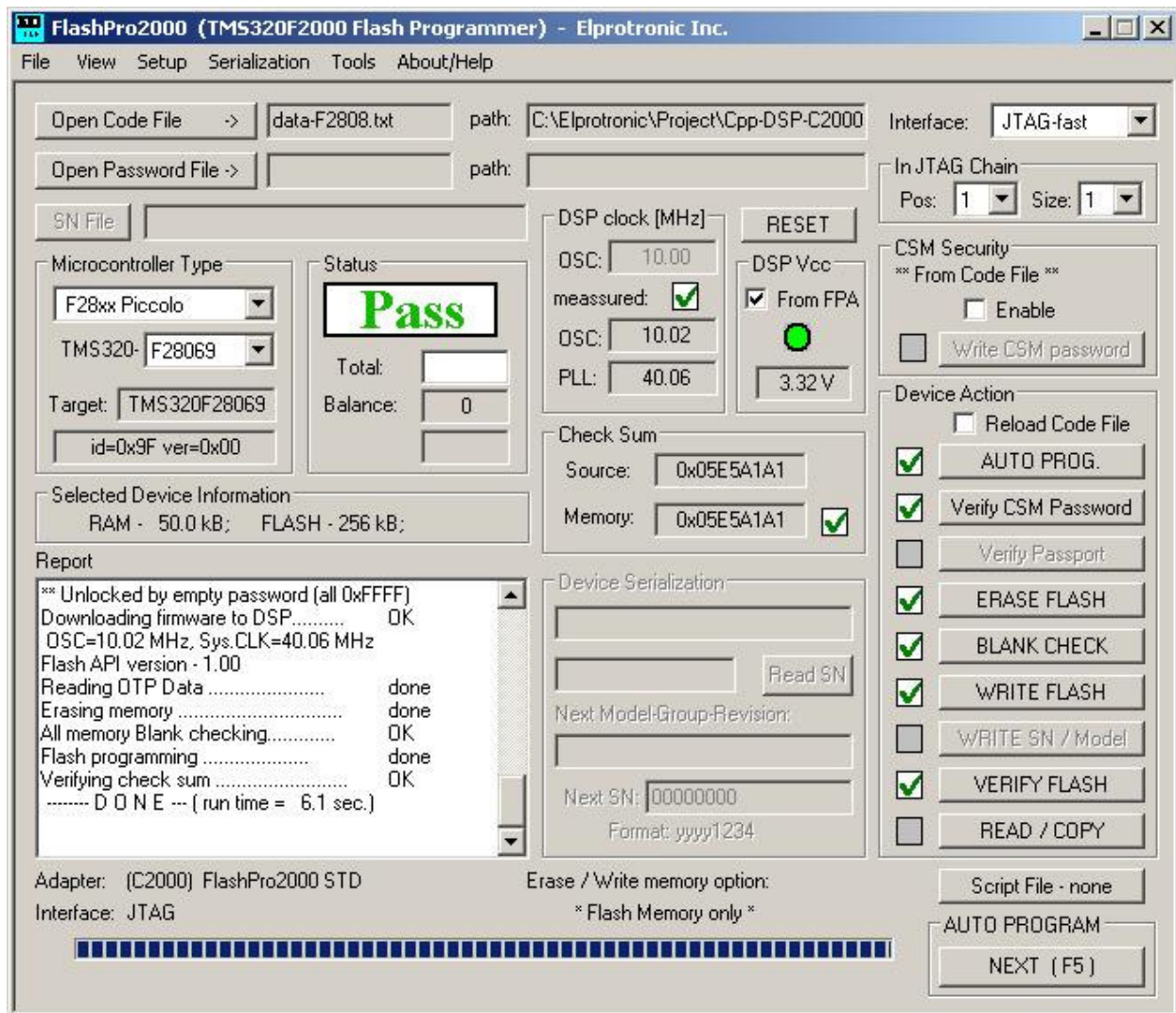
Figure 4-1. Programming dialog box screen.

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

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

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

4.1   Interface Type

The communication interface type - *SCI-Boot, JTAG-fast* and *JTAG-slow* can be selected in the Interface group. Proper communication interface should be selected, otherwise communication with the target device can fail. When the JTAG communication is used then the *JTAG-fast* interface should be selected. If your target device has installed suppressors, capacitors or EMI filters in the JTAG lines or used JTAG cable is long that can degradate the JTAG communication speed, then the *JTAG-slow* interfaces should be selected.
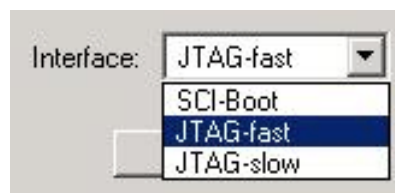
Figure 4.1

4.2   Microcontroller Type

The microcontroller type can be selected from the pull down field of the processor type group. The pull down field contains a list of all microcontrollers supported by *FlashPro2000* software

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:**' (figure 4.2-2). This allows the software to warn you if the connected microcontroller does not match the one specified by the user.
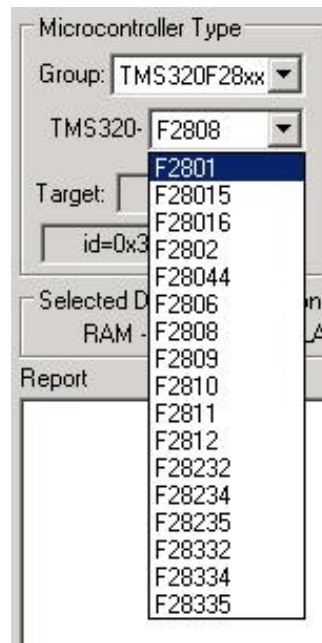
Figure 4.2-1
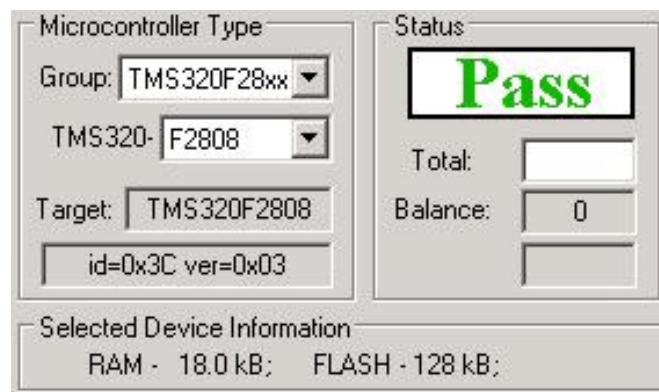
Figure 4.2-2

When the selected and detected MCU type are not the same then following error is displayed.

Figure 4.2-3

Procedure is terminated. You should select correct name of used microcontroller and try again.

4.3   Code File Management

FlashPro2000 flash programmer provides a few options to manage code files. These options allow the user to open a code file, convert to other format and save the programming data into a code file. Following code formats are supplied - all are 16 bits data width with extended addressing mode (over 16-bits addressing) - Texas Instruments *.txt, Intel *.hex, Motorola *.s19, *.s28, *.s37 and TI's CCS (from Code Composer Studio) *.out formats.
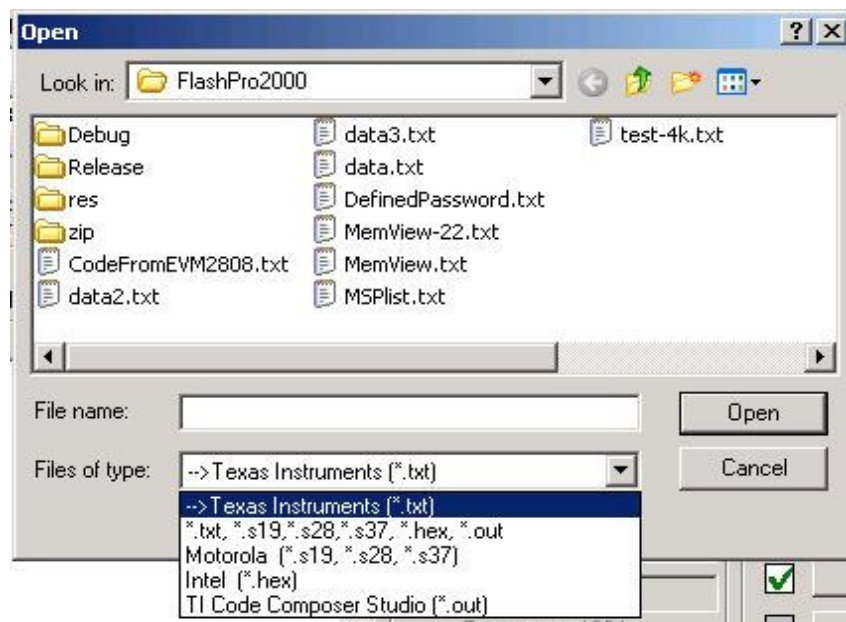
Figure 4.3-1

The *Open Code File* button, or the *Open Code File* from the **FILE** pull down menu, prompts for opening the object file that contains the code data, as shown in Figure 4.3-1. When the file is selected the contents of the object file are downloaded into the PC memory. If the selected microcontroller does not have enough memory to fit the data contained in the code file, the warning message in Figure 4.3-2 will be displayed and more detailed information displayed in the report window (Figure 4.3-3).

Figure 4.3-2

Figure 4.3-3

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 dialog box screen). Check sum calculated from the code file will be displayed in the *"Check sum - Source"* window. Contents of the selected file can be viewed by the selecting of *'Code File Data'* from the *'View'* menu (see chapter 5).

The *Save Code File* option saves the data currently contained within the PC code data block into a 16 bits data width code file. When the user selects this option from the File menu, the window in Figure 4.3-4 will appear, prompting for the name of the file to be created.

All of the aforementioned Code File options work with three most popular code file formats. These formats are the Texas Instruments, the Motorola and the Intel file formats. *FlashPro2000* will work with any of these formats and will easily convert one file format to another by using the Open Code File and Save Code File options.
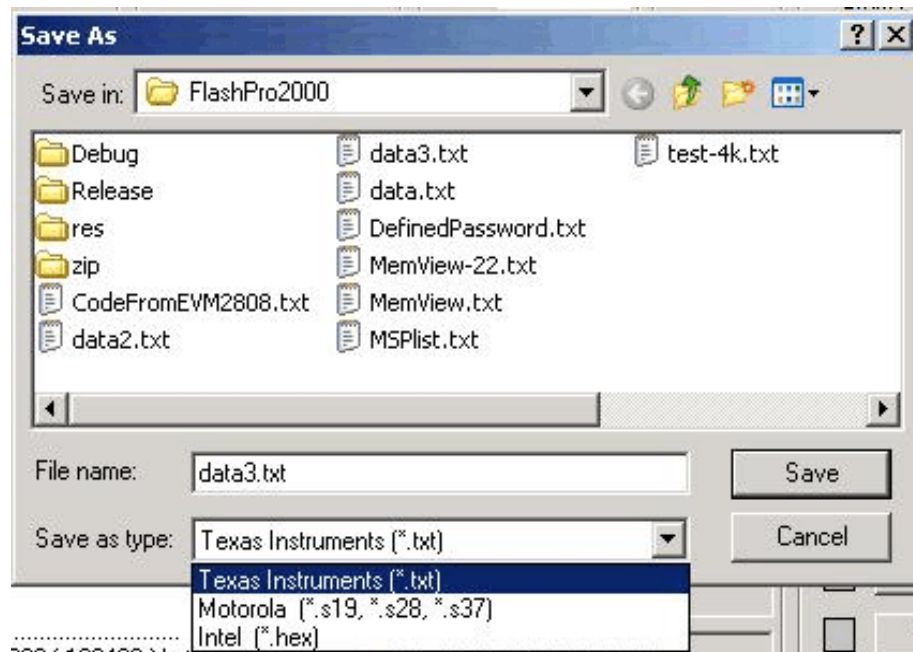
Figure 4.3-4

4.4   CSM and Security Password

The F24xx and F28xx microcontrollers allows to protect the user firmware from being reverse-engineered. The security features a 64-bit (F24xx) or 128-bits (F28xx) password which the user programs into the flash. One Code Security Module (CSM) is used to protect the flash/OTP and the L0/L1 SARAM blocks. The security feature prevents unauthorized users from examining the memory contents via JTAG or SCI-BOOT port, executing code form external memory or trying to boot-load some software that would export the secure memory contents. To enable access to the secure block, the user must write the correct 64-bits (F24xx) or 128-bits (F28xx) "KEY" value with matches the value stored in the password location within the Flash.

The FlashPro2000 provide flexible secure password support, that allows to easy manage the old and the new passwords used in the programmed devices. Up to four secure passwords are used for unlocking the target device.

1. default password - all 0xFFFF
2. Password extracted from the code file
3. Password extracted from the password file
4. User defined password.

Contents of the all passwords can be viewed in the *CSM Security Password* dialog screen (Figure 4.4-1) (access from the pull-down menu - *View-CSM Passwords* for viewing passwords only, or from *Setup->CSM Password* for viewing and modifying password setup).
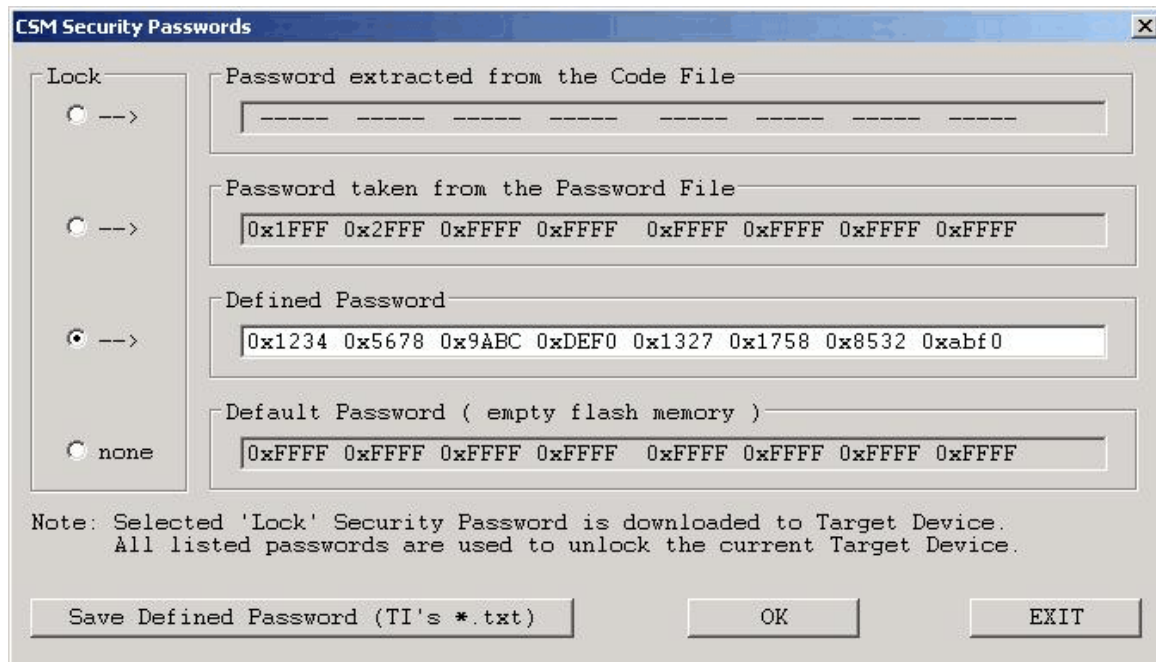
Figure 4.4-1

In the most natural scenario the code file contains the security password that would be downloaded to the flash and the target device before programming is blank. In this case the blank security password (all 0xFFFF) can be used to unlock an access to target device. After programming, the target device contains the new security password and the empty security password cannot unlock the access to programmed device. In this case the security password taken from the code file can unlock an access to target device. But if the target device is not empty before programming then it can have the security password written in flash other then the password taken from the code file, and other than the empty password. To cover this case one more password can be loaded to the passwords collection - password taken from the old code file downloaded to target device. When the *Open Password file* button is used, then from the whole code file are taken only the data containing the security password. All other data are ignored. For flexibility, one more defined security password is added. The defined password allows to enter any data that can be used as the security password - for unlocking or for locking the target device. All specified security passwords are used during unlocking an access to target device. As the first up the default password (all 0xFFFF) is used. If unlock failed, then the next security password is used until access to target is unlocked. In the report window is displayed information which the security password unlocked the target device (Figure 4.4-2)
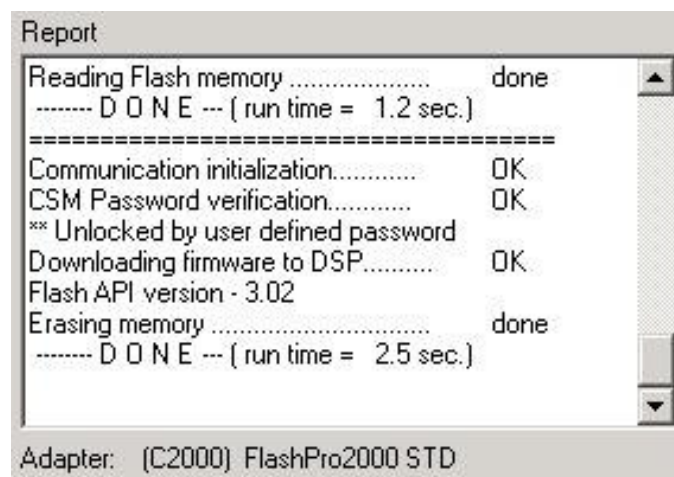
Figure 4.4-2

If none from the available security passwords are correct, then access to target device is blocked and target device can not be programmed. Following message is displayed in the report window (Figure 4.4-3)

Figure 4.4-3

When target is unlocked that to device can be erased and programmed. The security password for locking the target device can be selected from four listed above security passwords. By default the security password taken from the code file is used for locking the target device. But it can be modified if required by selection of the desired password in the *Lock* group located in the *CSM Security Password* dialog screen. Also - the Security Password option must be enabled in the main dialog screen (Figure 4.1 and 4.4-4)

Figure 4.4-4

In the *CSM Security* group is displayed an active Security Password (in this case the Security Password is taken from the Code File). When option - Enable is not selected, then the Security Password is disabled and the location of the Security Password in Flash is not saved. By default - when flash is erased then the flash contains empty password (all 0xFFFF).

