FlashBurn: A DSK Flash Memory Programmer

Russ Heeschen *SDS Productivity Tools Team*

ABSTRACT

The FlashBurn utility is a Windows program that works along with Code Composer Studio and an FBTC (FlashBurn Target Component) program to burn data and programs into the FLASH memory of a DSP target board.

Contents

List of Figures

List of Tables

1 Terms

The following table defines the terms that will be used in this document.

Table 1. List of Terms

Term	Definition
Code Composer Studio	TI's DSP software development platform.
Exchange Communications Protocol	This is a proprietary protocol that defines the required commands and data communications between the FlashBurn application on the PC and your target system FBTC program.
FBTC	FlashBurn Target Component: a DSP program that is downloaded to the DSP target system by the FlashBurn application. This DSP program communicates with the FlashBurn application using the communications protocol described in the Exchange Communications Protocol document. Example FBTC programs are provided by TI with source code.
Hex Conversion Utility	A utility program installed with Code Composer Studio. The utility is generally used to read a `.out` file and a `.cmd` file to produce a `.hex` file in one of several formats. Examples are hex500.exe for the 5000 family and hex6x.exe for the 6000 family of DSPs.
COM	As used here, a feature of the Windows operating system which allows one application to use services provided by another application.

2 System Requirements

2.1 PC

- PC with Windows 98 (SE or later), NT4 (SP4 or later), or Windows 2000
- Code Composer Studio v2.0 or later, with a connection to the target system (JTAG cablem HPIB, possibly others in the future)

2.2 Target DSP Families Supported

- C5x™ (Example FBTCs provided for DSK5402, DSK5416, EVM5510, DSK5510)
- C6x™ (Example FBTCs provided for DSK6211, DSK6416, TEB6416, DSK6711)
- FBTC examples will be provided for future TI development boards as boards are made available.

3 A Typical Interactive FlashBurn Session

This description presumes that the target system is a TI DSK6211 development board. The FlashBurn installation provides all of the files mentioned, plus source code and build files for re-creating them.

3.1 Prepare a Hex File for Burning into Flash

The FlashBurn installation provides examples of bootable DSP programs for TI's development boards. The examples are complete with source code, project files, and command files that are compatible with Code Composer Studio and its utilities. Use the examples to create `.out` files and `.hex` files in Motorola and ASCII formats. You can use these as a starting point for your own programs or data files. The details of how to prepare hex files for your own programs are outside the scope of this paper. You can also save hex data from a DSP memory image using Code Composer Studio's File–>Data–>Save menu and saving data in Hex format.

This example is based on "`FlashBlink62.hex`," one of the example files provided by the FlashBurn installation. The file was originally created from a Code Composer Studio `.out` file using the utility program `hex6x.exe`.

3.2 Prepare the Target System

FlashBurn uses Code Composer Studio to download a program (called the FBTC program) to the target system and run it. For this to work with the TI development boards, the best preparation is to run Code Composer Studio and use the standard gel file initialization provided for the board. This initialization needs to be done whenever the board is first powered–up or is reset. You can leave CCS open after the initialization, but it is not necessary if the correct driver is installed using CCS Setup. FlashBurn does not require any unusual connection nor preparation to be used with a TI development board.

Communications with the target system are fastest when the connection between the Host computer and the target system uses the TI XDS560 emulator and JTAG cable. If the connection is through the parallel HPIB, communications (and thus flash memory programming) run considerably slower.

3.3 Start the FlashBurn Application

The FlashBurn application uses a `.cdd` document to store its information. If there is no target system present, FlashBurn allows you to open, edit, and save documents, but the Program menu is disabled.

You can start FlashBurn, then use its File menu to open or create a new document, or you can start FlashBurn by double–clicking an existing `.cdd` document. In this example, we used the `.cdd` document `C:\ti\bin\utilities\FlashBurn\c6000\dsk6211\Blink62.cdd`.

When FlashBurn opens this document, several things happen:

- FlashBurn creates a connection with Code Composer Studio. If this connection fails, the application will exit.
- The processor type is checked. If the Processor Type specified in the `.cdd` document does not match the processor type of the target system, you'll be warned, and no

downloading or programming will be allowed. However, you may still edit and save the document.

