TMS

series

SIGNUM SYSTEMS CORPORATION

Flasher-C2000 On-Chip Flash Programmer

User Manual

SIGNUM SYSTEMS

DISCLAIMER

Signum Systems makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Also, Signum Systems reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Signum Systems to notify any person or organization of such revision or changes.

WARRANTY

Signum Systems warrants to the original purchaser that this product is free of defects in material and workmanship and performs to applicable published Signum Systems specifications for a period of SIX MONTHS from the date of shipment. If defective, the product must be returned to Signum Systems, prepaid, within the warranty period, and it will be repaired or replaced (at our option) at no charge. Equipment or parts which have been subject to misuse, abuse, alteration, neglect, accident, unauthorized installation or repair are not covered by warranty. This warranty is in lieu of any other warranty expressed or implied. IN NO EVENT SHALL SIGNUM SYSTEMS BE LIABLE FOR CONSEQUENTIAL DAMAGES OF ANY KIND. It is up to the purchaser to determine the reliability and suitability of this product for his particular application.

SIGNUM SYSTEMS
1211 FLYNN RD., UNIT #104
CAMARILLO, CA 93012, U.S.A
PHONE 805 • 383 • 3682
WWW.SIGNUM.COM

Signum Flasher for TMS320F28xx/240x/24x Devices

Signum Flasher-C2000 is a software utility that erases and programs the on-chip flash of the TMS320F28xxx, TMS320F280x, TMS320F281x, F240x and F24x devices using the Signum Systems JTAGjet emulator. It eliminates the need for Code Composer Studio.

Device Revision Support

F28xx Devices

Supported F28xx devices only include the F2810, 2811, and 2812 Devices with Revision C (DeviceID = 0x0003) or later. When a Flasher-C2000 programming command is executed, it reads the Device Identification Register (DEVICEID at 0x883) on your target DSP to determine if the silicon revision is C or later. Flasher-C2000 displays an error message if it detects a non-supported target device.

On newer F28xx devices, the single DeviceID register has been replaced by two registers: PartID and RevID. Flasher-C2000 uses both of them to detect the CPU type and revision.

F240x Devices:

Supported F240x devices include: TMS320LF240x (2402, 2403, 2406, 2407) and TMS320LF240xA (2401A, 2402A, 2403A, 2406A, 2407A).

F24x Devices:

Supported F24x devices include: TMS320F240, TMS320F241 and TMS320F243.

Flasher-C2000 Features

The Flasher-C2000 incorporates the following features:

- Adjusts timing delays for any valid oscillator clock frequency and PLLCR ratio.
- Enables, or disables, the ½ divider of the CPU clock, using the CLKINDIV bit in the PLLSTS register (F280x only). Also supports the ½ and ¼ dividers of the CPU clock in TMS320F2833x devices.
- Selects individual sectors for erasing, blank checking, or saving to a file.
- Enables the following flash operations:
  - Erase, Program, and Verify
  - Erase Only
  - Clear Only (240x/24x devices only)

  - Program, Verify
  - Program Only
  - Verify Only
  - Blank Check
  - Verify Clear (F240x/24x only)
  - Depletion recovery (F28xx/24x only)
  - Save Flash Image
  - Flash Lock and Flash Unlock (28xx/240x only)
  - Programming security password (28xx/240x only)

- Calculates the checksum for Flash, OTP, and Flash + OTP.
- Tests the frequency to confirm proper clock configuration (28xx only).
- Provides tool tips for easier identification of buttons and controls.
- Performs detailed diagnostics in the Operation Log field during flash operations, aiding problem identification and resolution.

Getting Started with Flasher-C2000

Prior to running the program, please make sure that the JTAGjet emulator USB drivers have been installed and that you have the required Flasher-C2000 license. If your emulator was ordered without the Flasher option, see *Appendix D - License Installation*, p. 40.

You can launch Flasher-C2000 either by executing Flash2800Win.exe or Flash2400Win.exe from the Start > All Programs menu, or by clicking the programs' icons on your desktop.

FIGURE 1 shows the On-Chip Flash Programmer window, the Flash2800Win program's interface.