When the *Enable* option in the *CSM Security* group is selected, but contents of the selected Security Password is empty or contains all 0xFFFF, then the following message is displayed when the target device is programmed (Figure 4.4-5)

Figure 4.4-2

In this case the *Enable* option should be disabled or selected Security Password contents corrected.

4.5 Power Device and Clock frequency test

The programming adapter is monitoring the Vcc taken from the programmed device. On the main dialog screen the value of the Vcc is displayed . When the Vcc is higher then 3.0V the green "LED" is displayed (Figure 4.5-1) . Communication with target device can be established. When the measured Vcc is below 3.0V then the following message is displayed when the communication with target device is initialized (Figure 4.5-2).
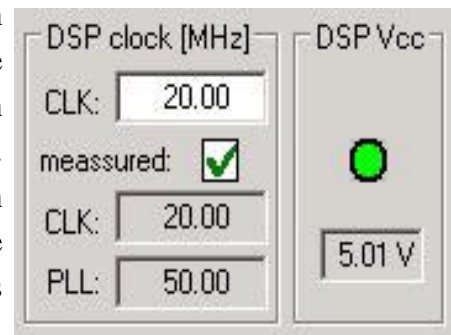
Figure 4.5-1

The programming procedure times used in the Flash API using the external clock frequency supplied to DSP. The frequency of this used clock must be specified first. The FlashPro2000 is measuring the frequency of the external clock connected to DSP and displaying the CLK frequency

Figure 4.5-2

and the system frequency (PLL frequency) in the main dialog screen in the ***DSP clock [MHz]*** group (Figure 4.5-1). For double checking it is required to enter the expected CLK frequency in the CLK field . If the measured frequency is not the same as the specified frequency with tolerance +/- 5%, then programming procedure is terminated, error displayed in the CLK group (Figure 4.5-3) and
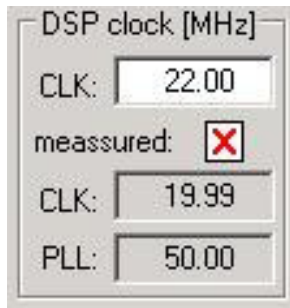
Figure 4.5-3

Figure 4.5-4

following message displayed in the report window (figure 4.5-4). The edited CLK data should be corrected or hardware should be checked if the real frequency is the same as the expected CLK frequency.

4.6   *Device Action box*

Device Action box contains 8 buttons (Figure 4.6-1) 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 measuring the Vcc supplied to target device. When the DC voltage level becomes higher then 3.0V, the communication with the target device is initiated via JTAG or SCI-BOOT Interface. When the JTAG/SBW Interface is selected then the security fuse is verified, if access to the microcontroller is available. When the BSL Interface is selected, then the password is verified to unlock access to the microcontroller and the Fast BSL is downloaded to the target device. Once the specified action is completed successfully the green check mark will appear. Also, the device will return to the state it was in before the action was executed.
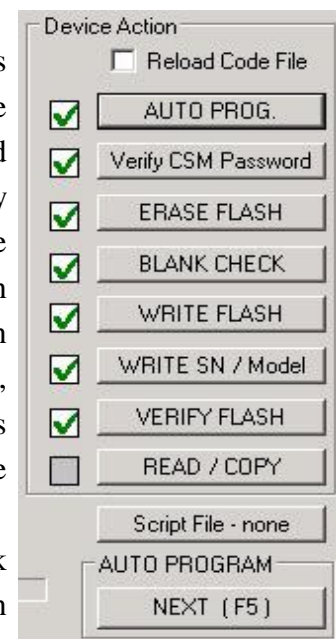
Figure 4.6-1

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.4.6.2). 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.4.6-3) as Active (blue), Pass (green), or Fail (red). On the bottom of the programmer dialog 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.

```
Report
┌──────────────────────────────────────────────┐
│Communication initialization............    OK    ▲│
│CSM Password verification............    OK      │
│** Unlocked by password from Password file      │
│Downloading firmware to DSP.........    OK       │
│Flash API version - 3.02                         │
│Erasing memory ...........................    done  │
│All memory Blank checking............    OK      │
│Flash programming ....................    done   │
│Verifying check sum .......................    OK  │
│ -------- D O N E --- ( run time =  10.0 sec.)   │
│                                              ▼│
└──────────────────────────────────────────────┘
```

Figure 4.6-2

```
┌─Status──────────┐
│  ┌───────────┐  │
│  │  Pass     │  │
│  └───────────┘  │
│  Total:  │  22 │ │
│  Balance: │  21 │ │
│  SN qty  │   9 │ │
└─────────────────┘
```
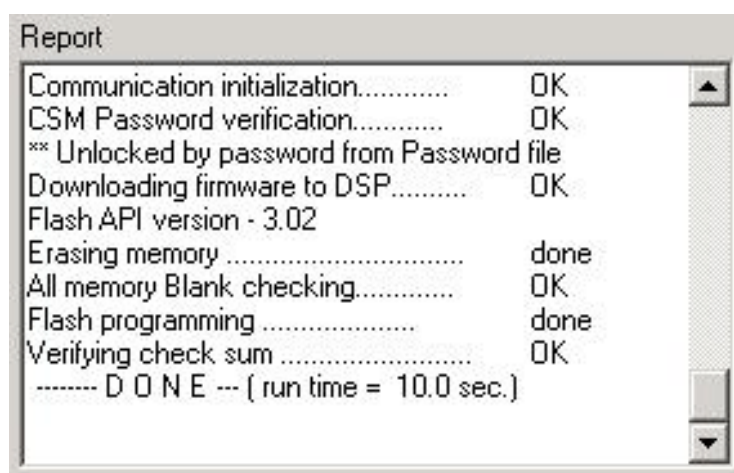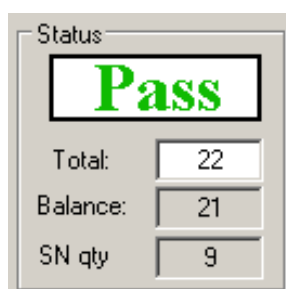
Figure 4.6-3

4.6.1 Auto Program button

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

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

In the report window you can see a typical report message during the Auto Program procedure (see Fig. 4.6-2).

Status window (see fig. 4.6-3) 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. When the serialization from file option is used , then in the bottom line is displayed the number of the serial numbers left in the list.

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

4.6.2 Verify CSM Password button

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

4.6.3 Erase Flash button

This button enables the flash memory segments, or mass (all) memory to be erased. If any option other then *'Erase All Memory'* is selected in the Memory Options Setup (see chapter 6.1 *Memory Erase/Write/Verify Group* for details), then the following question message box will be displayed:
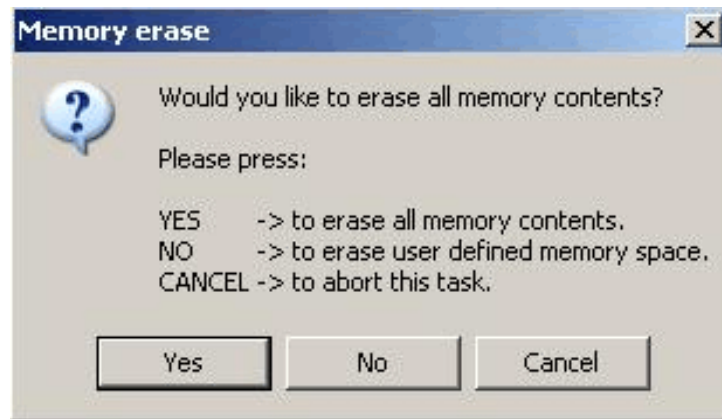
Figure 4.6.3-1

4.6.4 Blank Check button

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

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

4.6.5 Write Flash button

When write flash button is clicked, then contents from the code file without the Security Password contents will be written to the flash memory.

Note 1:   *See chapter 5.1 **Memory Erase/Write Group** for details on how to specify memory segment for writing.*

Note 2:   *The Security Password is removed from the Code contents and the unprotected code is downloaded to flash. If the Security Password should be downloaded to target device, use the **Write CSM Password** at the endo or use **Autoprogram** button to process whole programming action to target device.*

4.6.6 Verify Flash button

The Verify Flash function compares the contents of the flash memory with data from the code file. Verify flash function initiated this way will always use the standard memory verification method, even if the fast verification method is selected from the memory write verification options (see *chapter **5. Memory Option Dialog Screen***). At the beginning of the verification the check sum of the programmed data specified in the code file are calculated. If checksum is OK, then the full verification (read word by word) is established. When the ***Autoprogram*** function is used then only checksum verification is tested if the fast verification is selected.
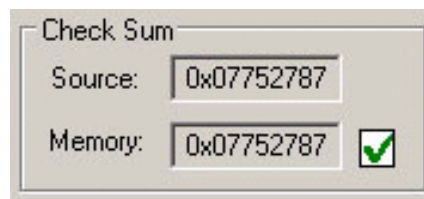
Figure 4.6.6-1

Calculated checksum is displayed in the Check Sum group window - one checksum present the checksum taken from the code file and the second one - taken from the target device. In the Check Sum group is displayed only the simplest - arithmetic checksum, however during the checksum calculation is also verified the Pseudo Signature Analysis (PSA) of the data taken from the code file and target device.

The Arithmetic Check Sum is calculated as the sum of unsigned integer data of all words (16 bits) specified in the code file and taken from the target device. Words that are not specified in the code file are ignored. For example if code file contains only 3 words and even if the target device

has a flash with size 256 kB, then only 3 words from the target device will be taken for check sum calculation. That allows to compare the contents of the used code file with the part of programmed target device. Example below show the check sum calculation of the small code (in TI *.txt format) containing only 6 words.

@3F0000   //address 0x 3F0000
2386 A53F   F253
@3F0100   //address 0x 3F0100
176A E238   38AC

Check sum = 0x2386 + 0xA53F + 0xF253 + 0x176A + 0xE238 + 0x38AC = 0x0002ED66

Because that simple check sum calculation can give the same result with different code, then the PSA is calculated together with the Check Sum above. The PSA is calculated for each continues parts of code or data from flash. When the PSA calculated from code and target device are the same then the check sum above is not modified. If the PSA calculated for the part of code are not the same, then the par of the check sum above taken from the target devices is taken as minus 1 (0xFFFFFFFF) and final checksum verification failed.

Following PSA (16 bits width) procedure is used in the FlashPro2000

```
UINT PSA_calculation( long start_address, int size, UINT16 data[] )
{
 int k;
 UINT16  PSA, PSA_POLY;
      PSA_POLY = 0x0805;
      PSA  = (0xFFFF & start_address) ^ (0xFFFF & (start_address >> 16));
      for( k = 0; k < size; k++ )
      {
         if ( PSA & 0x8000 )
            {
               PSA = PSA ^ PSA_POLY;
               PSA = (PSA << 1) + 1;
            }
            else
            {
              PSA = PSA << 1;
            }
         PSA = PSA ^ data[k];
      }
  return( PSA );
}
```

In the code example above would be calculated two chunks of data - first for data taken from addresses 0x3F0000 and second - taken from addresses 0x3F0100. Size of each chunk - 3 words.

Chunk1
   @3F0000   //address 0x 3F0000
  2386 A53F F253
     Chunk-1 check sum = 0x1BB18

Chunk2
   @3F0100   //address 0x 3F0100
  176A E238   38AC
     Chunk-2 check sum = 0x1324E

If the PSA of chunk-1 from the code file and target devices are the same then the chunk-1 checksum (0x1BB18) is taken to the global check sum calculation, otherwise minus one (0xFFFFFFFF) is taken for the checksum calculation from the target device. The same rule applied to the second data chunk. If two above chunks are calculated without errors (PSA are the same) then the check sum of these two chunks is

$$CS = 0x1BB18 +\ 0x1324E = 0x0002ED66$$

the same as in the example above. If the PSA from the first chunks are not the same, then the CS result taken from the target device is minus one regardless the chunk-1 arithmetic check sum result

$$CS = 0xFFFFFFFF +\ 0x1324E = 0x0001324D$$

When the PSA results are not the same even in one chunk calculated from the code data and taken from the target device, then the final verification result failed, regardless the CS result.

If code is longer and data are placed without gap, then code is divided to chunks with size of each chunk up to 64 words. This means that the PSA are calculated from the chunks size up to 64 words each.

Note-1:   *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.*

Note-2:   *Checksum is calculated for the whole code downloaded to target device and include the Security Password, Serial Number and Model if enabled. From that reason the source nd target device checksum value can be not the same when the next target device is programmed and when the serialization is enabled, even if the code file is not modified. The checksum variation during programming with serialization is normal.*

4.6.7 Read/ Copy Flash button

When *'Read/Copy'* button is clicked, then data can be read from the target microcontroller and displayed in the Flash Memory Data window (see Fig.4.6.7-1). This window can also be selected from *'Flash Memory Data'* from the *'View'* menu. Flash memory data viewer, shown in figure 4.6.7-1, displays the code address on the left side, data in hex format in the central column, the same
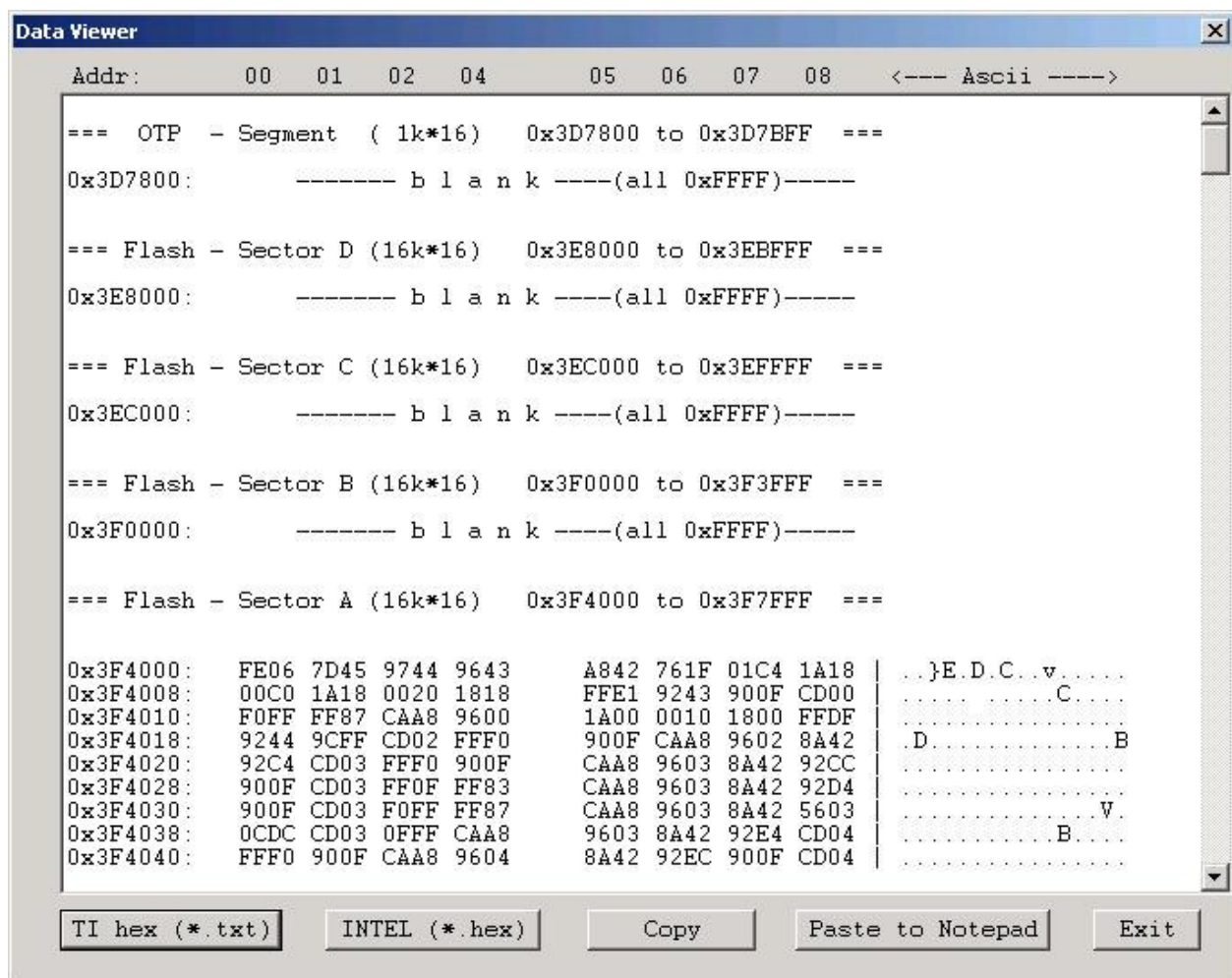
Figure 4.6.7-1

data in Ascii format in the right column. The contents of the code viewer can be converted to Texas Instruments *.txt or Intel *.hex file format (16 bits width) by clicking on the *'TI hex (*.txt)'* or *'Intel (*.hex)'* 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 would be saved in the specified by user file name and opened as a current Code File. Also programmer setup would be modified for the copy procedure. Especially the serialization will be disabled and the *'Flash Memory only'* option will be selected in the *'Write/Erase/Verify Address Range'*. Following message shown on figure 4.6.7-2 is displayed.