- If the FBTC program is provided (see the **FBTC Program File** field in Figure 1), then the program is downloaded to the the target system and started.
- FlashBurn opens communications with the FBTC program on the target system.

If all the above steps succeed, FlashBurn will display the document for editing, and the Program Menu and related toolbar icons will be enabled.

If you see a dialog stating that the correct target cannot be found, close the FlashBurn application, open the CCS application, then restart FlashBurn while CCS is still open. This may be necessary when the DSK is connected using the parallel port.

3.4 Edit the Document

Only one document may be open at any time. Figure 1 below shows the user's view of the `Blink62.cdd` FlashBurn document. This document is designed for burning an example program into the flash memory of a 6211 DSK, starting at the beginning of the memory.

A `.cdd` document is divided into two main fields.

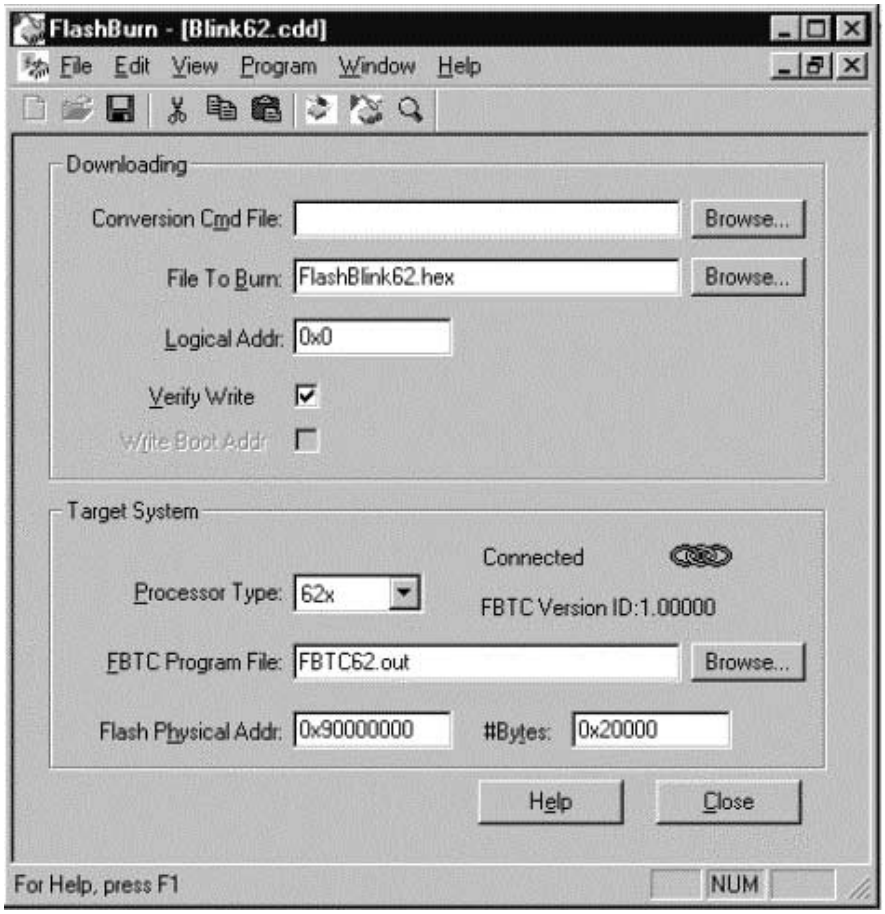

Figure 1. FlashBurn Document

The upper field contains the name of the data file to be burned into the flash memory (`FlashBlink62.hex`) and the logical address at which to start loading the data (0). The **Conversion Cmd File** field will be discussed later.

The lower field contains items that pertain to the target system's hardware. In this example, the processor type is 62x because the document was prepared for a 6211 DSK board. The physical address and size of the on-board flash memory are as shown in Figure 1.

3.5 Erase the Flash Memory

To erase the entire memory, select Program–>Erase Flash (or click the eraser icon on the toolbar). A progress bar will show the progress of the erasure. Some flash memories require several seconds to erase, others require very little time.