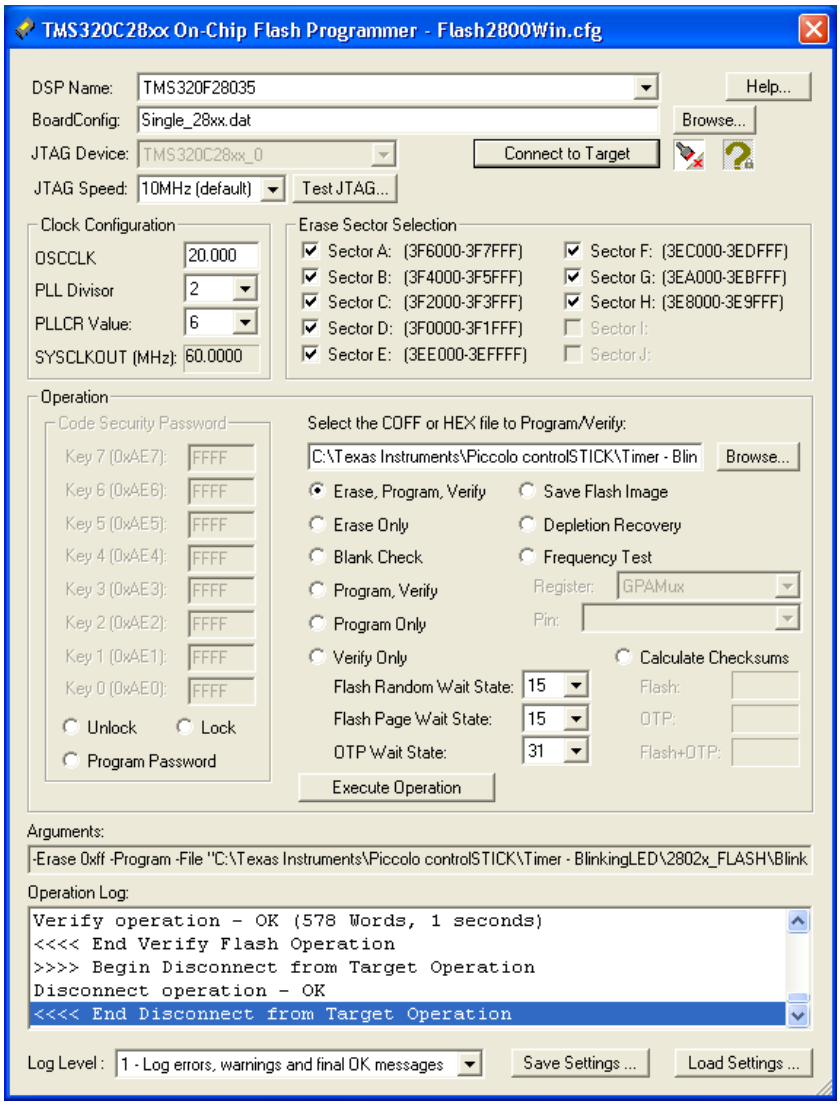

FIGURE 1

The user interface for the Flash2400Win software is shown in FIGURE 2.

As the functionality of Flash2400 is similar to that of Flash2800Win, this manual describes primarily to the Flash2800 operations. Any significant differences concerning the F240x/24x devices are indicated in the text.

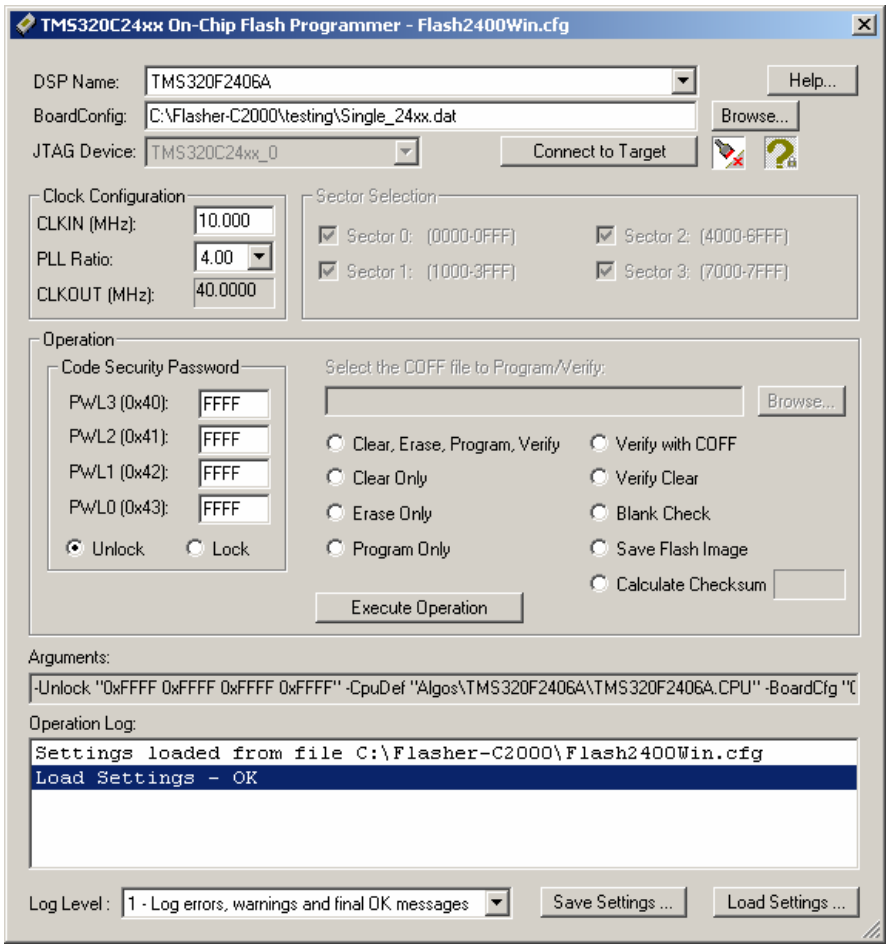

FIGURE 2

Begin by selecting the target DSP device in the DSP Name field. Your selection must be accurate to enable the flash programmer to properly determine the algorithm for programming the on-chip flash. The algorithm files are listed in the .cpu CPU definition file. The file name matches the selected DSP device name. The CPU definition files for all known F28xx, F240x and F24x devices are included with the installation.