Figure 4.6.7-2

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

4.7 Next button

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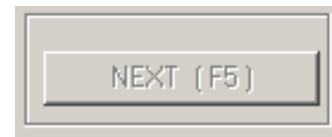
Figure 4.7-1
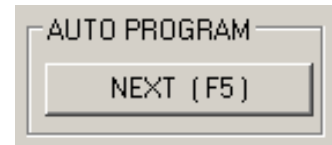
Figure 4.7-2
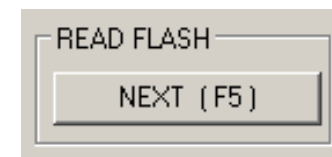
Figure 4.7-3

4.8 Script button

The *'Script '* button is the dynamically programmable device action button that allows to take a desired action taken from the script file. See chapter 9.4 for details of the script file commands. The **Script** button has a name **Script File - none** (Figure 4.8-1) if the script file is not defined or **Script:** with used file name when the script file is active (Figure 4.8-2). When the **Script** button is pressed and the current script file is not active, then the **Open File** dialog is displayed and the desired script file should be selected. When the **Script file** button is not empty and the new script file if required, then the new file can be selected from the pull down menu - **File-> Open Script File**.

Figure 4.8-1

Figure 4.8-2

The **Script** button is very useful if target devices are connected in the JTAG chain and more then one unit should be programmed. Using one button it is possible to program a few devices connected in the JTAG chain with different code downloaded to each device. Below is an easy script file used to download two independent codes to two target devices connected in the JTAG chain. Using the notepad editor create the script file and save it eg. as the file **"script-chain.sf"** or any other file name.

```
;---------------------------------------------------------------------
; easy script file for two devices connected in the JTAG chain
;---------------------------------------------------------------------

 LOADCFGFILE C:\Elprotronic\Project\Cpp-DSP-C2000\FlashPro2000\target1of2.cfg
 LOADCODEFILE C:\Elprotronic\Project\Cpp-DSP-C2000\FlashPro2000\data2.txt
 AUTOPROGRAM

 LOADCFGFILE C:\Elprotronic\Project\Cpp-DSP-C2000\FlashPro2000\target2of2.cfg
 LOADCODEFILE C:\Elprotronic\Project\Cpp-DSP-C2000\FlashPro2000\data4.txt
 AUTOPROGRAM

 END
;---------------------------------------------------------------------
```

When the script file above is used then the first configuration file and the fist code file are downloaded first and the Autoprogram function is executed. The first target devices is programmed. The second target devices is not used in this process. When finished, then the next configuration file and code file are downloaded and again the Autoprogram is executed programming the second target device. When finished then the action used the script file is finished.

Before running the script file the configuration files required in the project should be created using the GUI software first - the configuration files named *target1of2.cfg* and *target2of2.cfg* used in the script file above. To do that connect target devices to programming adapter, select desired configuration for the first devices, select JTAG 1 and JTAG size (2 in the case above). Save the configuration file as *target1of2.cfg* and create configuration for the second device and save it as *target2of2.cfg*. When it is done then press the *Script* button and select the *script-chain.sf* file. Two targets devices will be programmed when the *Script File* button 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 0000 to FFFF when contents of data exist in the code file, otherwise it is displayed as a dots '....'(if data does not exist in the code file). When code size exceeds Flash

```
Data Viewer                                                                  ×

  Addr:        00   01   02   04      05   06   07   08    <--- Ascii ---->

=== OTP  - Segment ( 1k*16)   0x3D7800 to 0x3D7BFF   ===

0x3D7800:            --------------- g a p ----------------

=== Flash - Sector D (16k*16)   0x3E8000 to 0x3EBFFF   ===

0x3E8000:            --------------- g a p ----------------

=== Flash - Sector C (16k*16)   0x3EC000 to 0x3EFFFF   ===

0x3EC000:            --------------- g a p ----------------

=== Flash - Sector B (16k*16)   0x3F0000 to 0x3F3FFF   ===

0x3F0000:            --------------- g a p ----------------

=== Flash - Sector A (16k*16)   0x3F4000 to 0x3F7FFF   ===

0x3F4000:   FE06 7D45 9744 9643    A842 761F 01C4 1A18 | ..}E.D.C.Bv.....
0x3F4008:   00C0 1A18 0020 1818    FFE1 9243 900F CD00 | ..... .....C....
0x3F4010:   F0FF FF87 CAA8 9600    1A00 0010 1800 FFDF | ................
0x3F4018:   9244 9CFF CD02 FFF0    900F CAA8 9602 8A42 | .D............B
0x3F4020:   92C4 CD03 FFF0 900F    CAA8 9603 8A42 92CC | ................
0x3F4028:   900F CD03 FF0F FF83    CAA8 9603 8A42 92D4 | ................
0x3F4030:   900F CD03 F0FF FF87    CAA8 9603 8A42 5603 | ..............V.
0x3F4038:   0CDC CD03 0FFF CAA8    9603 8A42 92E4 CD04 | ...........B....
0x3F4040:   FFF0 900F CAA8 9604    8A42 92EC 900F CD04 | .........B......
0x3F4048:   FF0F FF83 CAA8 9604    8A42 92F4 900F CD04 | .........B......

  TI hex (*.txt)      INTEL (*.hex)      Copy     Paste to Notepad     Exit
```

Figure 5-1

memory space of the selected microcontroller, then warning message

```
':== Data out of the Flash Memory Space of the selected TMS320F2000 =='
```

is displayed first.

The contents of the code viewer can be converted to Texas Instruments *.txt or Intel *.hex file format by clicking on the *'TI hex (*.txt)'* or *'Intel (*.hex)'* 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 or two code files can be compared
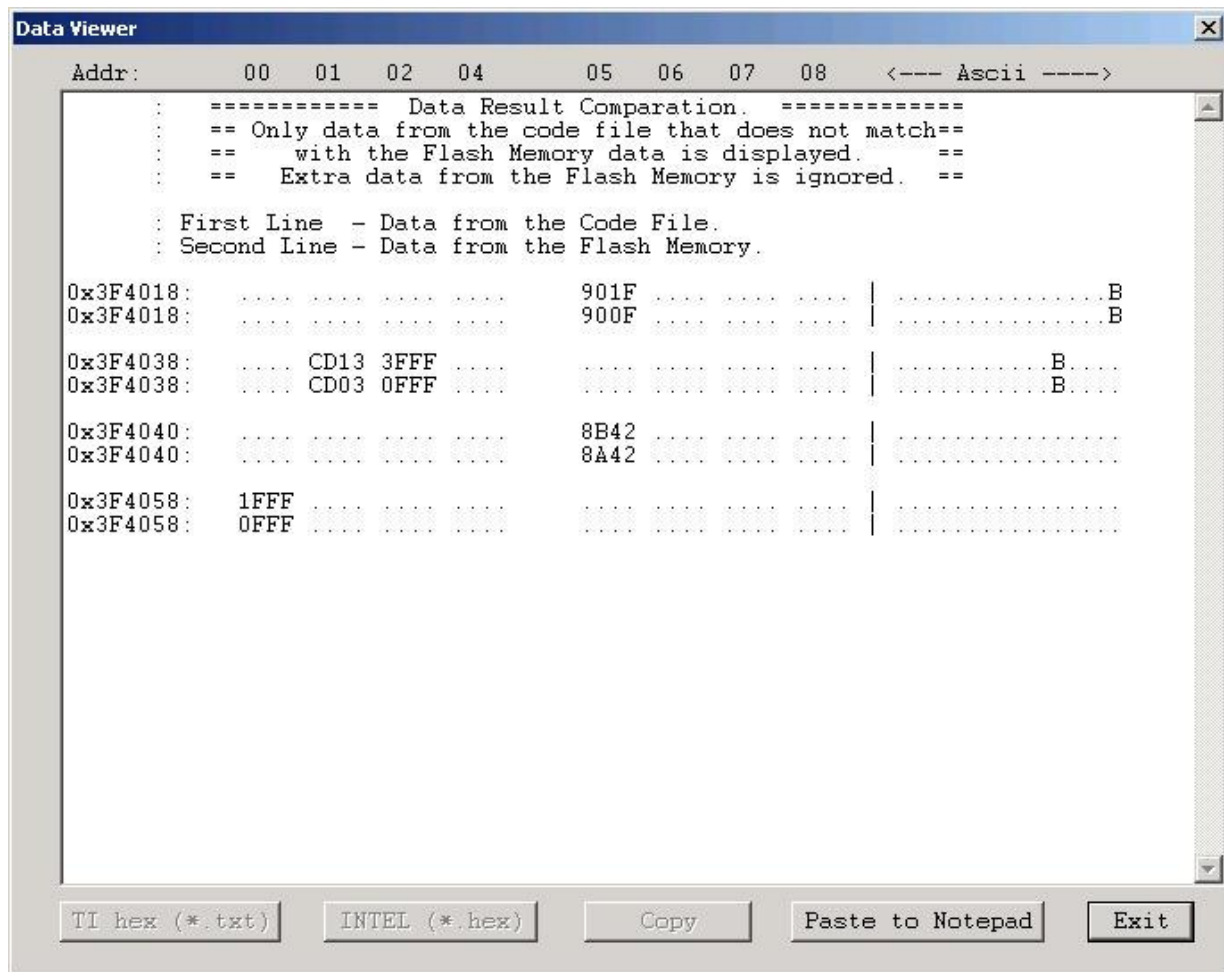
Figure 5-2

and differences displayed in a the viewer by selecting *'Compare Code & Flash Data'* or *'Compare two Code Files'* from the *'View'* menu. Only data that are not the same will be displayed. In the first line code file data will be displayed, and in the second line - Flash memory or data from the second file (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 Dialog Screen

The Memory Options Dialog Screen (Fig.6-1) has four 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 Retaind Data in Flash allows to specify range of flash memory that should be keep unmodified even if the flash sector would be erased. Contents of the specified Retain Data in Flash would be resorted after erasing and blank checking the flash. The fourth 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.
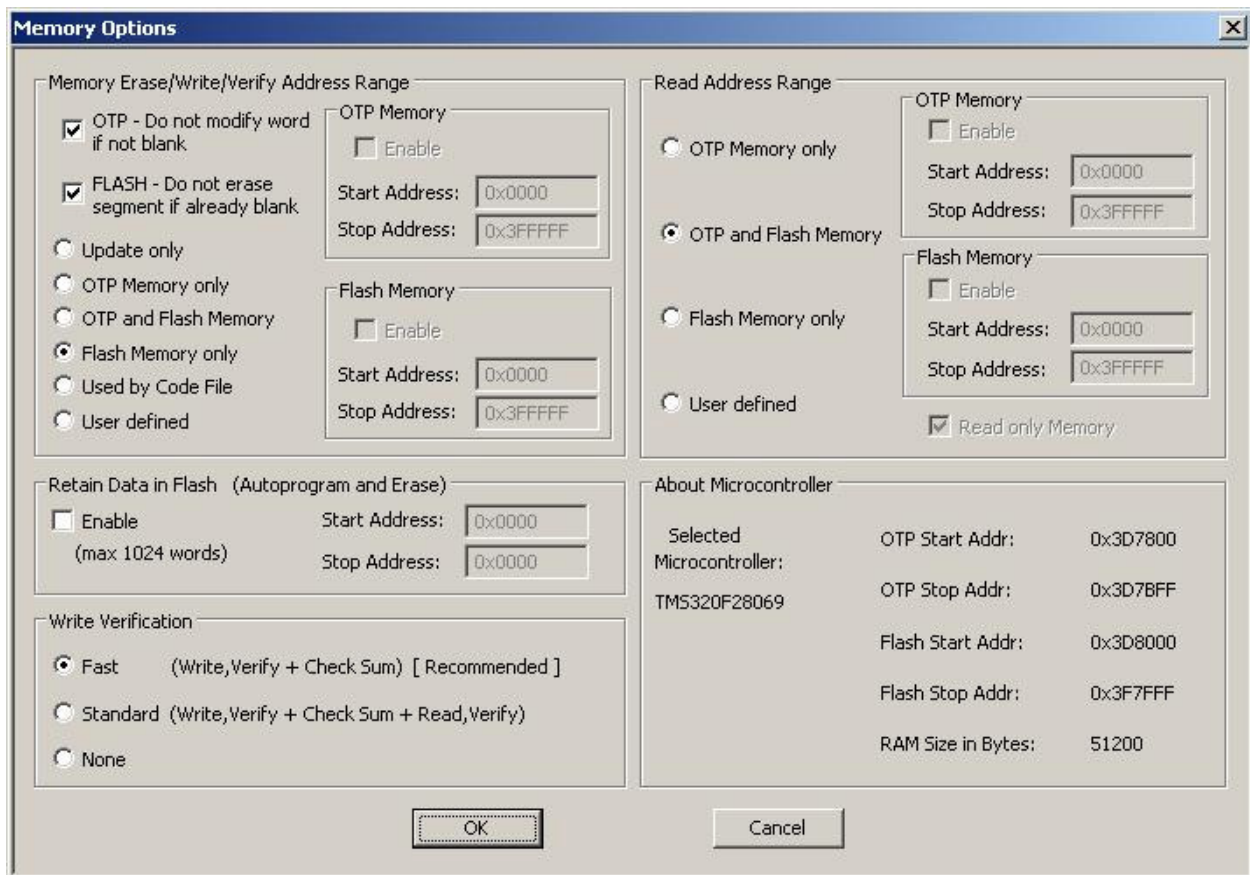
Figure 6-1

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 following available options:

1 **OTP- Do not modify word if not blank:**
When the OTP memory is programmed, then first the contents of the OTP memory are verified with the data prepared for write.

* If the OPT memory is blank in the locations where data are prepared for write then the writing process is continued.

* If the OTP memory contains exactly the same data as the data prepared for write, then the writing process is continued, however the programmed data in OTP are not programmed again.

* When the data in OTP are already programmed in the locations where the data prepared for write and data are not the same then the two options are available:

* if the **OTP- Do not modify word if not blank** is selected then the writing process is terminated. The whole OTP is not modified. Following messages are displayed in the pop-up message box (Figure 6.2) and in the report window (Figure 6.3).