Although some flash memory devices allow erasing only a sector or block of the memory, FlashBurn does not provide an interactive means to do this. FlashBurn will only erase the entire flash memory.

3.6 View the Flash Memory

To verify erasure or view the current data in the flash memory, select Program–>Show Memory (or click the magnifying glass icon on the toolbar).

In the Flash Memory Offset dialog, choose the display format and the logical address (the offset from the beginning of the flash memory) to be displayed. See Figure 2.

Figure 2. Show Memory Offset Dialog

You may enter the logical address as a decimal or hex (leading "0x") value.

The memory display is shown in Figure 3. The logical address and memory values are always shown as hex values.

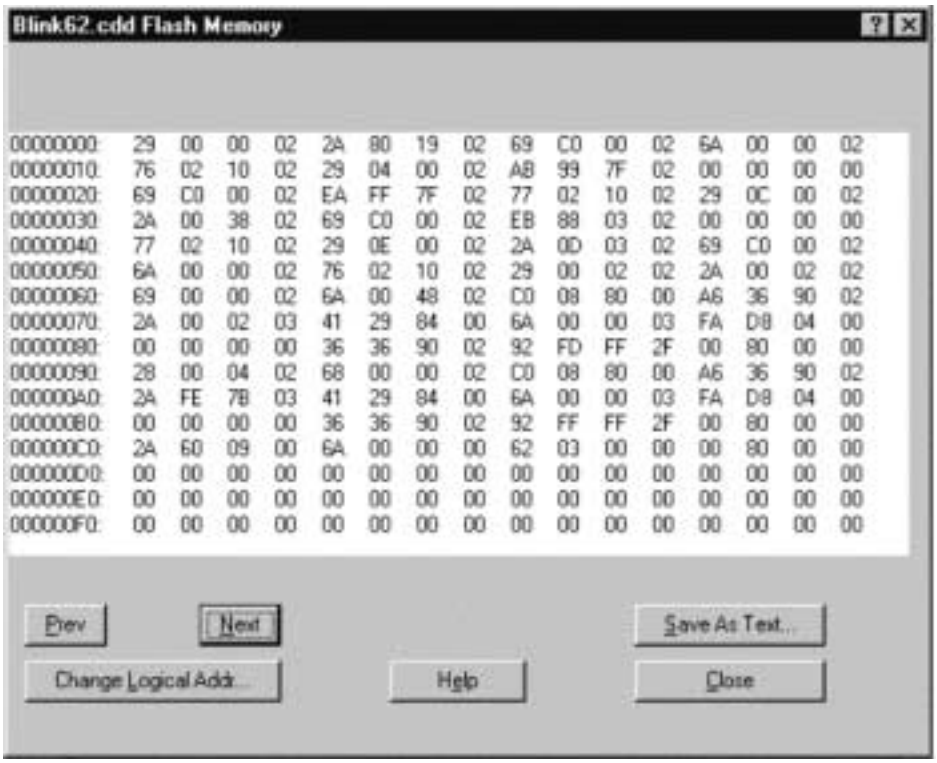

Figure 3. Flash Memory Display

Use the “Prev” and “Next” keys to view the previous or next 256–byte block of data.

3.7 Program the Flash Memory

To program the flash memory with the contents of the hex file, select Program–>Program Flash. A progress bar will be displayed. Typical programming time for the DSK6211 on a 600 MHz desktop PC the typical programming rate is 5K/sec using the XDS560 / JTAG connection. When a HPIB connection is used with the same PC, typical programming rate is 0.5 K/sec. If Write Verify is checked, the rate approximately halves for either connection.

In the example, the contents of the hex file will be written starting at logical address 0, the beginning of the flash memory. To cause the same data to be written at a higher location in the flash memory increase the value of the **Logical Addr.** field in the `.cdd` document.

Before reading the hex file, FlashBurn checks to see if the **Conversion Cmd File** field contains a file. If so, then FlashBurn will run the hex conversion utility appropriate to the processor family, passing it the given command file. FlashBurn waits until the conversion is complete, and then proceeds to read the hex file.

The **Conversion Cmd File** field is especially useful if you run FlashBurn in command line mode using a batch file (see below under “Command Line Mode”).