If a (new) Texas Instruments device, say TMS320F2808, is not on the drop-down list, you will need to create your own TMS320F2808.cpu CPU definition file, based on the existing definition files. Your file should point to the new flash algorithm .out COFF files. The flash API Interface COFF files are distributed by Texas Instruments free of charge and are compatible with Flasher-C2000. You will find them in the C2000\flashAlgorithms folder of CCStudio 4.x/5.x.

Once the DSP device has been selected, enter the Board Configuration file in the BoardConfig field. This file informs the programmer of the number of devices on the JTAG chain and allows it to choose the device to be programmed. Signum Systems provides several generic configuration files listed in TABLE 1:

FILE	CONFIGURATION
Single_28xx.dat	Boards with a single F28xx device on the JTAG chain.
Single_24xx.dat	Boards with a single F240x/F24x device on the JTAG chain.
Two_28xx.dat	Boards with two F28xx devices.
Two_24xx.dat	Boards with two F240x/F24x devices.
Multi_DSP_example.dat	Example for boards with more complex JTAG chain.

TABLE 1

The selection of the specific device on the JTAG chain for programming can be made in the JTAG Device field. The field remains read-only and grayed-out if the chain contains only one device.

The Connect to Target button

When the DSP Name and Board Configuration are set and the JTAGjet emulator is connected to the target board, click on the Connect to Target button to make sure that the device on the board can be accessed.

There are 5 status icons indicating the state of the connection with the target and flash:


Connection with the target board not established.


Connected to the target board.


The flash status unknown (no connection with the target).


Flash on the target is unlocked (unsecured).


Flash on the target is locked (secured)

If the connection has been successfully established, a Connection to Target OK message will be shown in the Operation Log field and the Connection status icon will turn green. Also, the Flash status icon will change from the question mark to locked or unlocked, depending on the state of the target flash.

If you experience problems connecting to the target, please see *Appendix B – 320F24x with Non-Standard Device ID* (p. 34) for more information.

Flasher-C2000's On-Chip Flash Programmer window can be used without connecting to the target. This can be useful when creating an argument set that can be used with the command line (batch) programmer Flash2000.exe. Simply copy and paste the contents of the Arguments field into a DOS batch file and insert the `Flash2000.exe` command in front of it.

The command line flash programming is explained in detail in *Appendix A – Flasher Command Line Interface.*

> **Note:** For the F240x devices, on-chip flash is accessible only when the DSP is operating in the microcontroller mode (MP/NMC=0). If the flash programmer detects that the DSP is operating in the microprocessor mode (MP/NMC=1), t displays an error message.
>
> On the TMS320LF2407A EVM board, the operating mode can be changed by moving the JP6 jumper from position 1-2 to 2-3. Please refer to the technical reference and schematics of the board for more details.

If the status icon indicates that the flash is locked, the only possible operation is Unlock. Please see section *Unlocking F28xx Device* (p. 14) for more information.

Flasher-C2000 User Interface

Some areas of the programmer's window are disabled under the following conditions:

- **The area is not applicable to a particular device.** For example, as the F2810 devices have only five sectors, A through E, sectors F through J are disabled. Also, since the CLKINDIV bit does not exist on the F281x and TMS320F2833x devices, the CLKINDIV checkbox is available only as an option for the F280x devices.
- **The area is not required for the currently selected flash operation radio button.** For example, if the Program Only flash operation radio button is selected, the Erase Sector Selection controls are disabled, because the Program Only flash operation does not erase flash.
- **A flash operation is in progress.**

When selecting options and operations in the programmer's window, the Arguments field shows the parameters that will be passed to the command line programmer. For production purposes, these parameters may be cut-and-pasted into a DOS batch file to allow automatic execution with a mouse click.

Please note that a batch file operations are allowed only as individual commands (Erase Only, Program Only, etc.). To combine multiple flash operations into a single macro command (batch file), create a batch file with one flash command per line. See *Appendix A – Flasher Command Line Interface* (p. 31) for more information about flash programming using batch files.

When a flash operation starts, a log of the progress appears in the Operation Log field (FIGURE 2). You can control the log detail with the Log Level drop list. Select log level 0 to display only the final outcome of the operations. Log level 6 offers maximum detail, and is recommended only for problems diagnosis performed by Signum Systems engineers.