Figure 6.2
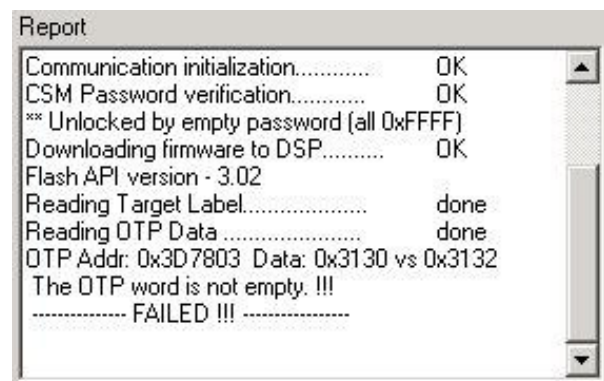
Figure 6.3

* if the **OTP- Do not modify word if not blank** is not selected then all bits in OTP are verified with bits prepared for write. If any bit is already programmed to '0' and

in the prepared data this bit is '1' then the writing process is terminated. The whole OTP is not modified. Following messages are displayed in the pop-up message box (Figure 6.4) and in the report window (Figure 6.5).

Figure 6.4

Figure 6.5

2 ***FLASH - Do not erase segment if already blank:***

All selected segments that should be erased or programmed are checked first if they are blank. If sector is already blank then it can be unmodified, or erased again regardless the flash contents. If the option ***FLASH - Do not erase segment if already blank*** is selected, then the blank sector is not erased again if it is already blank. That can speed-up programming process, because erase procedure requires a more then one second to erase one sector of the flash memory.

3. ***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,

```
@3F0008
25CA  8040  39E3  F802
@3F0020
4835  5972  ACB8
q
```

Figure 6.6

which will be downloaded to flash memory. For example, if code file contains only data as shown in figure 6.6 (in Texas Instruments format) then 4 words of data will be written starting at location 0x3F0008 and 3 words of data starting at location 0x3F0020. Before

writing operation, all data in the flash memory at the specified location should be blank (contain value 0xFFFF). The software will verify automatically if this part of memory is blank and will only proceed to program the device if verification is successful.

4.   **OTP Memory only:**
When this option is selected then the OTP memory would be programmed only. All other data specified out of OTP memory location will be ignored. When the OTP is programmed then the contents of the OTP are examined first if blank in the locations where the data are prepared for write. See the **OTP- Do not modify word if not blank:** above for details.

5.   **OTP and Flash Memory**
When this option is selected then whole memory can be programmed - OTP and Flash memory. Flash memory is erased first. When the OTP is programmed then the contents of the OTP are examined first if blank in the locations where the data are prepared for write. See the **OTP- Do not modify word if not blank:** above for details.

6.   **Flash Memory only**
This is the most commonly used option and is selected as the default. This option allows to erase and program whole Flash Memory only. The OTP memory is not programmed regardless the code contents.

7.   **Used by code file:**
This option allows the OTP and Flash Memory segments, used by data specified in code file, to be programmed. 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 be erased and new data written.

8.   **User Defined:**
This option is functionally similar to options described before, but addresses range of the erased/write/verify flash memory and space of the OTP 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 addresses edit lines will be enabled. The check boxes allow the user to select the space of the OTP and Flash memory to be used (erased, write, verified). If the space of the sector specified in Start/End addresses cover only

part of the Flash sector, then the whole sector will be areas, however only the specified part of the sector will be programmed

6.2   Retain Data in Flash

User defined option in the **Retain Data in Flash** group allows to specify other region to be restored after erasing the flash. Location of the retain data block is not limited and can be used at any part of flash memory. Maximum size of the retain data block is limited to 1024 words only.

6.3   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.   **OTP Memory only**
2.   **OTP and Flash memory**
3.   **Flash memory only**
4.   **User Defined**

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

6.4   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 is performed after memory write process is completed when the **Autoprogram** option is used. When the **Fast verification** is selected then the checksum and PSA of the code and target device contents are verified. See the **4.6.6 Verify Flash** chapter for details how the checksum and PSA are calculated

*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.

7. Adapter Options

7.1   Interface Selector Box

The communication interface type - **SCI-Boot, JTAG-fast** and **JTAG-slow** can be selected in the Interface group in the main dialog screen. Proper communication interface should be selected, otherwise communication with the target device can fail. When the JTAG communication is used then the **JTAG-fast** interface should be selected. If your target device has installed suppressors, capacitors or EMI filters in the JTAG lines or used JTAG cable is long that can degradate the JTAG communication speed, then the **JTAG-slow** interfaces should be selected.
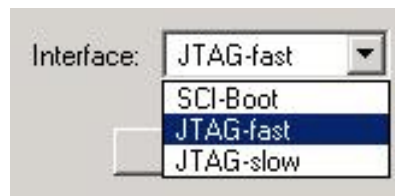
Figure 7.1

When the SCI-Boot interface is used, then the initial (after DSP reset) UART communication between target device and FPA is established with baud rate related to the selected CLK frequency. The initial communication baud rate is in the range 5 kb/s to 20 kb/s. That initial communication is used to download the fast boot loader to DSP RAM. When the fast boot is downloaded then the new communication protocol is initialized with communication speed up to 500 kb/s. Make sure that hardware in the SCI-TX/RX lines have no components that can degradate the communication speed.

7.2   JTAG Chain Selector Box

The JTAG communication standard allows to connect more then one devices in the chain. In the chain then the TDO from the first device is connected to the TDI of the second device, TDO from the second device connected to TDI of the next device etc. All other lines like TMS, TCK, RST are connected in parallel. The TDI of the first device is connected to TDI output of the JTAG interface and the TDO of the last device is connected to TDO input of the JTAG interface. The FlashPro2000 can make a communication with target device connected in the JTAG chain, but the JTAG configuration must be specified first. Software should know how many devices are connected

in the chain, what is the size of the IR registers of each devices connected in the chain and which device in the chain should be selected. The IR size in the TMS320F28xx is 38 bits. But other devices connected in the JTAG chain can have not the same IR size. The JTAG IR size setup dialog screen (available under pull-down menu *Setup-> JTAG Chain)* allows to select desired target device with know the IR size, or enter the IR size for unknown devices (Figure 7.2). Up to six devices can be specified in the JTAG IR size dialog screen. When the device is not specified in the pull-down menu then option *Other* should be selected and desired IR size entered in the IR size field.

Figure 7.2

When the IR size is unknown then it is possible to detect the total size of the IR size using the FlashPro2000 software. Connect device or devices in the JTAG chain, connect JTAG adapter to target device and press any action button, like *READ* button. At the communication startup the total IR size and number of devices connected in the chain are displayed in the report window regardless of the success or failed the following JTAG communication (Figure 7.3). In the example below are connected two MCU TMS320F28xx in the chain. The total IR size is 2*38 = 76 bits and 2 detected devices in the chain. In the software it was selected only one device in the chain and because number

of devices selected and detected are not the same - the procedure was terminated. If only one unknown device is connected to FlashPro2000 then the IR size can be detected and the meassured IR size can be entered in the JTAG IR size dialog box.
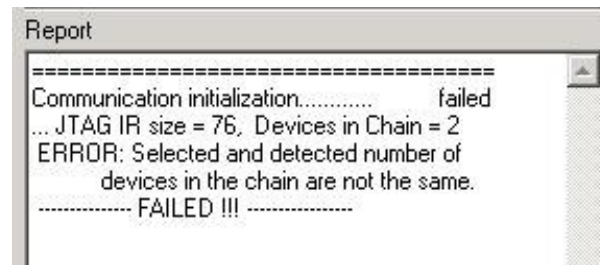
Figure 7.3

When the IR size setup is done then in the Main dialog screen it should be selected number of devices in the JTAG chain (1 to six) and position of the desired target device in the chain in the *JTAG Chain* group (Figure 7.4). In the example below are connected two devices in the chain and it is selected the first target device in the chain.

Figure 7.4

7.3   Reset Dialog Box

The Target's Reset Dialog (Figure .5) screen enables the user to select the Reset pulse duration and reset line state at the end of programming process. Access to the hardware RESET line is only available when the SCI-BOOT communication is used. When the JTAG communication is used then target device can be reset via JTAG commands and hardware reset line is not reachable.

7.3.1 Reset pulse duration

The reset pulse allows the adapter to initiate communication with a microcontroller using the SCI-BOOT Interface. In most cases the pulse width of 10ms is sufficient to initiate communication process. However, this may be affected by additional load on the reset line. Therefore, four additional settings, 100, 200, 500 ms and custom , are available. When the RESET IC circuit is used then the custom defined reset pulse duration can be used. Two parameters of the custom defined

reset pulse are defined - initialization reset pulse time (typically very short - 1 ms) and an idle reset time. Idle reset time must be set at least to duration of the reset time generated by the RESET circuit.

7.3.2 Final Target Device action

Every device action, like AUTO Program, Read etc. starts with the activation of the RESET line (active low). When the device programming action begins the RESET line is raised high. When

Figure 7.5

device action is finished, then RESET line is again asserted, protecting the target device from running the application program.

The target device can be set to run an application immediately after the target device programmed. This allows to verify functionality of the programmed device if required. In order to do this check the

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

or

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

option in the Reset Options window, shown in Figure 7-5. Application run time can be unlimited or limited up to 120 seconds. Limited time is specified in the *"Application Program RUN time"* box. When entered '0' in the *"Program RUN time"* box then time is unlimited.

7.3   Preferences Dialog Box

In the *Preferences* dialog screen (Figure 7.3) is possible to select the PDF reader, select the hex conversion tool and select desired audio option when warning or error are present.

In the first line it can be selected the PDF reader. By default it is entered Acrobat Reader - *AcroRd32.exe*. If other PDF reader is used in the system, then the full path of the used PDF reader should be entered in the edit line. Name and path can also be entered using the *Browse* button.

The FlashPro2000 software allows to convert the *.out file generated by TI's Code Composer Studio (CCS). However the TI's *hex2000.exe* conversion file is used during the conversion process. It is required to enter the path to the *hex2000.exe* file, otherwise the conversion of the *.out will not work. All parameters required during conversion the *.out file to 16-bits width Intel file are generated automatically by the FlashPro2000 software.

By default the *hex2000.exe* is located in the following CCS directory

C:\CCStudio_v3.3\C2000\cgtools\bin

The full path and file name selected in this case is

C:\CCStudio_v3.3\C2000\cgtools\bin\hex2000.exe

If from any reason the file is located in other place then the path and name of the hex2000.exe should be entered in the TI HEX conversion utility line. If the CCS is not installed in the PC, then it is recommended to copy and paste the hex2000.hex file from other PC where the CCS is installed and copy this file eg. to FlashPro2000 directory

C:\Program Files\Elprotronic\C2000\USB FlashProC2000

Path and name of the hex2000.exe should be provided in the TI HEX conversion utility line.

In the Option group box it is possible to enable the Report history in the report window (see figure 4.1). When enabled then the report history is displayed up to 8 kB characters (approximately 20 last communication messages). When disabled, then the only last programming report is displayed. Programming software can generate audio tones when error programming occurred or tone OK at the end of programming. Tone can be generated using PC speaker or audio wave generator. Option dialog box allows to select desired audio option (see Figure 7.6).
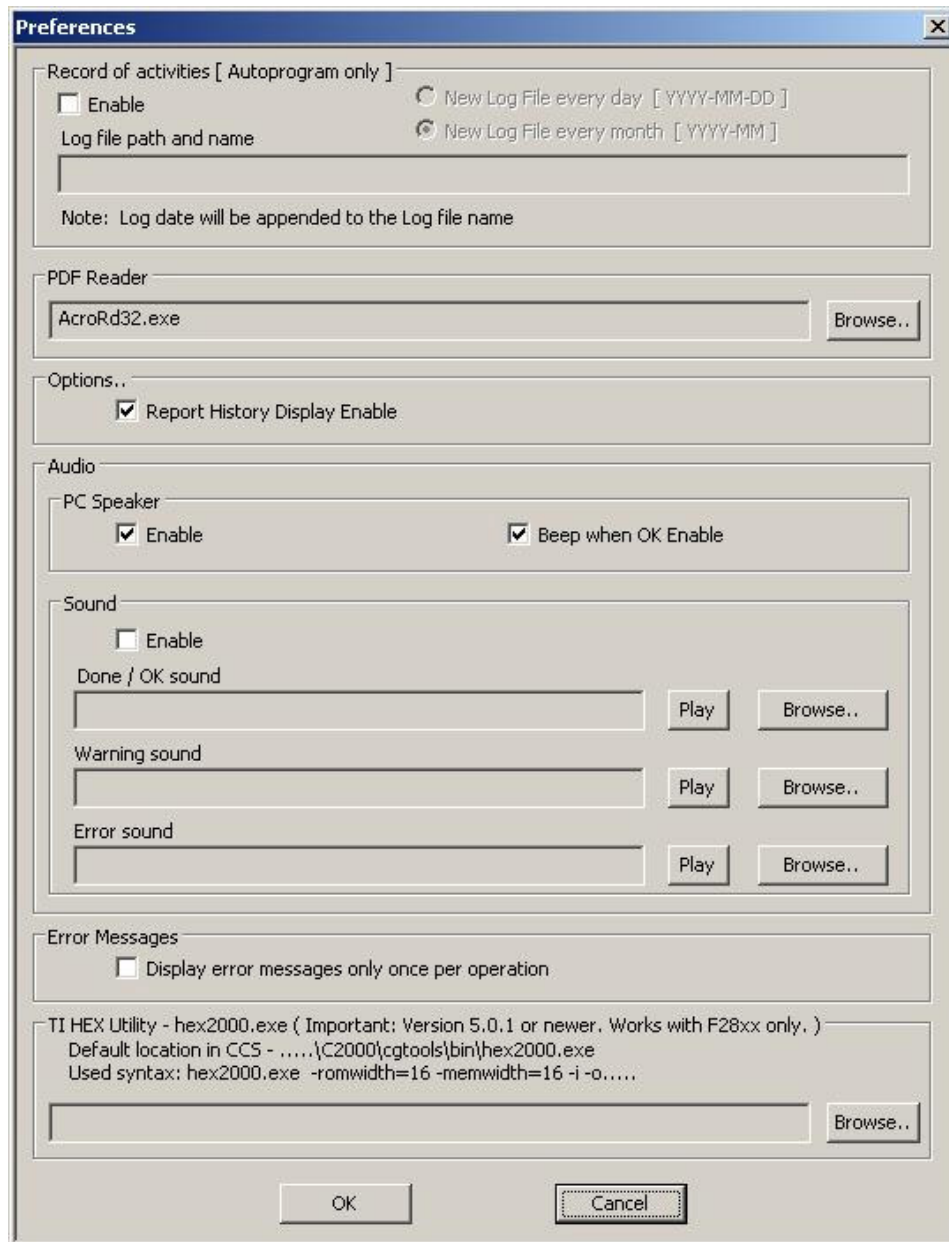
Figure 7.6

8. Serialization

8.1   Introduction

    FlashPro2000 programming software has ability to automatically create the target device's serial number and save it in the flash memory. The serial number (SN) that have already been used are stored in the data file. The new ***Serial Number*** can be created automatically by incrementing the ***Serial Number*** or can be taken from the file created by user. Used serial numbers are stored in a data file. Furthermore, model name, group, revision can be downloaded to target device.

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

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

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

Figure 8.1-1

8.2   Serialization Dialog Screen

Figure 8-2

Serialization dialog box, shown in figure 8-2, allows configuration for serialization process to be set. Serialization can be enabled, or disabled, by selecting the check mark in the ENABLE

Serialization box. When serialization is disabled all edit lines and check boxes are disabled. When serialization is enabled all fields must be set.

8.2.1 Serial number File

The 'Serial Number File Path and Name' specifies the full path and file name, where data base contents will be saved. Serial Number file contains following data, separated by tabulation:
1.   Serial Number Format (F0,F1,F2,F3,F4,F5,F6),
2.   Serial Number,
3.   SN action type (New SN, unmodified SN, overwritten SN, manual SN)
4.   Time and date, when SN has been created,
5.   Code File Name
6.   Model text.

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

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

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

8.2.2 Serial number formats

Programming software has seven methods for creating the serial number, referred to as *Display format*, and four methods of storing the SN in the memory, referred to as *In Memory Format* in the serialization dialog screen. When a serial number is created, current date (if required) is taken from the PC timer. Make a sure, that your computer has correct date and time.
Display Format:
1. YYYY-1234(5)   -(SN Format - **F0**) Serial number has 8 or 9 characters. First four characters contain current year, and remaining 4 or 5 characters contain the serial number, eg. SN 20030123 or 200300123 has a number 0123 (or 00123) created in the 2003 year.

---

2. YYMM-1234(5)   - (SN Format - **F1**) Serial number has 8 or 9 characters. First two characters contain last two digits of current year, next two characters contains current month, and remaining 4 or 5 characters contain a number, eg. SN 03030123.

