

# C2000Prog 1.3 - Manual

---

## Table of Contents

1 - Overview.....	1
2 - Quick Start.....	2
3 - Detailed Description.....	5
4 - Command Line Options.....	6
5 - CRC Checksum.....	7
6 - RTS/DTR Control.....	9
7 - Appendix.....	10

## 1 Overview

C2000Prog is a flash programming tool for TI C2000™ MCUs. Instead of using JTAG as the communication interface between the programming tool and the MCU, C2000Prog utilizes RS-232, RS-485 and CAN (Controller Area Network). The programmer is, therefore, well suited for deployment in the field where the JTAG port is typically not accessible.

Some salient features of the programmer are:

- Fast communication protocol that works reliably with USB-to-RS232 converters
- Support for point-to-point as well as multidrop networks
- Smart detection of which Flash sectors need to be erased (or manual section selection if desired)
- Automatic 32-bit CRC (Cyclic Redundancy Check) generation and programming (allowing the firmware to verify the flash integrity at MCU bootup)
- Compatible with standard Intel Hex file allowing for other data (such as FPGA code) to be programmed into DSC Flash
- Extended hex files that can be encrypted and contain all the settings for programming, including secondary bootloader
- Remote link files for programming extended hex files stored on a server
- Command-line interface, callable by other programs, for example for batch programming
- DTR/RTS control for resetting MCU in bootloader mode

## 2 Quick Start

### 2.1 Installation

The most recent version of the programmer can be downloaded from the CodeSkin website:

<http://www.codeskin.com/c2000Prog.html>

The application is installed using the Windows installer “setup.exe”. It can also be uninstalled from the control panel similar to other Windows software. C2000Prog requires a JAVA runtime environment (JRE 1.5 or more recent) to be installed. The JRE can be downloaded free of charge from the [Sun Microsystems](#) website.

### 2.2 Hex File Generation

C2000Prog supports Intel-Hex files with 16 bit address and data widths. They can be generated directly from the COFF files with the following command:

c2400-tools:

```
dsphex -romwidth 16 -memwidth 16 -i -o .\Debug\code.hex .\Debug\code.out
```

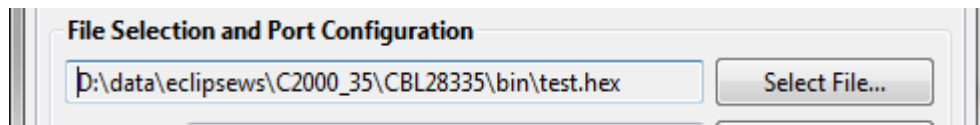
c2000-tools:

```
hex2000 -romwidth 16 -memwidth 16 -i -o .\Debug\test.hex .\Debug\test.out
```

For Code Composer Studio users it is recommended that this conversion be configured as a final build step. Note that regular COFF files (\*.out) are *not* supported.

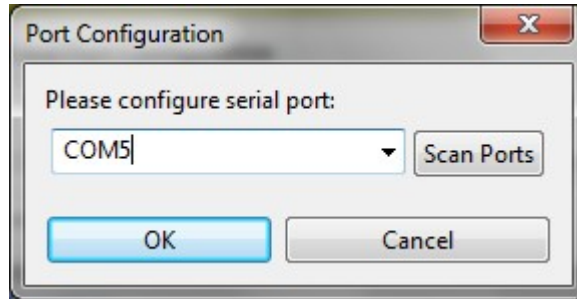
### 2.3 Programming over RS-232

In order to program an MCU using C2000Prog, first the hex file needs to be selected by means of the “Select File...” button.