Flasher-C2000 Settings

When launching, Flasher-C2000 reads the settings in the Flash2800Win.cfg (or Flash2400Win.cfg) files . This file contains a complete programmer configuration and is saved when the programmer exits.

To save the programmer settings to a file with a different name, press the Save Setting button at any time. The button enables you to efficiently create a set of configuration files for different boards. To reconfigure the programmer for another target board for which you previously created a configuration file, press the Load Settings button.

Here is a sample Flash2800Win.cfg file:

```
[Flash2800Win_Settings]
CLKINDIV=1
FlashPage_WS=15
FlashRand_WS=14
FreqTestPin=0
FreqTestReg=0
JTAGDevice=TMS320C2808_0
LogLevel=1
Operation=0
OTP_WS=30
PLLCR=10
SECTOR_MASK=0x3ff
SelDSP=TMS320F2812
BoardCfg=C:\Signum\FlashC2000\Bin\Debug\myboard.dat
OSCCLK=20.000
Key0=FFFF
Key1=FFFF
Key2=FFFF
Key3=FFFF
Key4=FFFF
Key5=FFFF
Key6=FFFF
Key7=FFFF
COFF=C:\Signum\FlashC2000\Bin\Debug\modem.out
```

F28xx Target Clock Configuration

Typically, the Clock Configuration segment of the On-Chip Flash Programmer is similar to the examples in FIGURE 3.

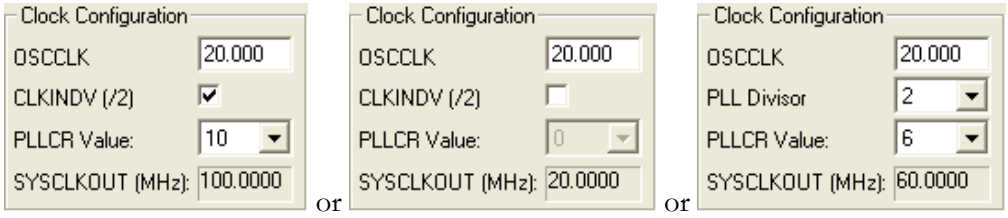

FIGURE 3

OSCCLK is the oscillator clock signal derived either from the on-chip oscillator or from the external oscillator, depending on the hard-wired clock configuration. CLKIN is the input clock to the CPU. Its value is derived from OSCCLK and PLL Block. SYSCLKOUT is the CPU output clock. The frequency of SYSCLKOUT is the same as CLKIN.

The PLL Block allows multiplying and dividing the OSCCLK frequency. The PLL multiplier and divisor are software selectable. The general formula of the PLL Block operation is

```
SYSCLKOUT = CLKIN = (OSCCLK * m) / n
```

where m is the value of the PLL multiplier and n is the value of the PLL divisor.

Information about the PLL operation is displayed in the tool tip for the PLLCR Value drop down list control.

The PLL multiplier m is determined by bits 3:0 (DIV) of the PLLCR register. The valid values are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. If PLLCR[3:0] is 0, the PLL is bypassed and

```
SYSCLKOUT = CLKIN = OSCCLK / n
```

Otherwise PLLCR[3:0] is the same as the PLL multiplier m.

The PLL divisor n for the TMS320F280x and TMS320F280xx devices is determined by bit 1 (CLKINDIV) of the PLLSTS register. CLKINDIV of 0 halves the frequency (n=2), while 1 disables the divisor (n=1).

In TMS320F281x devices, the PLL divisor n is set to 2 (n=2) and cannot be changed.

When the PLL divisor in TMS320F280x/280xx/281x is disabled (n=1), PLLCR is set to 0 and cannot be modified. In this case, SYSCLKOUT = CLKIN = OSCCLK.

The PLL divisor n in TMS320F2833x devices is determined by bits 8:7 (DIVSEL) of the PLLSTS register. (CLKINDIV is not implemented). DIVSEL values 0 and 1 divide the frequency by 4 (n=4), DIVSEL of 2 divides by 2 (n=2), while DIVSEL of 3 disables the divisor (n=1).

The default CLKIN values correspond to the factory settings of the eZdsp target boards. TABLE 2 summarizes the default settings.

DEVICE	OS-CLK (MHZ)	PLL DIVISOR	PLLCR VALUE	SYSCLK-OUT (MHZ)	SYSCLKOUT RANGE (MHZ)
F2801	20.000	/2, /1	10	100.00	15-100
F2801-60	20.000	/2, /1	6	60.00	15-60
F28015	20.000	/2, /1	6	60.00	15-60
F28016	20.000	/2, /1	6	60.00	15-60
F2802	20.000	/2, /1	10	100.00	15-100
F2802-60	20.000	/2, /1	6	60.00	15-60
F28044	20.000	/2, /1	10	100.00	15-100
F2806	20.000	/2, /1	10	100.00	15-100
F2808	20.000	/2, /1	10	100.00	15-100
F2809	20.000	/2, /1	10	100.00	15-100
F2810	30.000	/2	10	150.00	15-150
F2811	30.000	/2	10	150.00	15-150
F2812	30.000	/2	10	150.00	15-150
F28332	30.000	/2, /4, /1	10	150.00	15-150

DEVICE	OS-CLK (MHZ)	PLL DIVISOR	PLLCR VALUE	SYSCLK-OUT (MHZ)	SYSCLKOUT RANGE (MHZ)
F28334	30.000	/2, /4, /1	10	150.00	15-150
F28335	30.000	/2, /4, /1	10	150.00	15-150

TABLE 2

Flasher-C2000 modifies the global variable used by the Flash API Library every time a flash operation is performed.