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

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

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

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

7. Custom   - (SN Format - **F6**) 4 to 16 Ascii characters or hexadecimal numbers entered manually or from the Bar-Code Reader.

8. From the file   - (SN Format - **F7**) 4 to 16 Ascii characters or hexadecimal numbers taken from the user created file.

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

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

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

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

Display Format: YYYY-1234(5)   - size in FLASH - 4 bytes
SN 200300123 will be saved as
     YYYY - 2003   (Decy)   -> 0x07D3   (hex)

12345 - 00123   -> 0x007B   (hex)

    In flash memory this number can be seen as
            07D3   007B   -> *HEX(MSW first)*
            007B   07D3   -> *HEX(LSW first)*

    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)*

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)*

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)***

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)***

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)***

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

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

     or

       C169 A502   -> ***HEX (LSW/LSB first)***

8.2.2.2   BCD format

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

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

SN 20030123 will be saved as

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

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

    <--- Hex format bytes--->
    2003 0123   (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--->
        0303 0123   (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--->
        0405 1101 2300   (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
    1234 5678   ->   0x12 0x34 0x56 0x78 (bytes)

In flash memory this number can be seen as

        <--- Hex format bytes--->
        1234 5678   (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--->
        1234   (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 –>
3230 3033 301 3233   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 –>
3033 3033 301 3233   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 -->
          3034   3035  3131  3031   3233               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 -->
          3034   3231  3430  3132  3300              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 -->
          3132   3334  3536  3738              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 -->
          3132   3334                            1234          (Size - 4 bytes)
```

Display Format: *Custom* or *from the file*   - size in FLASH - defined size in bytes
     Taken from the file or entered manually Ascii string will be saved in the flash memory.

When the *Ascii* format is selected, then the Ascii string is saved in memory **"as is".**
All Ascii characters can be used**.** For example the entered following string

　　　02WX24S234

will be saved in memory as

　　　3032　5758　3234　5332　3334　　->　　"02WX24S234"

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

　　　　02A3B109E12F

will be saved in memory as

　　　　HEX(MSB first) -> 02A3　B109　E12F
or　　　*HEX(LSB first)*　-> 2FE1　09B1　A302

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 dialog 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:

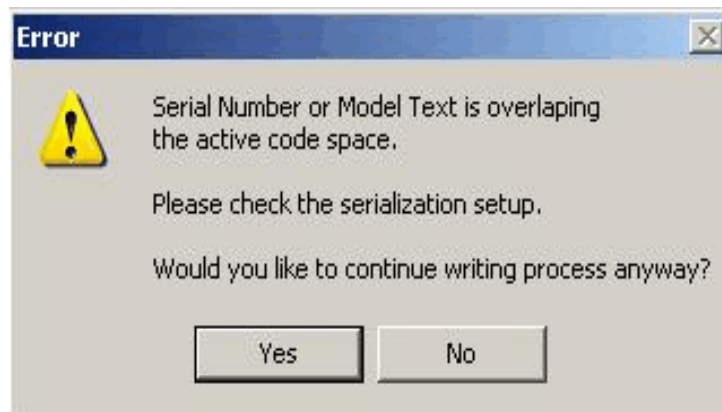

Figure 8.2.1-1

8.2.3 Model, Group, Revision

Custom text or data (hex), saved in target device's flash memory is a string or data, up to 32 characters (bytes) long, in Ascii or hex format. It can contain any text or data, but this feature is

intentionally created to allow the hardware model, revision and group to be saved. Typically the object code does not contains this kind of information, but it may be useful in some applications.

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

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

8.2.4 Device Serialization box

Device Serialization box, located on the main programming dialog screen (see figures 8.2.4-1 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 dialog 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.

Figure 8.2.4-1

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 Dialog Screen

Serialization Report Dialog 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 FlashPro2000 software allows to download the serial number from custom defined data file. When the data file is used then in the serialization dialog 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 dialog 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 dialog screen is used.

#SN_HEX_MSB   ;optional
Select the HEX MSB first format regardless setup in the serialization dialog screen.
#SN_HEX_LSB   ;optional
Select the HEX LSB first format regardless setup in the serialization dialog screen.

When the format is specified as the hex format (**#SN_HEX_MSB** or **#SN_HEX_LSB)** then string can contains gaps between numbers or 0x on the front. All gaps will be removed. Up to 256 characters (gaps excluded) can be specified for one number. All characters must be located in one line for the same SN.
Example:
The hex number **A434BC5696AD10ACF0**
can be specified as
   A434BC5696AD10AC
 or
   0xA434BC5696AD10AC
 or
   A434 BC56 96AD 10AC
 or
  A4 34 BC 56 96 AD 10 AC F0

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

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

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

can the SN be specified as follows

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

and list of following serial numbers

```
0001
0002
0003
```

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

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

that defined following serial numbers

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

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

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

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

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

The same Serial Number list with specified prefix /suffix

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

001
002
003
004
005
; ======================================================
```

When the SN data file is prepared, then at the first the data base file should be opened(see Figure 8.2). When the desired *Serial Number Format* is selected, then using the *SN/IEEE file* button located in the main dialog screen (Figure 4.1) the desired SN file should be opened. Selected file is converted to final format and all listed serial numbers are verified with the data base file if there was note used before. If the specified SN have been used before, then these numbers are

removed from the SN list. When the SN file is read and verified, then the pending SN list is displayed in the screen (Figure 8.4-1) with following information displayed on the top of the list

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

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

      * number of the accepted SN

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


```
Serial numbers list from the file                        ×

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




         Paste to Notepad              Exit
```
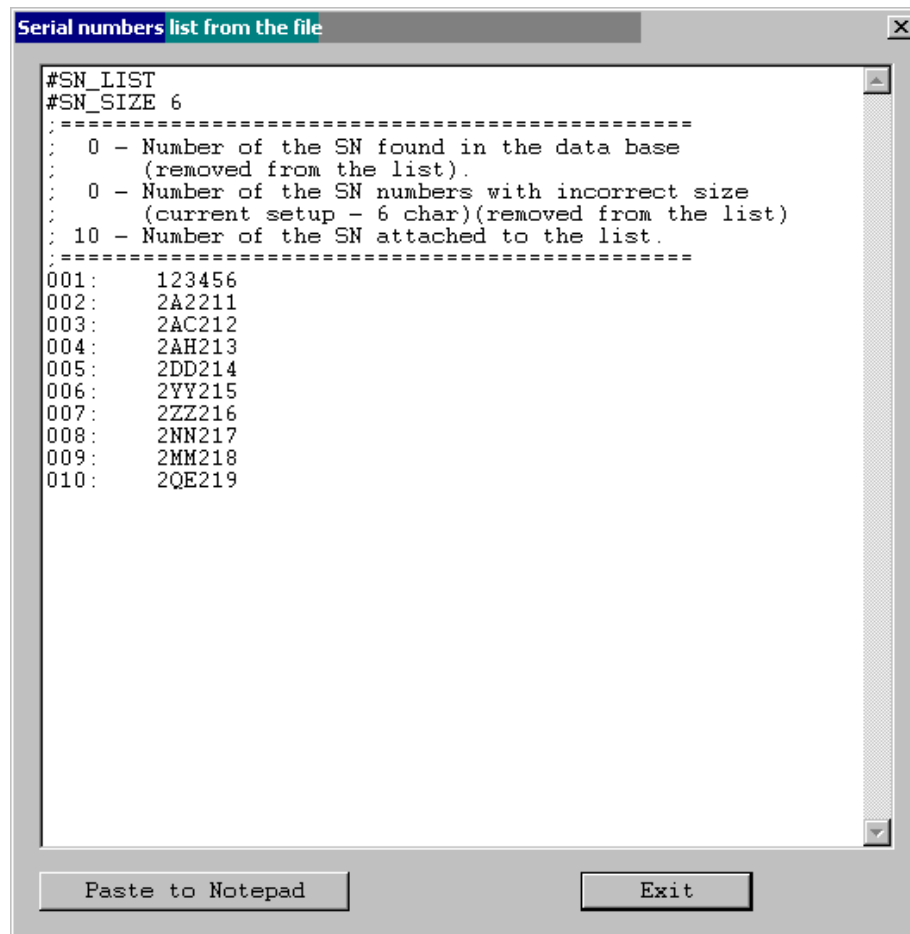
Figure 8.4-1

9. Check Sum Options

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

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

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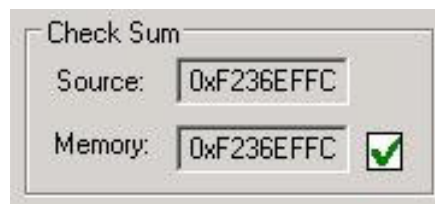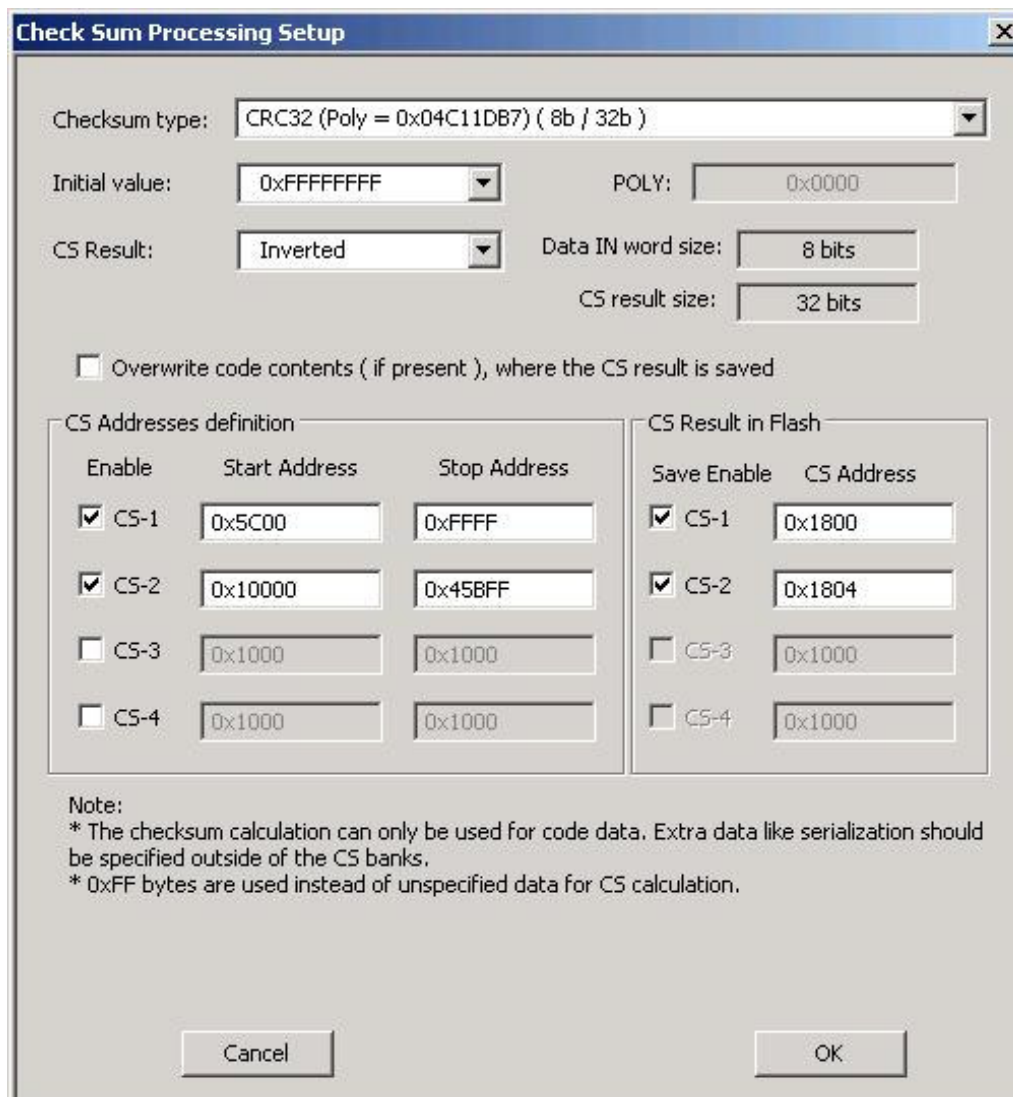

Figure 9.1

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.

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.
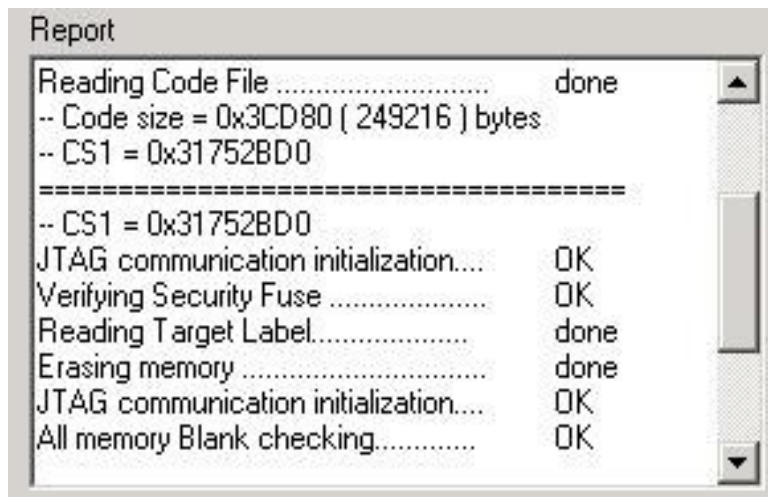
Figure 9.3

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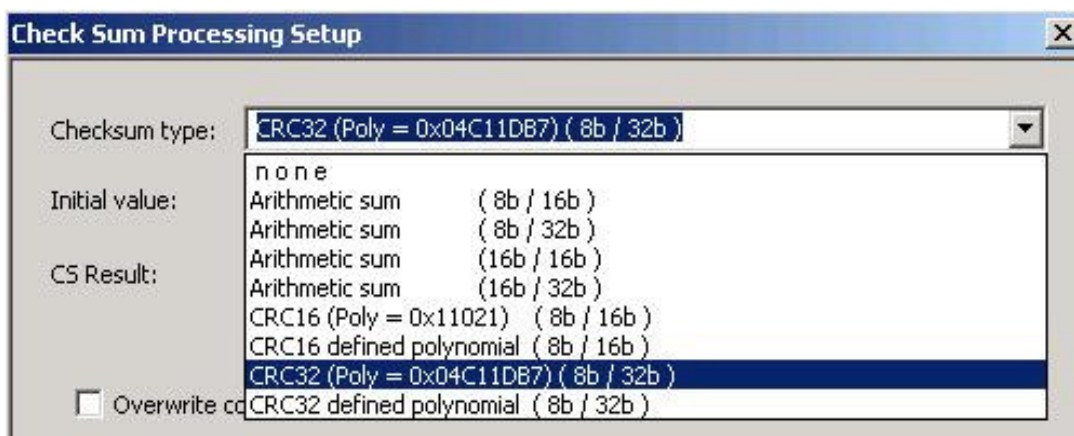
Figure 9.4

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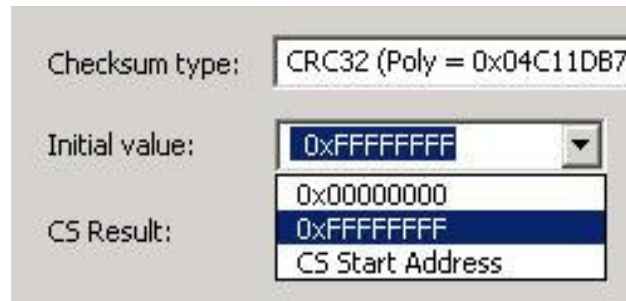
Figure 9.5

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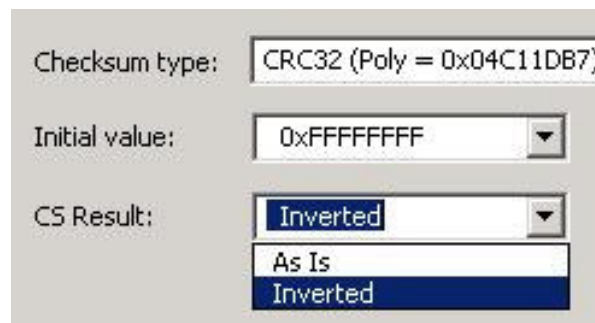
Figure 9.6

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) ( 0xFF & data[addr]);
   CS = CS + (unsigned int) ( 0xFF & (data[addr]>>8));
```

```
}
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 int) ( 0xFF & data[addr]);
   CS = CS + (unsigned int) ( 0xFF & (data[addr]>>8));
}
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++ )
{
 CS = CS + (unsigned int)data[addr];
}
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++ )
{
 CS = CS+(unsigned long)data[addr];
}
```

```
        CS = 0xFFFFFFFF & CS;
        if( cs_inverted )
            CS = 0xFFFFFFFF ^ CS;
```

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

and

CRC16 defined polynomial   - (8b / 16b)

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

```
        CS = CS_initial_value;
        for ( addr = StartAddress;  addr <= EndAddress; addr++ )
        {
          CS = CS_CRC16_8to16( (long)(0xFF & data[addr]), CS );
          CS = CS_CRC16_8to16( (long)(0xFF & (data[addr]>>8)), 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)(0xFF & data[addr]), CS );
          CS = CS_CRC32_8to32( (long)(0xFF & (data[addr]>>8)), 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. Script File - defined programming sequence

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

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

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

10.1 Script button

The **'Script'** button is the dynamically programmable device action button that allows to take a desired action taken from the script file. The **Script** button has a name **Script File - none** (Figure 10-1) if the script file is not defined or **Script:** with used file name when the script file is active (Figure 10-2). When the **Script** button is pressed and the current script file is not active, then the **Open File** dialog is displayed and the desired script file should be selected. When the **Script file** button is not empty and the new script file if required, then the new file can be selected from the pull down menu - **File-> Open Script File**.
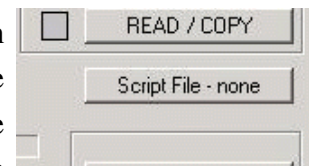
Figure 10-1
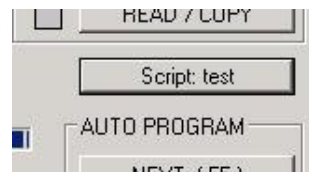
Figure 10-2

The **Script** button is very useful for implementing a short programming sequence not present directly in the **Device Action** group buttons Below is an easy script file used for downloading two independent codes to target device - first code used for hardware test if possible, and when hardware is ok, then the second code is downloaded as the final code to target device. The same sequence can be used with other buttons, but sequence should be always repeated, that of course is not convenient.