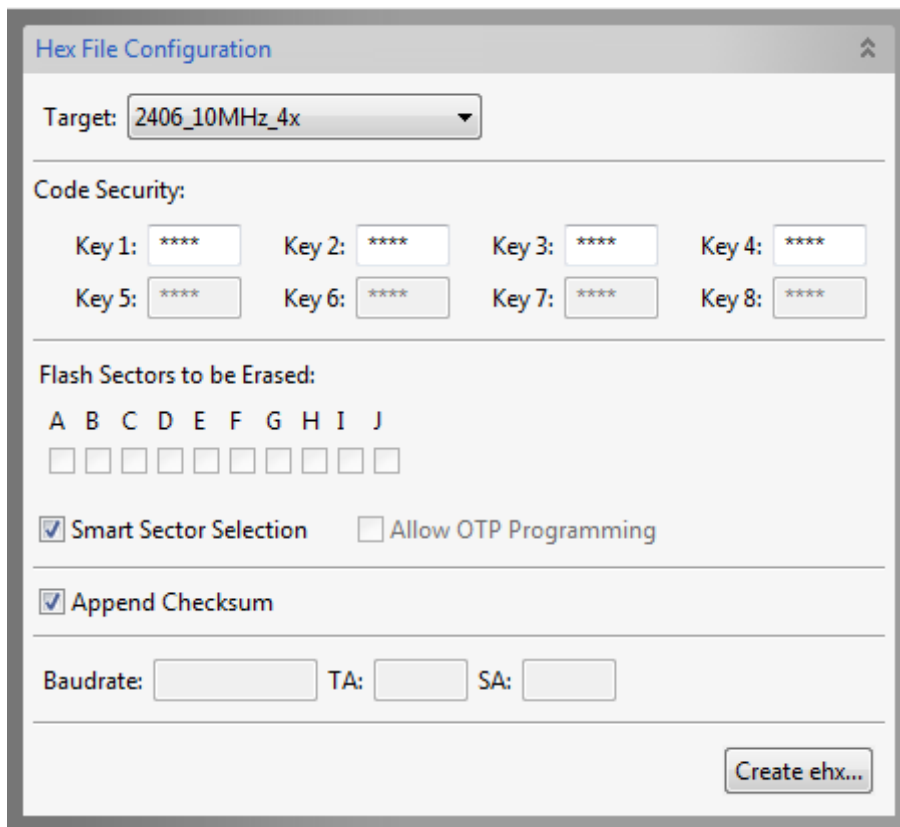


Next, the COM port must be configured, either by typing its name directly into the text-field, or clicking on the “Configure Ports...” button. Valid entries on the windows platform are COM1, COM2, etc.

The “Scan Ports” button in the Port Configuration Dialog automatically scans for all available serial ports. Note that on some computers this feature can take a very long time to execute.



An Intel hex file only contains raw flash data. It does not define any target specific information such as the type of MCU to be programmed, its oscillator frequency, the communication protocol to be used, etc. This information must be configured manually in the “Hex File Configuration” panel.



First, the target must be selected by choosing the appropriate combination of MCU part-name and clock frequency. Contact CodeSkin at [info@codeskin.com](mailto:info@codeskin.com) if no match is found.

For programming a locked MCU, valid CSM keys (passwords) has to be entered (as 4 digit hex numbers).

Next, the sectors requiring erasing need to be selected. This can either be done manually by checking the individual boxes, or by choosing “Smart Sector Selection”. The smart sector feature automatically detects which sectors need erasing by parsing the contents of the hex file.

As a final option, “Append Checksum” can be selected, which causes the programmer to append a 32-bit CRC checksum to the hex data. This checksum can be used by the MCU to verify the integrity of the flash data, as described later in this document.

With all hex file configurations made, the flash can be programmed by hitting the “Program” button. This will open a new window which displays status information as the programming progresses.

## **2.4 Extended Hex Files**

The hex file configuration and contents of the Intel hex file can be combined and saved/distributed as an “Extended Hex File” (\*.ehx). This format is preferable over the raw hex file, as it allows programming without requiring any manual configuration of the programmer options.

From the graphical user interface of the programmer a ehx file can be generated by clicking on the “Create ehx...” button. An even more convenient approach is to automatically generate the ehx file after the hex file has been created from the COFF file. This is easily achieved by calling C2000Prog via its command line options, as below:

```
C2000ProgConsole.exe -shell -target=28335_30MHz -hex=test.hex -ehx=test.ehx
```

Again, it is recommended that this command be configured as a final build step.

Extended hex files can also be configured to include “License Conditions” that must be accepted by the user before C2000Prog allows programming. This is achieved by means of the “-license” command line option as explained later in this document.

## **2.5 Remote Hex Files**

Remote hex (rhx) files are a means for distributing a *link* to an ehx-file instead of the actual ehx-file. They are created from the GUI (“Special” menu) or via command line options.

Examples for valid links are:

file:///c:/file.ehx

http://hostname/directory/file.ehx

http://username:password@hostname/directory/file.ehx

<ftp://username:password@hostname/directory/file.ehx>

### 3 Detailed Description

The CodeSkin flash programming client C2000Prog is not only suitable as a development tool, but has also been designed for batch programming during manufacturing, and for firmware upgrades in the field.

It is written in Java and therefore easily portable to platforms other than Windows. C2000Prog is free and can be downloaded from <http://www.codeskin.com/c2000Prog.html>. Its installation does not require administrative rights (as long as a current Java runtime environment is installed). It is also possible to [execute C2000Prog](#) without a local installation by means of the Java Web Start technology. C2000Prog is typically run with a graphical user interface. However, it can also be called in the background from another program (via command-line options).