Flasher-C2000 cannot verify that the OSCCLK value you enter into the On-Chip Flash Programmer window matches your target board's actual OSCCLK. If you specify an OSCCLK value that does not match your target board settings, the timing delays used by the Flash API Library will not be correct and the on-chip flash may not be programmed correctly.

F240x/F24x Target Clock Configuration

The Clock Configuration for F240x window is shown in FIGURE 4.

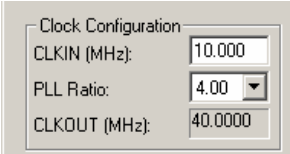

FIGURE 4

The input clock (CLKIN) and the PLL ratio are multiplied together to determine the output clock (CLKOUT). The default values of CLKIN correspond to the factory settings of the eZdsp and EVM target boards. TABLE 3 summarizes the default settings.

DEVICE	CLKIN (MHZ)	PLL RATIO	CLKOUT (MHZ)	CLKOUT RANGE (MHZ)
F2407	7.500	4.00	30.00	15-30
F2401A	10.000	4.00	40.00	15-40
F2407A	10.000	4.00	40.00	15-40

TABLE 3

To program the flash on TMS320LF240x devices, CLKOUT must be between 15-30 MHz, whereas TMS320LF240xA devices require CLKOUT between 15-40 MHz. The flash programmer automatically adjusts the flash algorithms if any of these default values changes when you press the Execute button or the Lock button. However, the programmer does not verify that CLKIN or the PLL ratio you set matches that on your target board. If you specify a CLKIN value or PLL ratio that does not match your target board settings, the timing delays will be incorrect, which may

Note: TMS320F24x devices have a fixed PLL ratio: 2x for the F240 and 4x for the F241 and F243. The Input clock (CLKIN) frequency should be 10.0 MHz for these devices. Flash programming algorithms are tuned for the output clock (CLKOUT) frequency of 20.0 MHz for the F240, and 40.0 MHz for the F241 and F243. These algorithms cannot be adjusted for CLKIN frequencies other than specified above.

result in a faulty programming of the on-chip flash.

F28xx Code Security Password

The eight 16-bit passwords (Key 0 – Key 7) form the 128-bit password that the Code Security Module (CSM) uses to lock the device. Key 7 is located at address 0xAE7 in the program space and is the most-significant word. Key 0 is located at address 0xAE0 in the program space and is the least-significant word.

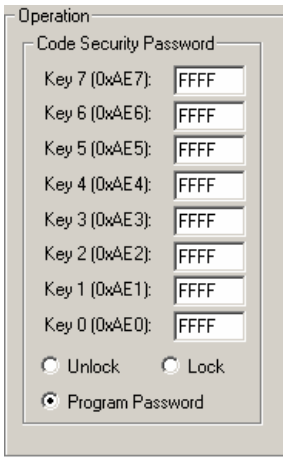

FIGURE 5

The security password is stored at memory locations PWL0 - PWL7 (0x3F7FF8 - 0x3F7FFF on TMS320F28xx devices, and 0x33FFF8 – 0x33FFFF on TMS320F2833x devices). New devices are unsecured. On unsecured devices, all password locations are set to one (all 0xFFFF's). While it is possible to read and program flash memory on an unsecured device, it is impossible to lock such a device. To secure or lock a device, set the security password to a value that is different from the password on unsecured devices (0xFFFF). See Programming Passwords (p. 11) to learn more about programming secure and unsecured passwords.

Once a security password has been programmed, the device locks on the next power-cycle (power-down, power-up). A secured device that is locked disallows flash memory read and write operations. To read or write flash memory, unlock the device.

> **Caution:** If you lose the security password, you will not be able to view flash memory or reprogram it using a new application code or a new password.

F240x Code Security Password

The Code Security feature is available only on TMS320LF240xA devices. When working with TMS320LF240x/24x devices, Flash-C2000's user interface grays out the security password related fields.

Four 16-bit locations (PWL3-PWL0) form a 64-bit password. PWL3 is located at address 0x40 in the program space and is the most-significant word. PWL0 is located at address 0x43 in the program space and is the least-significant word.

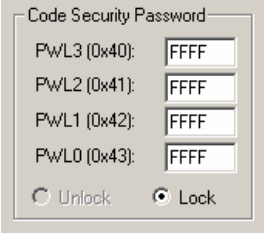

FIGURE 6

By default, a new device is unsecured; its flash memory can be read and written. On an unsecured device, the password values are either all zeros or all ones.