Using the notepad editor create the script file and save it eg. as the file *"test.sf"* or any other file name. See this chapter below for all available instructions that can be used in the script file.

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

 LOADCFGFILE C:\Program Files\Elprotronic\C2000\USB FlashPro2000\test.cfg
 LOADCODEFILE C:\Program Files\Elprotronic\C2000\USB FlashPro2000\test.cfg
 AUTOPROGRAM

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

  IF BUTTONNO GOTO finish

 LOADCFGFILE C:\Program Files\Elprotronic\C2000\USB FlashPro2000\final.cfg
 LOADCODEFILE C:\Program Files\Elprotronic\C2000\USB FlashPro2000\final.cfg
 AUTOPROGRAM

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

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

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

10.2 Script file option

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

When the executable file FlashPro2000.exe is called with a script path as an argument e.g.

C:\Program Files\Elprotronic\C2000\USB FlashPro2000\FlashPro2000.exe

FlashPro2000.exe   -rf   C:\Program Files\Elprotronic\C2000\USB FlashPro2000\script.txt

or when the icon with the FlashPro2000.exe and script file path is executed then programmer starts automatically programming sequences according to procedure specified in the script file. Access to other buttons are blocked. When script file sequence is finished then program is terminated. There is not option to modify the running sequence when script sequence is used. This option is useful in production, because nobody can modify sequence that has been prepared for the production purpose.

10.3 Script commands

LIMITATIONS:

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

SYNTAX:

white spaces before instructions, labels etc are ignored.

; comment   - all contents after semicolon are ignored.

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

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

      NOTE: After a label you cannot specify any commands in the same line. Line can only contain a label.

LIST OF INSTRUCTIONS:

MESSAGE   - message declaration ,that can be saved to file if required
   " message - line -1 "   - Text.
   " message - line -2 "   - Each line contents must be located between characters " "
   " max up to 50 lines "   - Number of content lines - up to 50 lines.

SAVEMSG   filename   - save message created in MESSAGE to specified file
APPENDMSG   filename   - append message created in MESSAGE to specified file
SAVEREPORT   filename   - save message from the GUI report window to specified file
APPENDREPORT   filename   - append message from the GUI report window to specified file

GUIMSGBOX   ENABLE   - display the message boxes (warning, errors etc) created by GUI enabled
GUIMSGBOX   DISABLE   - display the message boxes (warning, errors etc) created by GUI disabled
IFGUIMSGBOXPRESS   OK   - apply button OK / YES when the message box created by GUI is generated, but disabled to be displayed.

IFGUIMSGBOXPRESS   CANCEL   - apply button CANCEL / NO when the message box created by GUI is
                                        generated, but disabled to be displayed.

MESSAGEBOX type **FCTEXT** - pop-up message box with buttons.
                                - message taken from the FCONTROL function (User's DLL)
MESSAGEBOX type   - pop-up message box with buttons.
   " message - line -1 "   - Text displayed in message box.
   " message - line -2 "   - Each line contents must be located between characters " "
   " max up to 50 lines "   - Number of content lines - up to 50 lines.

  Message box type list
    OK   - One button   OK
    OKCANCEL   - Two buttons   OK , CANCEL
    YESNO   - Two buttons   YES , NO
    YESNOCANCEL   - Three buttons   YES , NO, CANCEL

GOTO label
CALL label   - CALL procedure.
RETURN   - return from CALL.

IF   condition   **GOTO** label
IF   condition   **CALL**   label
  condition list:
    BUTTONOK   - if button **OK** pressed in the message box.
    BUTTONYES   - if button **YES** pressed in the message box.
    BUTTONNO   - if button **NO** pressed in the message box.
    BUTTONCANCEL   - if button **CANCEL** pressed in the message box.
   DONE   - if selected process e.g. **AUTOPROGRAM**, **Read File** etc. finished successfully.
    FAILED   - if selected process e.g. **AUTOPROGRAM**, **Read File** etc. failed.
    CONTROL = number   - if status from the FCONTROL function = NUMBER

FCONTROL type argument   - call the external function from FxControl DLL
PAUSE   number   - pause in miliseconds - 1 to 100000 range (1ms to 100 s).

OPENDLLFILE   filename   - FxControl DLL file - Full path and DLL File name.
LOADCFGFILE   filename   - Configuration file - Full path and File name.
LOADCODEFILE   filename   - Code file - Full path and File name.
LOADPASSWFILE   filename   - Password file - Full path and File name.
LOADSNFILE   filename   - File with Serial Number list - Full path and File name.
RESET   -equivalent to pressed button RESET on the main dialog screen.
AUTOPROGRAM   -equivalent to pressed button AUTOPROGRAM on the main dialog screen.
VERIFYFUSE   -equivalent to pressed button VERIFY SEC. FUSE on the main dialog screen.
VERIFYPASSWORD   -equivalent to pressed button VERIFY PASSWORD on the main dialog screen.
ERASEFLASH   -equivalent to pressed button ERASE FLASH on the main dialog screen.
BLANKCHECK   -equivalent to pressed button BLANK CHECK on the main dialog screen.

WRITEFLASH	-equivalent to pressed button WRITE FLASH on the main dialog screen.
VERIFYFLASH	-equivalent to pressed button VERIFY FLASH on the main dialog screen.
READFLASH	-equivalent to pressed button READ/COPY on the main dialog screen.
READSN	-equivalent to pressed button READ SN on the main dialog screen.
WRITECSMPASSWORD	-equivalent to pressed button Write CSM Password on the main dialog screen.

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

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

END   - end of script program.

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

FCONTROL type argument   - call the external function from FxControl DLL

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

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

Parameters type and argument are specified in the script file and are transferred from the programming software to DLL. Status from F_Control and message are transferred from DLL to programming software.

Programming software package contains the source code of the user defined DLL. Package has been prepared using MS Visual C++.net package. Source code is located in directory

    C:\Program Files\Elprotronic\FxControl-DLL

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

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

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

   OPENDLLFILE   filename

Below is an easy script file contents that allows to create following sequence;
1. Vcc supplied to target device is turn-OFF and first message box with buttons OK/CANCEL is displayed. Programmer is waiting until button OK or CANCEL is pressed.
2. When confirmed, then first configuration file **test-A.cfg** is downloaded to programmer. Configuration file **test-A.cfg** should be prepared first using programming software with desired configuration, selected desired code file etc. Programmer's configuration should be saved using **"Save setup us .."** option.
3. When test code is downloaded and processor started (if enabled in **test-A.cfg** file) then message box is displayed and software is waiting until button YES / NO is press. Meantime manual target's device test can be done. If test is positive, then button OK should be pressed. Or button NO if test failed.
4. When button OK has been pressed then programmer downloads **finalcode.cfg** configuration file to programmer. Current configuration can activate serialization if required, reload final code to be downloaded etc. When the new configuration is reloaded then final code is downloaded to target device, serialization is created etc.
5. On the end programmer returns to beginning and waiting for the next target device to be connected.

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

  IF BUTTONCANCEL GOTO finish
  LOADCFGFILE  C:\Program Files\Elprotronic\C2000\USB FlashPro2000\test-A.cfg
  MESSAGEBOX    OK
   "Press OK to download the test program."
  AUTOPROGRAM

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

```
    IF BUTTONNO GOTO START

  LOADCFGFILE  C:\Program Files\Elprotronic\C2000\USB FlashPro2000\finalcode.cfg
   AUTOPROGRAM
   GOTO START

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

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

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

  IFGUIMSGBOXPRESS   CANCEL          ;press CANCEL if GUI box is generated,
                                     ;but disabled to be displayed,
 ; ifGuiMsgBoxPress   cancel         ;This is OK also. Commands are not case
                                     ;sensitive
  LOADCFGFILE   test-script.cfg   ;recommended full path and name
  IF FAILED GOTO fileerror

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

  AUTOPROGRAM

  IF DONE CALL testOK
  IF FAILED CALL testFailed

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

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

>testFailed
```

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

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

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

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

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

    FCONTROL   - calls external user defined function in the DLL

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

Required DLL file should be created first.