C2000Prog uses a file-format called “extended hex” (\*.ehx) for storing both the firmware and the target configuration (chip, CSM keys, etc). The ehx file can be password protected (encrypted) for IP and CSM-key protection. It is also possible to embed license conditions in an ehx file, which the user has to accept before the flash is reprogrammed. The ehx file is usually created automatically as the final build step (by calling C2000Prog via the command line interface from Code Composer Studio). It can also be generated from the C2000Prog graphical interface.

As an alternative to distributing ehx files, a “remote hex” file (\*.rhx) can be generated. A rhx file contains no actual programming or target configuration data, but only a link to an ehx file stored on a remote site (http or ftp). The access to the remote site as well as the rhx file itself can be password protected. The advantage of distributing rhx files rather than ehx files is that the programming data remains under control of the originator and can be retired (removed from the remote site) upon obsolescence.

Another important feature of C2000Prog is that it can be configured to append a 32-bit cyclic redundancy code (CRC) to the flash data. This allows the firmware to verify the Flash integrity at bootup and during operation.

In addition to RS-232, C2000Prog supports several other communication protocols, including multidrop RS-485 and CAN (Controller Area Network).

## 4 Command Line Options

C2000Prog can be launched from a command prompt with command line options. The executable to be called for this mode is "C2000ProgConsole.exe".

C2000Prog takes the following command line options:

-shell	enables shell operation (without launching GUI)
-target=TARGET_ID	selects target (based on name entry in target.xml/targets.custom.xml file)
-port=COM_PORT	selects communication port
-hex=FILE_NAME	selects hex/ehx file to be programmed
-keys=KEY1,KEY2, KEY3, ..	configures keys for unlocking flash (optional)
-sectors=SECTOR_MASK	configures which flash sectors are erased, where SECTOR_MASK is a hex number -sectors=1: sector A -sectors=2: sector B -sectors=3: sectors A&B -sectors=A: sectors B & D if the "-sector" option is not used, then sectors to be erased are automatically detected
-crc	enables addition of CRC checksum (optional)
-ehx=FILE_NAME	creates an extended hex file (ehx) or remote hex file (rhx) instead of programming the flash
-pass=PASS_PHRASE	pass-phrase for extended hex-file
-license=FILE_NAME	embeds the contents of a text file as a licensing-condition
-url=URL	URL for generating remote hex files (rhx)
-baud=BAUDRATE	baudrate for some protocols (such as CAN)
-ta=TARGET_ADDRESS	Target-address for multidrop protocols (such as CAN)
-sa=SOURCE_ADDRESS	Source address for multidrop protocols (such as CAN)

Examples:

The following command programs the flash using a password protected extended hex file:

```
C2000ProgConsole.exe -shell -hex=test.ehx -pass="very secret" -port=COM1
```

The command below programs the flash of a 2811 with an external clock of 30 MHz. The target is connected to serial port COM4 and the hex-file selected is "test.hex". All keys to unlock the flash are specified as 0x1234 and the sectors selected to be erased are A,B,C and D (hex 0xF = binary 1111).

```
C2000ProgConsole.exe -shell -target="2811_30MHz" -port=COM4 -hex=test.hex \
    -keys=1234,1234,1234,1234,1234,1234,1234,1234 \
    -sectors=F -crc
```

An extended hex file with password protection and license agreement is created as shown here:

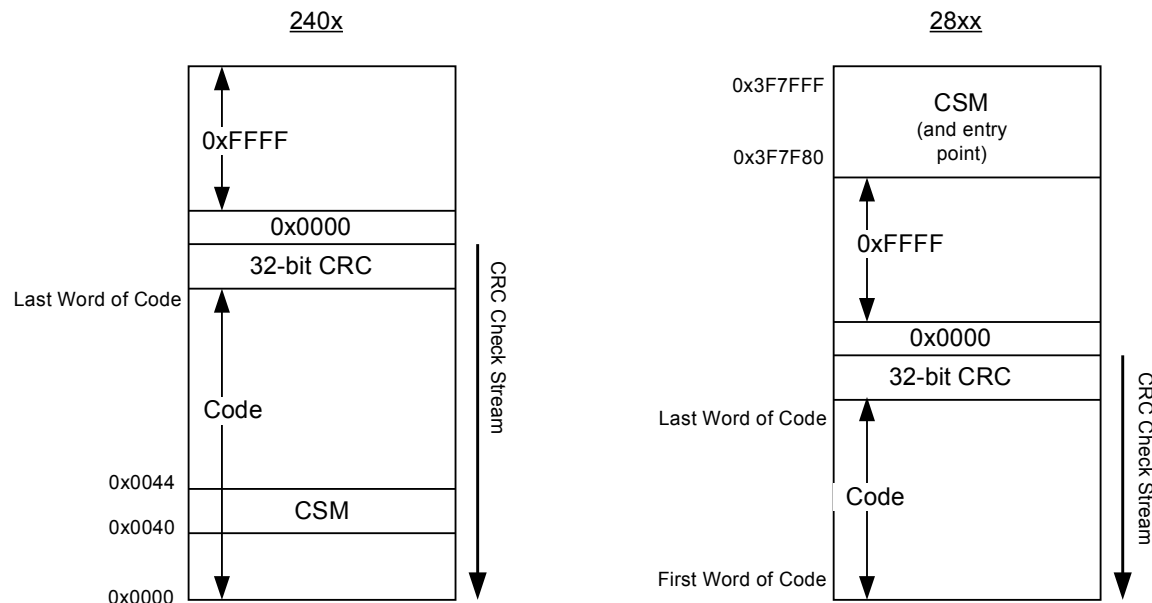
```
C2000ProgConsole.exe -shell -target="2811_30MHz" -hex=test.hex\
    -keys=1234,1234,1234,1234,1234,1234,1234,1234 \
    -sectors=F -crc -ehx=test.ehx -pass="very secret" \
    -license=license.txt
```

This creates a password protected remote rhx file:

```
C2000ProgConsole.exe -shell -url="http://hostname/directory/file.ehx"\
    -ehx="test.rhx" -pass="very secret"
```

## 5 CRC Checksum

C2000Prog can be configured to automatically append a 32-bit CRC to the code being programmed into flash. This allows for the embedded application to verify the flash integrity at each powerup or even periodically during operation.



If the “Append CRC” box is checked, or “-crc” command line option is used, C2000Prog will first parse the hex-file and determine the lowest and highest address to be programmed. For a 240x MCU, this includes the CSM zone (4 keys), for a 28xx MCU, the CSM zone is ignored (including keys, reserved words, and program entry points). Next, the 32-bit CRC is calculated and appended at the top of the memory, i.e. at the two addresses above the highest address of the hex-file determined before. In addition, a CRC delimiter, one zero word (0x0000), is added on top of the two CRC words.

The 32-bit CRC algorithm used has the following parameters:

- Polynomial: 0x04c11db7
- Endianess: big-endian
- Initial value: 0xFFFFFFFF
- Reflected: false
- XOR out with: 0x00000000
- Test stream: 0x0123, 0x4567, 0x89AB, 0xCDEF results in CRC = 0x612793C3

In contrast to a typical data-stream with the CRC transmitted at the end, the C2000Prog CRC must be verified by processing the flash data starting with the CRC, i.e. one memory address below the CRC delimiter (0x0000). A successful data-verify results in a CRC register value of zero (0x00000000).

A typical flash verification algorithm running at DSP powerup appears as follows:

1. Set a memory pointer to the highest possible program address.
2. Decrement the pointer until it points to the CRC delimiter (0x0000), skipping all 0xFFFF values.
3. Decrement the counter by one more address (at which time it points to the first CRC word).
4. Initialize the CRC register to 0xFFFFFFFF.
5. Update the register with the value addressed by the memory pointer (CRC polynomial: 0x04C11DB7).
6. Decrement memory pointer.
7. Repeat 5-6 until the memory pointer reaches the lowest program address.
8. If at this point the register holds 0x00000000, then the data integrity has been successfully verified.