The device becomes secure on the first power-cycle (power-down, power-up) after the security password has been set. A secured device does not allow you to read or write flash memory until it is unlocked with the password with which it is locked.

> **Caution:** If you lose the security password, you will not be able to view flash memory or reprogram it with a new application code or a new password.

Programming Passwords

The Code Security Module (CSM) password can be programmed in two ways:

- As part of regular flash programming,

— or —

- Using the Flasher-C2000 Program Password operation.

Regular Flash Programming

To program the password as part of regular flash programming, use the linker command file for your F28xx application for placing the password values into flash memory at the password locations PWL0 – PWL7 (0x3F7FF8 – 0x3F7FFF on TMS320F28xx devices and 0x33FFF8 – 0x33FFFF on TMS320F2833x devices).

On F240x devices, the PWL3 – PWL0 keys are placed at locations 0x40 – 0x43.

Program Password

To program the password using Flasher-C2000's Program Password operation, follow this procedure.

1. Ensure that your application's linker command file does not overwrite the flash memory locations reserved for the security password, or makes them all 0xFFFF.

2. Erase flash memory, making sure that all sectors are selected.
3. Program your target application COFF or HEX file to the flash.
4. In the On-Chip Flash Programmer window, enter the password into the Code Security Password fields,

check the Program Password checkbox for F28xx devices (FIGURE 5),

— *or* —

select Lock for F240xA device (FIGURE 6),

and click the Execute Operation button.

It is also possible to reverse the order of the last two steps above, that is, to program the password before programming the flash with your application.

When executing the Program Password operation, Flasher-C2000 reads the password locations, verifying that they are cleared (all 0xFFFF). If they are not, the programmer displays a warning (FIGURE 7).

FIGURE 7

Caution: Setting a password to all zeros permanently locks the F28xx devices.

The warning dialog defaults to the safer of the two alternatives—canceling the Program Password operation. Clicking the Yes button results in programming the new password to flash memory over the existing password. If the action is unintended, it is fraught with the possibility of losing the ability to unlock the device.

This is because for CSM code security operation, all flash memory addresses between 0x3F7F80 - 0x3F7FF5 on TMS320F28xx devices (0x33FF80 – 0x33FFF5 on TMS320F2833x devices) cannot be used as program code or data, but must be programmed to 0x0000 when the code security password is programmed. The 128-bit password (at 0x3F 7FF8 – 0x3F 7FFF) *must not* be programmed to zeros. Doing so would permanently lock the device. (More information can be found in your device's data sheet.)

Before it programs the password, Flasher-C2000 reads the password locations. If it finds out that they are not all zeros, the programmer seeks your permission to set them to all 0x0000's (FIGURE 8).

FIGURE 8

When the Program Password button is pressed, Flasher-C2000 verifies that fields Key 0 through Key 7 are not all zero. If the keys are all zero, the Program Password operation is aborted and the following message appears in the Operation Log field of the On-Chip Flash Programmer window:

```
Setting the password to all zeros will permanently lock the device.
This is not recommended. Program Password operation was cancelled.
```

If the password keys are not all zeros, Flasher-C2000 programs the values in fields Key 0 through Key 7 into flash memory at the password locations.

> **Caution:** You must never halt, restart, or reset the CPU while the program password flash operation is in progress. Doing so may damage the flash memory.

Locking F28xx Devices

To lock a secure F28xx device, select Lock and press the Execute Operation button (FIGURE 9).

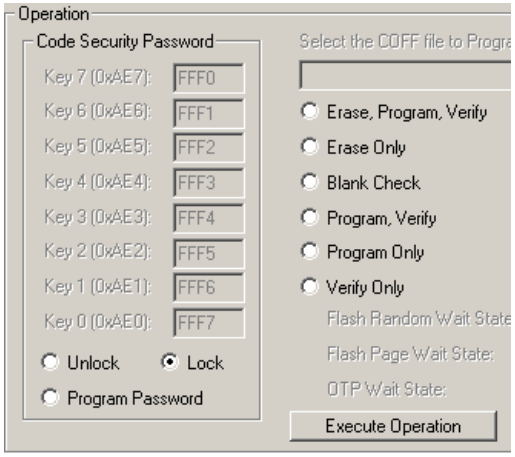

FIGURE 9

Locking a device does not require a password. Flasher-C2000 locks a secure device by writing to the Code Security Module Status and Control Register (CSMSCR) at address 0xAEF and setting the FORCESEC bit (bit 15).

When a secure device is locked, a yellow padlock icon is displayed in the On-Chip Flash Programmer window in the locked ("secured") position, and all flash memory locations reads as zeros. From that time on, access to debug the contents of secure memory by any means requires a valid password. The code, however, can run out of secure memory (a typical end-customer usage).

Locking F240x Devices

For information on locking the F240x devices, please refer to section *F240x Code Security Password* (p. 10).