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

  IF BUTTONCANCEL GOTO finish
  LOADCFGFILE   C:\Project\test-A.cfg
  MESSAGEBOX    OK
   "Press OK to download the test program."
```

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

11. Project and Configuration Load / Save

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

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 FlashPro2000 Programmer provides the option to Lock configuration settings. When the user selects the *Lock/Unlock Setup* option from the Setup menu, the FlashPro2000 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.
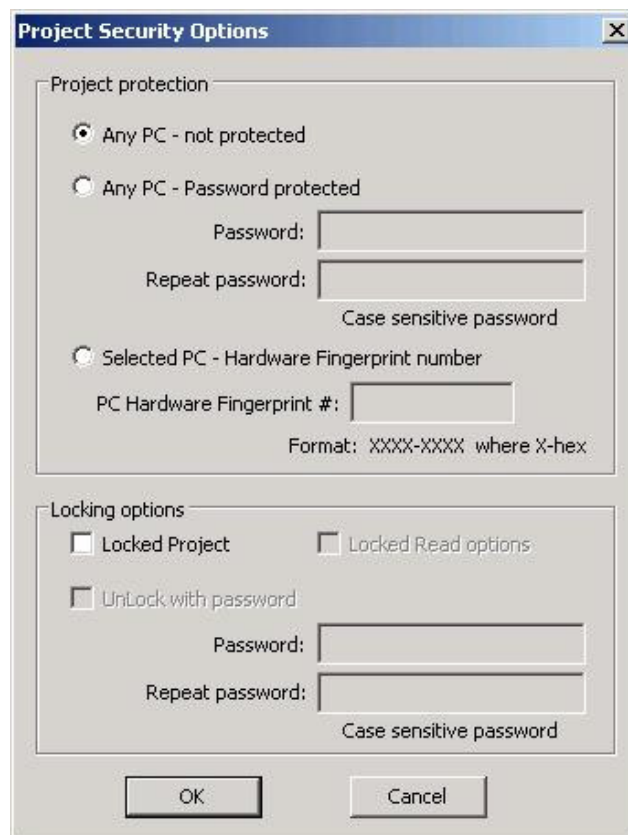
11.2 Load / Save Project

The Project option (Save/Load) contains more then the programmer configuration only, but can also the code and the CSM 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 (FlashPro2000, GangPro2000, etc.) used password or destination's PC "hardware fingerprint" number. So - the project file created with the FlashPro2000 software cannot be used with the FET-Pro430 or GangPro430 and vice-verse. Each project file should be create in the same type of software. Project file is CRC protected and CRC check is performed when the file is loaded .

Project can be unprotected or protected with the destination PC "hardware fingerprint" number or password protected. This allows to create the project that can be used only on the specific PC when the project is encrypted with the destination PC "hardware fingerprint" number (useful in

production) or create the project that can be used only when the correct password is entered every time when the project is open. Project can be unlocked or locked with almost all blocked buttons and pull down menu items. When the project is locked, then only major buttons like *Autoprogram* or *Verify* are active - and only a few pull-down menu items are accessible. All options that allows to read the code contents are blocked.

When the new project is create then it is recommended to select the *New Setup* from pull down menu and set the default option of all parameters and names used in the programmer. As the next - the desired processor, code file, password file if required and all desired option (see all available options described in this manual) should be selected. When it is done, it should be verified if programmers works as expected. When all works, then the current setup can be saved as the project file. Select the *Save Project as..* from **File** pull down menu. Following dialog will be displayed (Figure 11.2-1) 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 dialog screen (figure 9.2-1). This number has hardcoded format and contains eight hex characters with dash between 4th and 5th character eg.

<div align="center">

6FA4-E397

</div>

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)

Locking option:

Figure 11.2-2

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" dialog screen with selected options

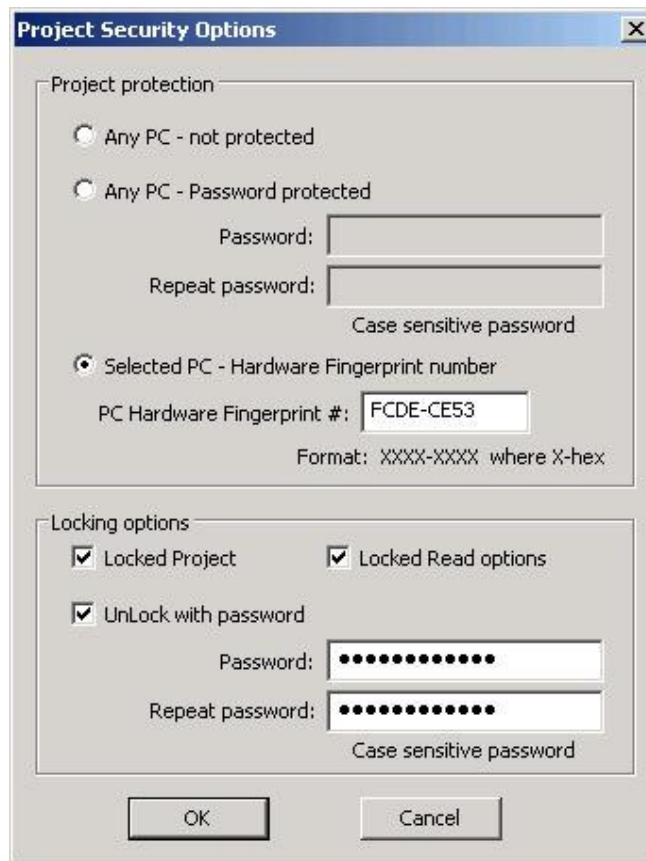   Project protected with *PC's "hardware fringerprint"* number, locked and unlocked with password.

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 dialog screen are not blocked.

11.3 Commands combined with the executable file

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

-prj Project file name (Open Project file)

-sf   Setup_file_name (Open Setup file)

-cf   Code_file_name (Open Code file)

-pf   Password_file_name (Open Password file)

-nf   SN_ file_name (Open Serial number list file)

-rf   Script_file_name (Run programming sequence from the Script File)

-lock

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

When the -prj option is used, then the -sf, -cf, -pf, -rf options are ignored.

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

"C:\Program Files\Elprotronic\C2000\USB FlashPro2000\FlashPro2000.exe"

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

"C:\Program Files\Elprotronic\C2000\USB FlashPro2000\FlashPro2000.exe" -sf E:\ElproTronic\MFG\prg-04.cfg
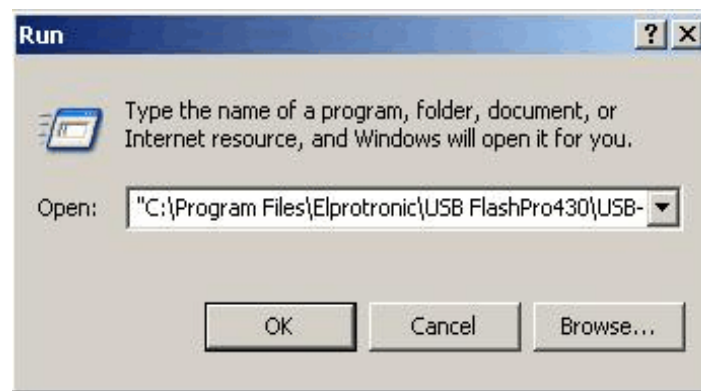
Figure11.3-1
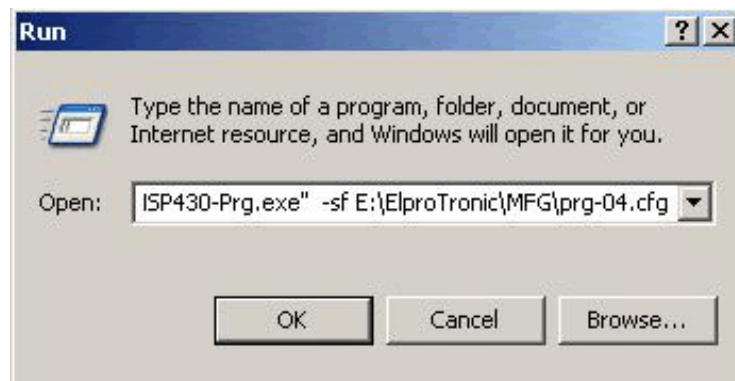
Figure 11.3-2

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

"C:\Program Files\Elprotronic\C2000\USB FlashPro2000\FlashPro2000.exe" -lock -sf E:\ElproTronic\MFG\prg-04.cfg

or

"C:\Program Files\Elprotronic\C2000\USB FlashPro2000\FlashPro2000.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) Using Browse.. in the Create Shortcut dialog box select the following executable file
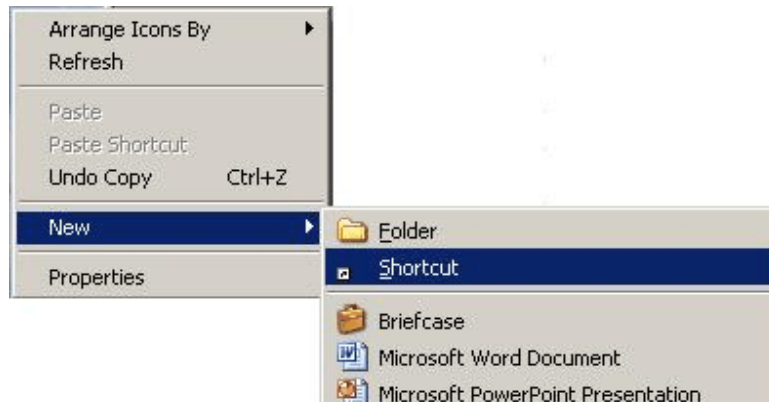

Figure 11.3-3