Note that if the hex conversion fails in any way, FlashBurn will not notify you of any problem. It will attempt to read the hex file, assuming all was OK.

For a specific example showing in detail how a BIOS application was implemented in Flash Memory on a DSK5402, see SPRA773, *Developing a CCS 2.0 DSP/BIOS Application for FLASH Booting on the TMS320C5402 DSK*.

4 The FBTC Program

The FlashBurn application does not have any knowledge of how to erase or program a flash memory. It relies on the **F**lash**B**urn **T**arget **C**omponent (FBTC) program to provide these services for the target system. However, in order to make the user interface more effective, and to assure compatibility, FlashBurn does expect the FBTC program to provide certain details about the target system, and to follow a well-defined communications protocol.

These details are described in the document *FlashBurn Project Exchange Communications Protocol* (provided as part of the installation). The protocol is a Master/Slave implementation, with FlashBurn as the master. As long as an FBTC program follows this protocol, FlashBurn should work as required.

The FlashBurn installation provides example FBTC programs that are compatible with TI's development boards and follow the communications protocol. These FBTC programs are supplied with complete C language source code and build files. If your target system is similar to a TI development board, but differs only in memory map or flash chip type, you may want to customize the example FBTC for your needs. As long as you preserve compatibility with the communications protocol, your FBTC should work with FlashBurn.

If you prefer to create an FBTC program from scratch to suit your needs, it must implement the communications protocol mentioned in the Reference section above.

The FlashBurn installation includes the *FlashBurn Project Exchange Communications Protocol* document and the *FBTC Programmer's Reference Manual* for the example FBTC programs.

5 Logical Address vs. Physical Address

FlashBurn views the flash memory chip as a single device with continuous memory, starting at address 0. This is the logical address of the flash memory.

FlashBurn relies on the FBTC program to provide it with the physical address of the flash memory chip and the size in 8–bit bytes of the flash memory. If you make any hardware change to the target system that changes the physical address or the size of the flash memory (e.g., change from a 0.5M chip to a 1.0M chip of the same type), then you should edit the FBTC program so that it responds with the correct information.

Conversely, before FlashBurn performs any significant flash access, it sends the contents of the **Flash Physical Addr.** and **#Bytes** fields to the FBTC, and the FBTC uses those values for its addressing. Normally, the document and the FBTC would agree on the physical address and the size of the flash memory.

If you happen to be working with target systems that differ only by physical address or size of the flash memory, you can use the same FBTC to program both boards by using the FlashBurn document fields **Flash Physical Addr**. and **#Bytes** to make temporary changes to the physical address and size. Here's how it works:

- When a `.cdd` document is opened, Flashurn requests the flash memory physical address and size from the FBTC.
- Using the values from the FBTC, FlashBurn compares them to the corresponding values in the document.
- If they differ, FlashBurn notifies you of the difference, and asks if you'd like to replace the document's values with those from the FBTC. This notification happens only when the document is first opened.
- Thereafter, FlashBurn will precede erasing, programming, or other flash memory access by sending the contents of these fields to the FBTC. The FBTC will use these values instead of its originally programmed values.

6 Creating Hex Files and the Hex Conversion Cmd File Field

For burning data into the flash memory, FlashBurn expects an input file to be in hex format. You can use Code composer studio to create a `.out` file of your program or data, then use a hex utility provided with CCS to convert the `.out` file to a `.hex` file.

The FlashBurn installation provides examples of simple programs with source code and build files. You can use these to create bootable programs in `.hex` file format, and burn them into the flash memory on TI development boards.

Hex files may be in one of five possible formats:

- Motorola S1/S9 (16–bit addresses) in a `.hex` file.
- Motorola S2/S8 (24–bit addresses) in a `.hex` file.
- Motorola S3/S7 (32–bit addresses) in a `.hex` file.
- ASCII Hex (16–bit addresses) in a `.hex` file.
- Code Composer Studio Saved Data Hex format (saved using CCS File–>Data–>Save menu, and specifying Hex format) in a `.dat` file.

You can use the **Conversion Cmd File** field of the FlashBurn document (`.cdd` file) to specify a command file (of type `.cmd`) that is compatible with the hex conversion utility for the target's processor family. Examples of command files are provided in the installation.