Unlocking F28xx Devices

To unlock a secure F28xx device, enter the 128-bit password into the Key 0 – Key 7 fields and

select Unlock. Press the Execute Operation button (FIGURE 10). One of the following takes place:

- If the password is correct, the secure device is unlocked, the Unlock operation is disabled, (grayed-out) and the Lock operation is enabled.
- If the password is incorrect, the Unlock operation remains enabled and the Lock operation is, or remains, disabled (grayed-out).

When a secure device is locked, a yellow padlock icon is displayed in the locked ("secured") position.

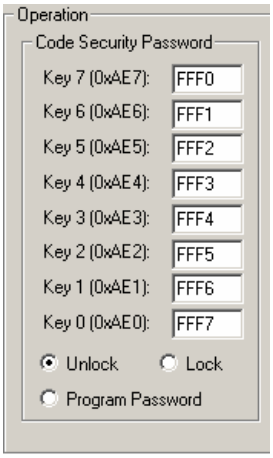

FIGURE 10

Unlocking F240x Devices

All flash memory locations of a secured F240x device read as random data values when debugging is attempted. To unlock a secured F240x device, enter the 64-bit password into the PWL3-PWL0 fields, select the Unlock button, and press the Execute Operation button.

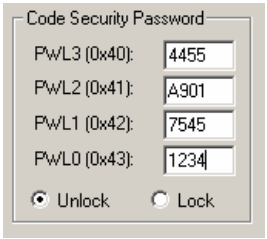

FIGURE 11

Flash Operations

The Operation area contains nine possible flash operation radio buttons and entry fields for the required parameters. When you choose a certain flash operation, the Flasher-C2000 may disable the parameters that are not required for the selected operation.

For example, the Calculate Checksums flash operation does not require a Sector selection, so the Erase Sector Selection area is disabled. The help topic for each flash operation explains which parameters are required for that operation.

- For information on how to specify sectors, refer to section *Erase Sector Selection,* p. 27.
- For information on how to specify a COFF or HEX file, refer to section *Specifying the COFF or HEX file,* p. 28.
- For information on how to specify the Verify Operation parameters, refer to section *Specifying the Flash Operation Wait States (F28xx only),* p. 28.

The Execute Operation button starts the flash operation using the current settings.

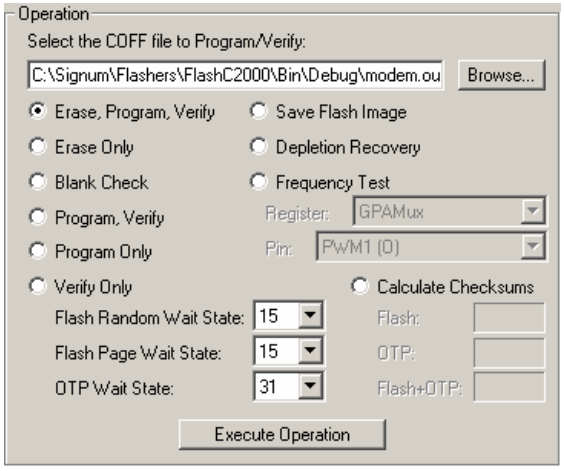

FIGURE 12

When the selected operation is under way, the Operation Log field displays information about the progress. You can control the amount log detail by choosing the appropriate level from the Log Level drop-down list.

The F240x flash operations do not include the Depletion Recovery operation, Frequency Test operation, and Wait State entry fields.

Erase, Program, Verify Operation

The Erase, Program, Verify flash operation performs the following actions:

- Erase the sectors selected in the Erase Sector Selection area. The F240x devices require an additional Clear operation.
- Program the COFF or HEX file specified in the Operation area into flash memory.

- Verify flash memory against the COFF or HEX file specified in the Operation area.

The required parameters for the Erase, Program, Verify operation are Sector Selection, a COFF or HEX file, and the Verify Operation wait state parameters (F28xx devices only).

- For information on how to specify sectors, refer to section *Erase Sector Selection,* p. 27.
- For information on how to specify a COFF or HEX file, refer to section *Specifying the COFF or HEX file,* p. 28.
- For information on how to specify the Verify Operation parameters, refer to section *Specifying the Flash Operation Wait States (F28xx only),* p. 28.

Typically, you would select all sectors for the Erase, Program, Verify flash operation. However, sector selection is enabled as you may have previously erased some, but not all sectors, using the Erase Only flash operation.