"C:\Program Files\Elprotronic\C2000\USB FlashPro2000\FlashPro2000.exe"

Figure 11.3-4

(see Figure 11.4) and at the and add the required command keys (see Figure 11.5) eg.

"C:\Program Files\Elprotronic\C2000\USB FlashPro2000\FlashPro2000.exe" -lock -sf E:\ElproTronic\MFG\prg-04.cfg

Click button *Next* and follow instruction to create icon. Using *Copy* and *Paste* and modify required configuration file names a lot of icons can be created with independent configuration setups. Clicking on the selected icon FlashPro2000 programming software will start with the selected
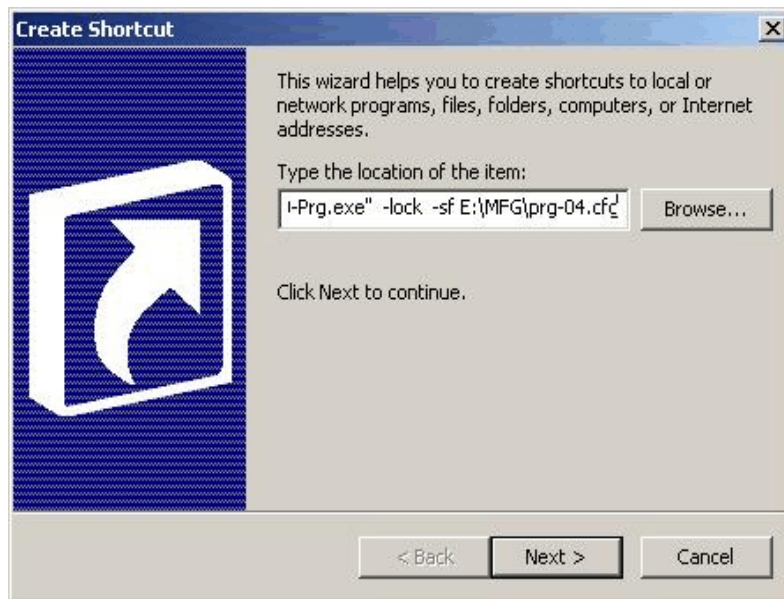

Figure 11.3-5

configuration setup, and locked if required.

12. Target connection

The Flash Programming Adapter (FPA 4.4) contains 14-pins header connector that support the JTAG and SCI-BOOT communication with C2000 DSPs. The FlashPro2000 hardware package contains two adapters that can convert the pinout from FPA to desired interface

1. **C2000-JTAG** adapter (Figure 12.1 and 12.2) that convert the pinout from FPA adapter to standard TI's JTAG presented in the TMS320F2xxx DSP

2   **C2000-SCI-BOOT** adapter (Figure 12.3 and 12.4) that convert the pinout from FPA adapter to SCI-BOOT customized adapter

The definition of all the pins in the FPA adapter is given in the tables 12-1.

Table 12.1   FPA Interface connector

Pin #	Name	JTAG	SCI-BOOT
1 (Red)	TDO	JTAG Data output from target device - input to FPA	Not used.
2	VCC-In / Sense	Do not use in the FlashPro2000 application	Do not use in the FlashPro2000 application
3	TDI	JTAG Data Input to target device - output from FPA	Not used.
4	Sense	Target's Device Vcc Sense 0 to +5V -iInput to FPA	Target's Device Vcc Sense 0 to +5V - Input to FPA
5	TMS-In	TMS Input to target device - output from FPA	BOOT-SCI selection - output from FPA
6	NC	EMU-1 - output from FPA	DSP Reset XRSn - output from FPA
7	TCK-In	JTAG TCK Input pin - output from FPA	Not used.
8	TEST	JTAG RESET - output from FPA	Not used.
9	GND	Ground	Ground
10	NC	Not used.	Not used.
11	\RST	EMU-0 - output from FPA	Not used.
12	SCI-TX	Not used.	SCI-TX from the target - input to FPA
13	NC	Not used.	Not used.
14	SCI-RX	Not used.	SCI-RX to target - output to FPA

12.1 JTAG connection
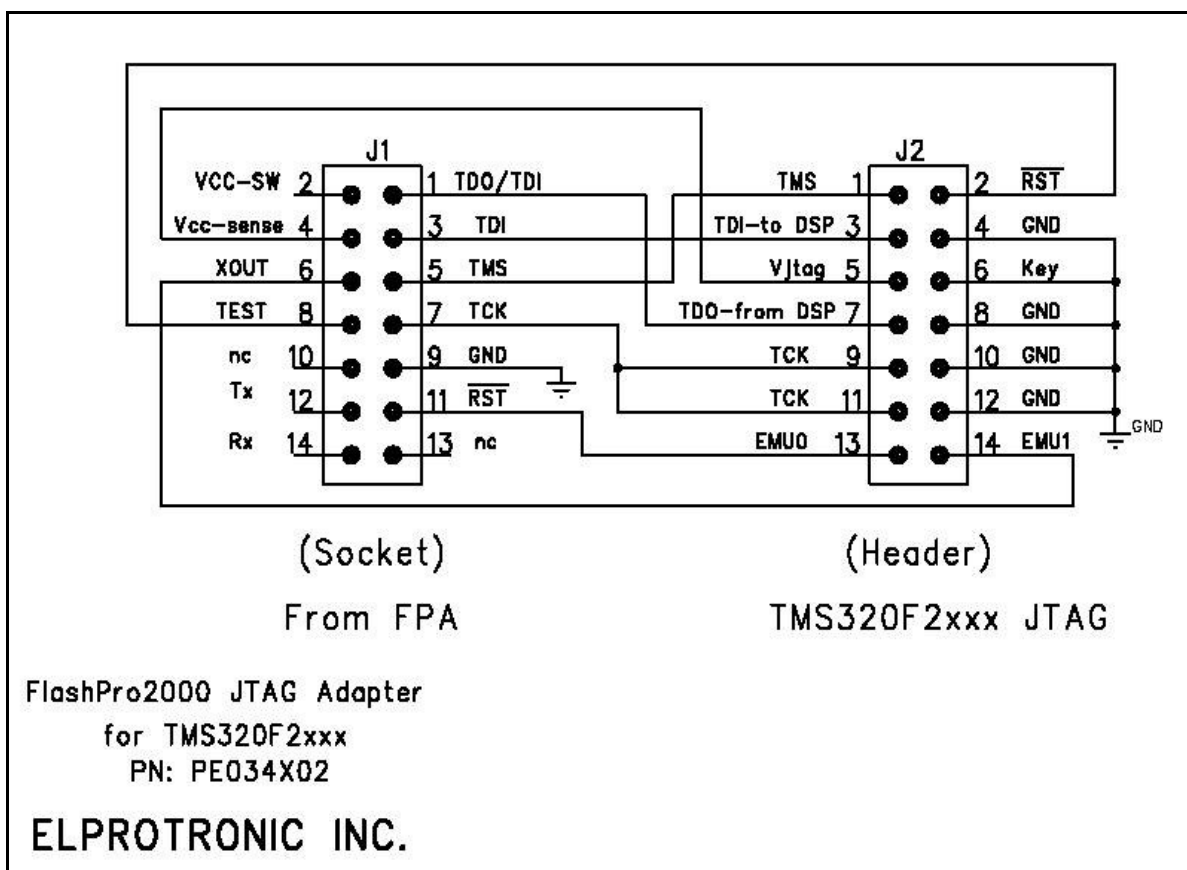

Figure 12.1

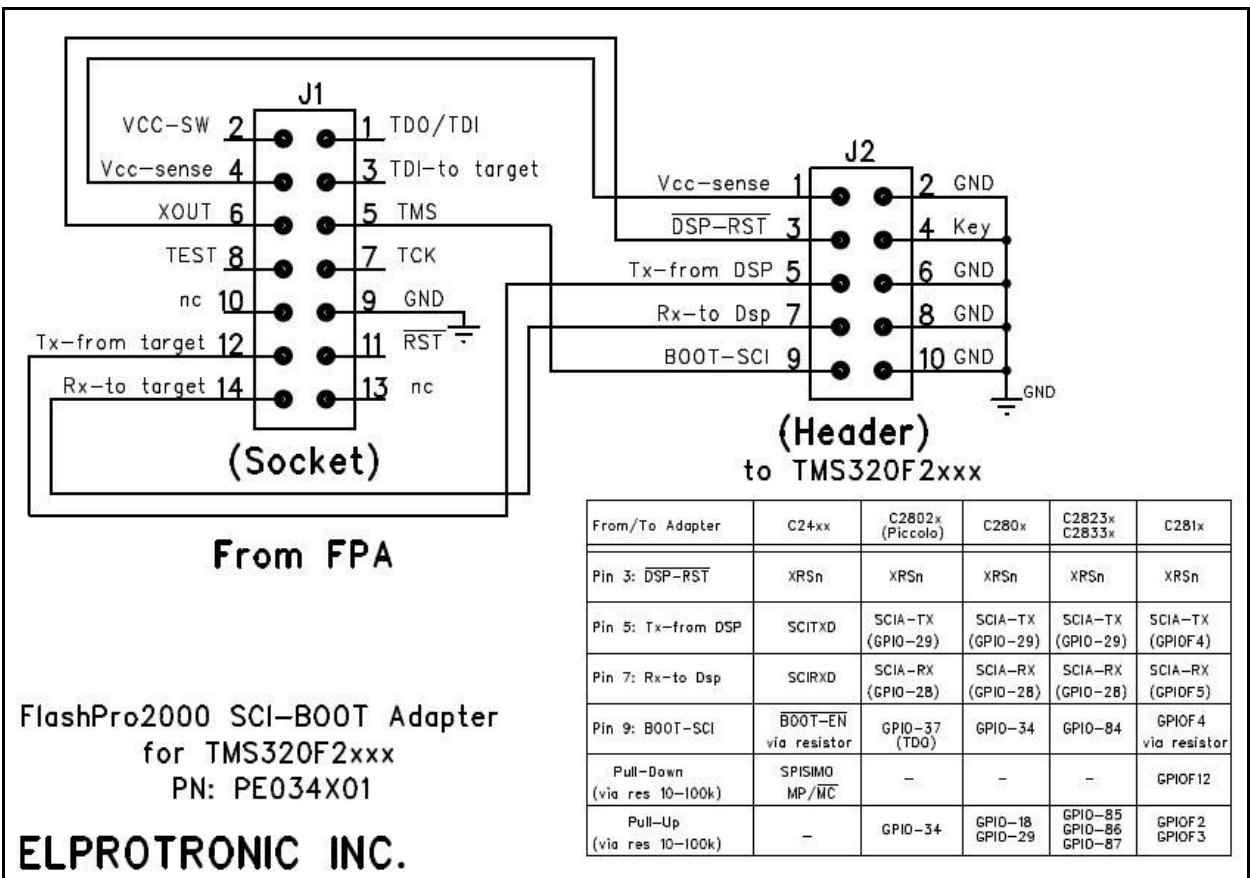Figure 12.2

12.2 SCI-BOOT connection

Figure 12.3

Figure 12.4

When the JTAG communication is used then the **C2000-JTAG** adapter should be plug-in to FPA adapter and using 14-wires ribbon cable the C2000-JTAG adapter should be connected with standard TI's 14-pins JTAG connector as it shown on Figure 3.2-1.

When the SCI-BOOT interface is used then the **C2000-SCI-BOOT** adapter should be plug-in to FPA adapter and using 10-wires ribbon cable the C2000-SCI-BOOT adapter should be connected with customized 10-pins SCI-BOOT connector. The SCI-BOOT connections are not the same for all TMS320F2xxx microcontrollers and the connection related to used hardware should be created. See TI's documentation for details. Below are presented schematics of the typical connection to target devices (Figure 12.5, 12.6, 12.7, 12.8 and 12.9).
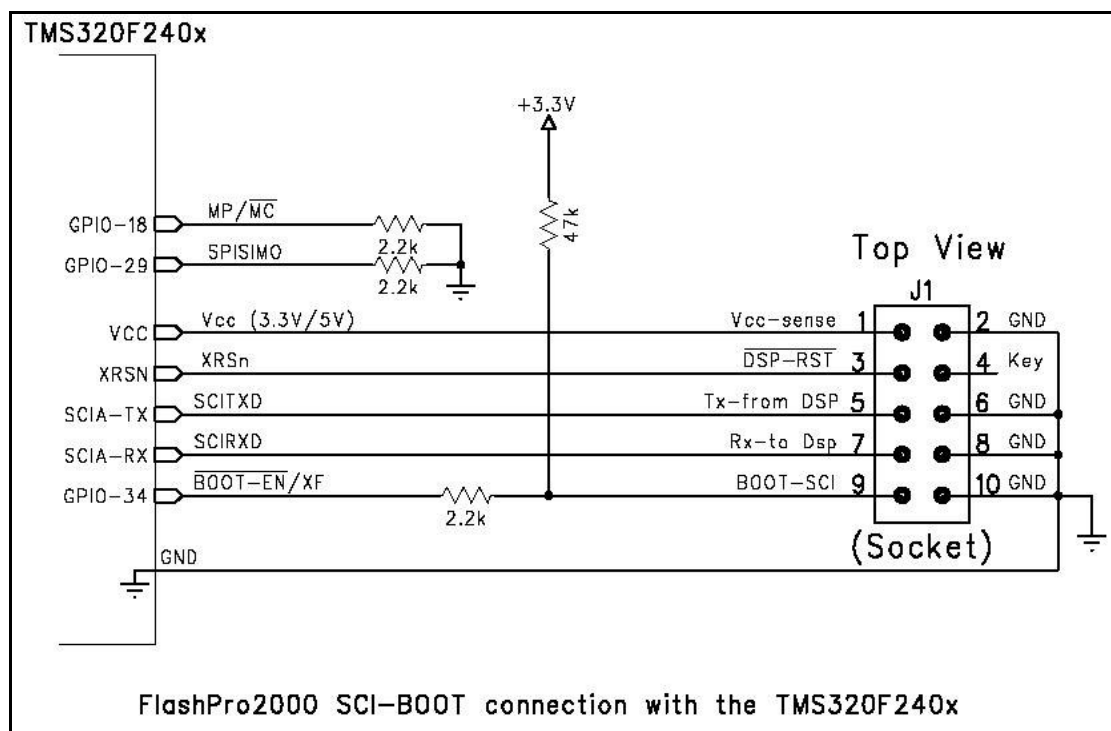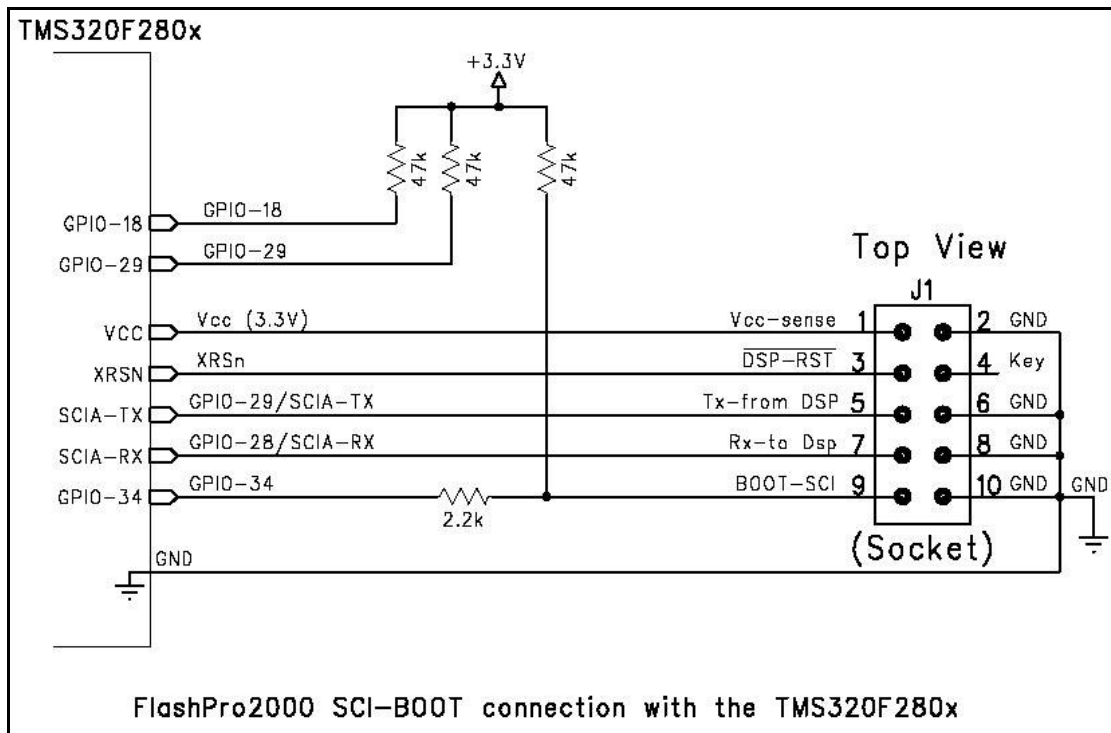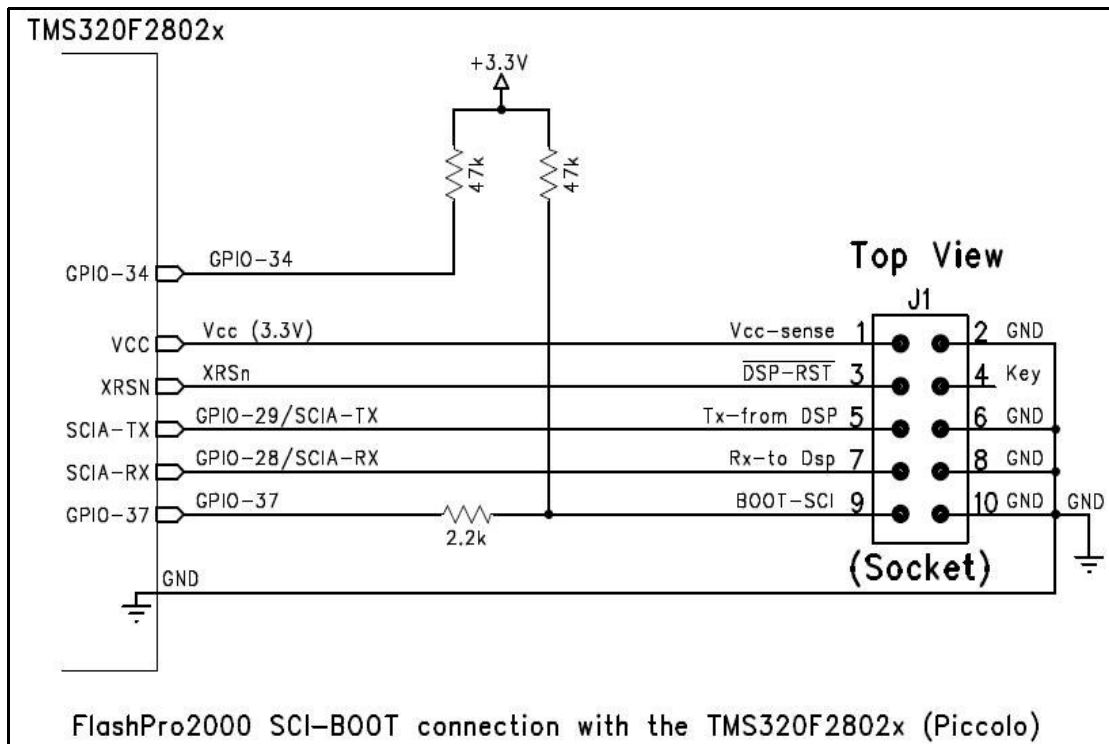

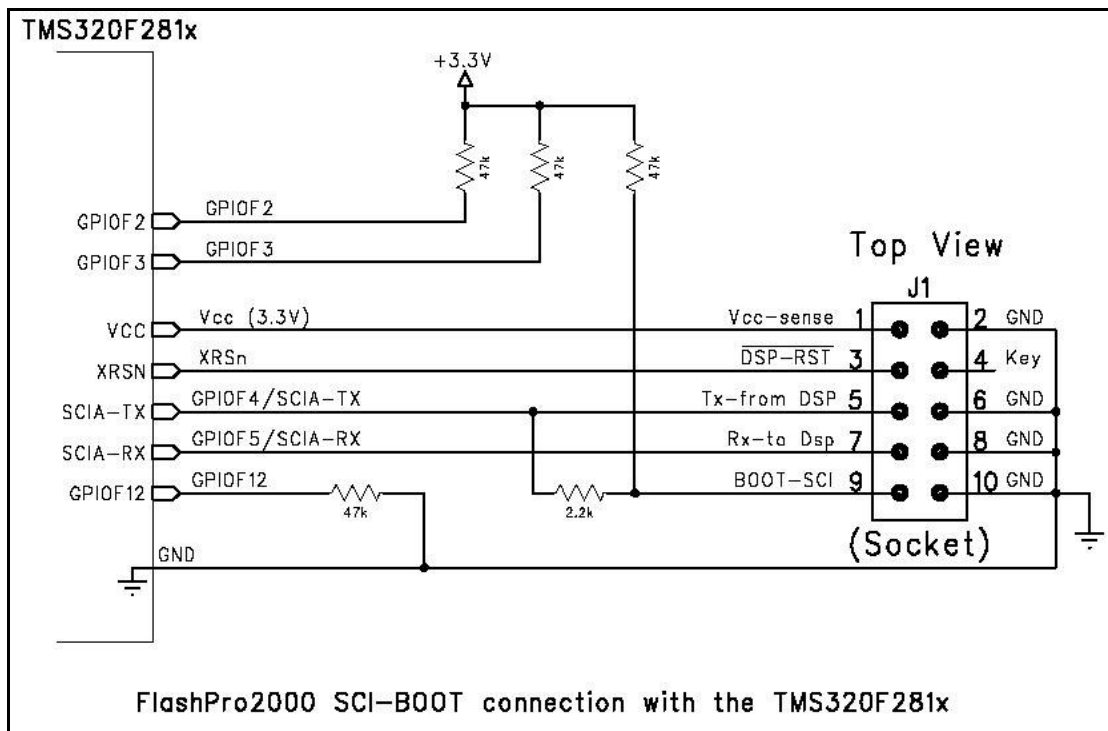Figure 12.5

Figure 12.6

Figure 12.7

Figure 12.8
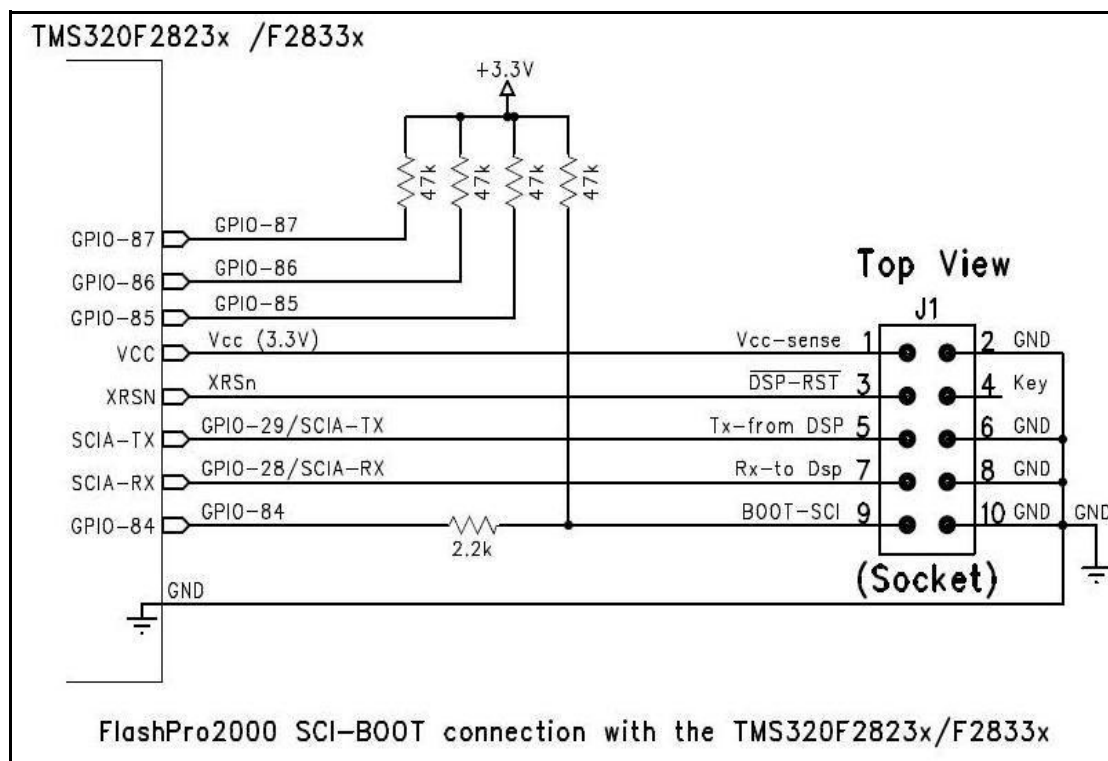
Figure 12.9

Appendix A - *Specification*

Specification:

FPA adapters - **USB-FPA-4.4**

PC Communication Interface:	- Full Speed USB-1.1 (12Mbits/s)
USB connector	- Adpater site: USB-type B, Computer site: USB-type A
Target connector	- 14 pins header connector with JTAG and SCI-BOOT communication capability. Adapters required to convert the pinout from FPA to TI's standard JTAG or SCI-BOOT.
DC Power - from USB Interface	- 5V +/- 20%, 70mA + target's current (0-100mA)

Target Device DC supply

- external	- 3.0 V to 5.0 V

Communication speed via JTAG Interface

Fast JTAG	- up to 3 Mb/s
Slow JTAG	- up to 1Mb/s

Communication speed via SCI-BOOT Interface

Initialization	- UART - baud rate 5 to 20 kb/s
Standard communication	- up to 500 kb/s
Size:	- 76 x 43 x 20 mm (3.0 x 1.68 x 0.8 inch)
Verification Compliance:	- CE (European CISPR 22 and EN 55022).
	- FCC Part 15, Subpart B- Class B Unintentional Radiators for Uses in Home, Commercial and Industrial Areas.

Figure A-1

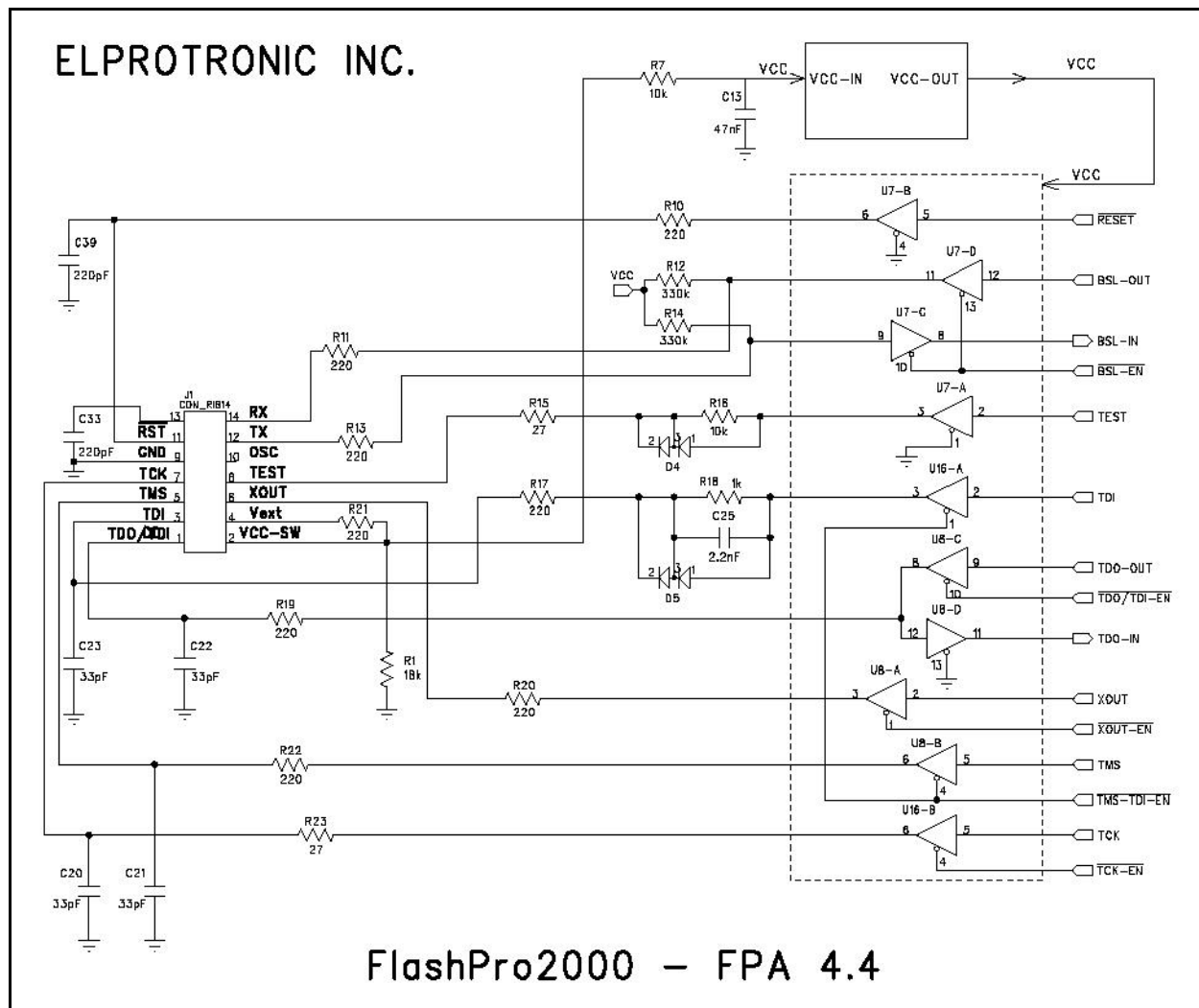I/O schematic of the FPA-4.4 in the FlashPro2000 is shown on Figure A-2.

Figure A-2

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

[Elprotronic Inc.](#):

[XS-FP2000](#)