Sample Code for 28xx with code in flash sector A:

```
#define FLASH_TOP (const Uint16*) (0x3F7F7FL)
#define FLASH_BOT (const Uint16*) (0x3F6000L)

const Uint16* FlashPtr;
Uint32 CRCRegister;
```

```

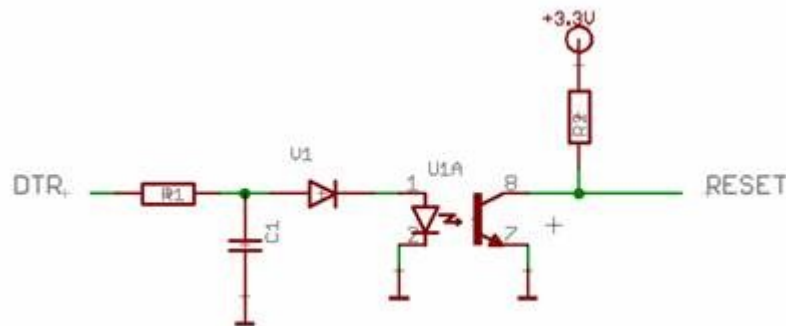
FlashPtr = FLASH_TOP;
// search for CRC delimiter
while((*FlashPtr != 0x0000) && (FlashPtr > FLASH_BOT)){
    FlashPtr--;
}
// process stream, CRC first
CRCRegister = 0xFFFFFFFF;
while(FlashPtr > FLASH_BOT){
    FlashPtr--;
    // each CRC32Step() shifts one byte into the CRC register
    CRCRegister = CRC32Step((*FlashPtr >> 8) & 0xFF, CRCRegister);
    CRCRegister = CRC32Step((*FlashPtr >> 0) & 0xFF, CRCRegister);
}
// at this point CRCRegister should be reading zero

```

## 6 RTS/DTR Control

RTS/DTR control allows C2000Prog to select bootload mode (using RTS) and reset the target (using DTR). In order to use this feature, the hardware must be setup so that a positive DTR line resets the target and a positive RTS selects bootload mode.

Below is an example of how such a control can be implemented.



# 7 Appendix

## 7.1 License

Except where otherwise noted, all of the documentation and software included in the C2000Prog package is copyrighted by CodeSkin, LLC.

Copyright (C) 2006-2010 CodeSkin, LLC. All rights reserved.

Permission is granted to anyone to use this software for any legal purpose, including commercial applications.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 7.2 Customizing C2000Prog

The principal C2000Prog configuration file is "targets.xml" in the "targets" folder. It contains a description of all targets that are supported by the programmer as well as some baudrate and timing options. Limited customization of this file is possible – please contact codeskin.com for more details. The programmer will also parse a file named "targets.custom.xml", if present. It is recommended that "targets.xml" be renamed to "targets.custom.xml" before any customization is made. This ensures that a reinstall or upgrade of the tool, which preserves "targets.custom.xml", will not erase those custom settings.

C2000Prog also parses the user directory as an alternate location for storing target configuration files. On a Windows machine, this corresponds to the following location:

XP:	C:\Documents and Settings\username\codeskin\targets
Vista/Windows-7:	C:\users\username\codeskin\targets

On a Linux box, the corresponding location is

/home/username/.codeskin/targets

## 7.3 About Bootloaders

When programming, C2000Prog is interacting with a so called “bootloader” running on the target. This bootloader is often divided into two components:

- **Primary Bootloader (PBL):** The primary bootloader is a small piece of code that is permanently programmed into the target and called immediately after reset. The primary bootloader can branch to the application code (if present) or receive the secondary bootloader (SBL) into RAM over the communication link and execute it. The primary bootloader typically also includes a security algorithm for unlocking the chip before the secondary bootloader can be loaded.
- **Secondary Bootloader (SBL):** Contrary to the primary bootloader, the secondary bootloader is not permanently stored in the target. Instead, it is being loaded via the primary bootloader when needed. The SBL contains the flash programming algorithms. It erases the flash, receives the application code over the communication link, and programs the flash memory. After the programming is complete, the secondary bootloader can reset the chip.

There are several advantages to dividing the bootloader into two separate parts:

1. Since the primary bootloader does not include flash programming algorithms it can have a small footprint. Due to its low level of complexity it can be validated to a high level of confidence before it is shipped with a product.
2. The secondary bootloader contains the more complex algorithms. However, since it is not permanently programmed into the target, but rather distributed with the programming tool, it can be updated easily if needed.
3. Not having any flash programming algorithms permanently programmed in the target can be considered safer in case of rogue behavior of the application code.

Almost all C2000™ MCU ship with a primary bootloader programmed into the boot-ROM. While this bootloader is supporting several communication interfaces, only the SCI mode (RS-232) is of practical use in the field (the CAN implementation is too limited). CodeSkin has developed secondary RS-232 bootloaders for most C2000™ MCUs and distributes them as part of C2000Prog programming tool. The compiled versions of the secondary bootloaders are free; if so desired, the source code can be licensed for a fee.

CodeSkin has also developed custom primary bootloaders that can be used in lieu of the TI version. They are licensed as source code, and allow for the implementation of customer specific features, such as servicing an external watchdog, and proprietary security algorithms. The CodeSkin primary bootloaders also support communication protocols other than RS-232. For example half-duplex RS-485 and CAN bus. Along with the primary bootloader, the source code of a matching secondary bootloader is provided.