The CCS Saved Data Hex format can be useful if you want to create hex files by hand editing a file to insert your own values, or use Code Composer Studio to edit SRAM memory, then have CCS save the edited locations using the File–>Data–>Save menu. The file format is described in CCS Help (under "Data format"). FlashBurn supports hex data, and ignores the page value.

Here is a brief format description:

Table 2. CCS Saved Data Format

CCS Saved Data File	:==	<header><\n>[<data><\n>…]
header	:==	<magicNumber> <format> <startAddr> <page> <nWords>
magicNumber	:==	magic number: always “1651”
format	:==	Numeric format: FlashBurn only supports “1”, which is hex format.
startAddr	:==	Starting address saved. Ignored by FlashBurn. Use the **Logical Addr** field of a `.cdd` document to specify where the data should be burned into the flash memory.
page	:==	Page number of the data. Ignored by FlashBurn. FlashBurn instead assumes that the data is in Data Page format suitable for the Flash Memory of the current board connected.
nWords	:==	Ignored by FlashBurn. FlashBurn reads until it sees end–of–file.
\n	:==	Newline. Actually, any whitespace (newline, space, tab character) may be used to separate the fields, but CCS writes the header on one line, and each data value one per line.
data	:==	Value in hex format (C–style hex, e.g. 0x12F3C0).

7 Command Line Mode

FlashBurn may be used in Command Line Mode. Its various usages are described here.

```
>FlashBurn
```

This command will simply start the FlashBurn application. The application will run in interactive mode, requiring user input.

```
>FlashBurn foo.cdd
```

This command will start the FlashBurn application and open the document `foo.cdd`. The application will run in interactive mode, requiring user input.

```
>FlashBurn foo.cdd –q [–e] [–n] [filename(s)]
```

This is the most useful way to use command line mode. This command will start FlashBurn and open the document `foo.cdd` in “quiet” mode. In quiet mode, flashburn will use the information in the document and attempt to burn the data in the file specified in the **File to Burn** field.

The square brackets denote optional added control:

- Use the **–e** option to erase the flash memory prior to burning any data. Without the –e option, the memory is not erased. The **–e** option only applies if **–q** is also specified.
- The **–n** option will cause FlashBurn to notify you with a modal dialog box if an error occurs. Otherwise, errors will not be shown on screen. To see if any errors occurred, you will have to look at the log file (called `LogFlashBurn.txt`) that FlashBurn writes when it exits. The **–n** option only applies if **–q** is also specified.

- If you list additional filenames on the command line, FlashBurn will load the `.cdd` document (`foo.cdd` in this example) However, it will ignore the filename in the **File to Burn** field and instead process the file or files listed in order, as follows:
  - Files of type `.hex` and type `.dat` will be burned, as long as they are in one of the supported formats.
  - Files of type `.cmd` will be passed to the appropriate hex conversion utility for processing.
  - Any other files will be simply sent to the target as though you had selected Program –> Send User File to Target (the file's contents are sent to the target, and by default are ignored, unless you have modified the target to do special processing).
  - Note that if the **–e** option is present, erasure is done only once, prior to processing any of the files listed on the remainder of the command line.

Here's a more elaborate example (assume this is for a 6211 DSK):

>FlashBurn foo.cdd –q –e blinkcom.hex blink01.cmd blink01.hex blink02.cmd blink02.hex

1. Opens the `foo.cdd` document.
2. Erases the entire flash memory.
3. Burns the file `blinkcom.hex` into the flash memory.
4. Runs the hex conversion utility with `blink01.cmd` as its input.
5. Burns the file `blink01.hex` into the flash memory.
6. Runs the hex conversion utility with `blink02.cmd` as its input.
7. Burns the file `blink02.hex` into the flash memory.

This example presumes that `blink01.hex` is created by the `hex6x.exe` conversion utility using the `blink01.cmd` file, and `blink02.hex` is created by the `hex6x.exe` hex conversion utility using the `blink02.cmd` file, and there are no address conflicts between any of the three `.hex` files (since the –e switch causes a single erasure before any burning is done).

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265