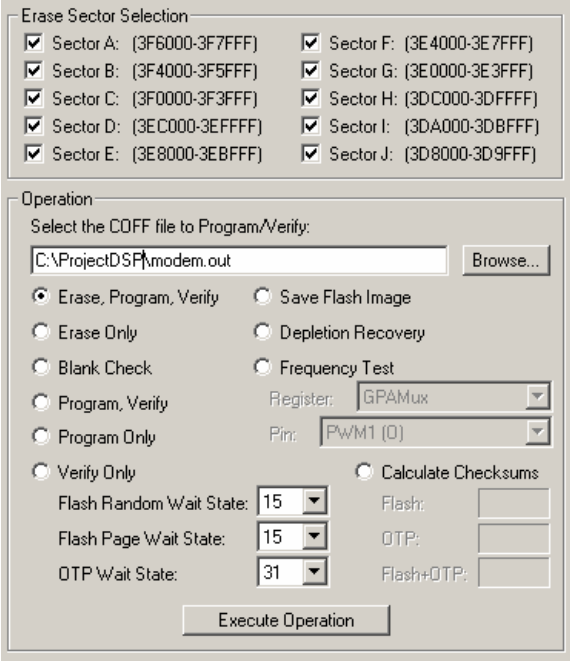

FIGURE 13

Once you have specified all the required parameters, press the Execute Operation button to start the Erase, Program, Verify flash operation.

> **Caution:** You must never halt, restart, or reset the CPU while the Erase, Program, Verify flash operation is in progress. Doing so may damage the flash memory.

Clear Only Operation (F240x/F24x only)

The Clear Only flash operation is only available on the F240x and F24x devices. It must be performed before each Erase operation.

This operation sets all bits in the sectors selected in the Clear Sector Selection area to zero. Sector Selection is the only required parameter for the Clear Only flash operation. For information on how to specify sectors, refer to section *Erase Sector Selection,* p. 27.

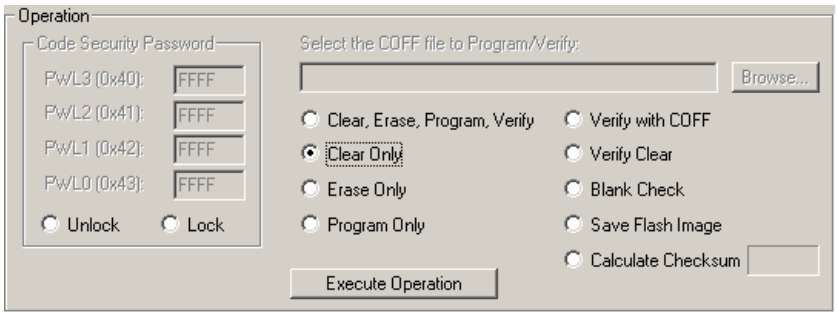

FIGURE 14

Once you have specified which sectors are to be cleared, press the Execute Operation button to start the Clear Only flash operation.

> **Caution:** You must never halt, restart, or reset the CPU while the Clear Only flash operation is in progress. Doing so may damage the flash memory.

Erase Only Operation

The Erase Only flash operation erases the sectors selected in the Erase Sector Selection area.

The only required parameter for the Erase Only flash operation is a Sector Selection (FIGURE 15). For information on how to specify sectors, refer to section *Erase Sector Selection,* p. 27.

Once you have specified which sectors you wish to erase, press the Execute Operation button to start the Erase Only flash operation.

On the F240x and F24x devices, the Erase operation should only be used on sectors that have been cleared. If you are unsure if a sector is cleared, select the Verify Clear operation, then press the Execute Operation button and the programmer will tell you. Alternatively, execute the Clear Only operation followed by the Erase Only operation. Erasing a sector causes all its bits to be set to one.

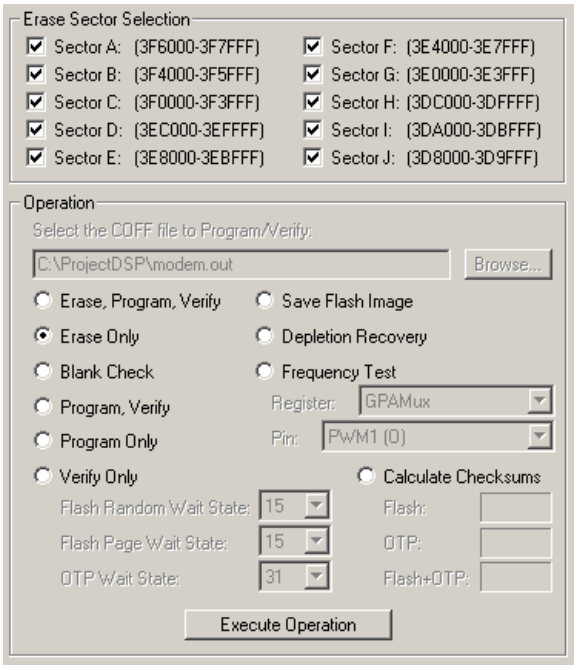

FIGURE 15

> **Caution:** You must never halt, restart, or reset the CPU while the Erase Only flash operation is in progress. Doing so may damage the flash memory.

Blank Check Operation

The Blank Check flash operation reads the entire flash device to determine if it is erased (FIGURE 16). The sectors selected in the Erase Sector Selection area are ignored. The result of the blank check operation is reported in the Operation Log area of the On-Chip Flash Programmer window.

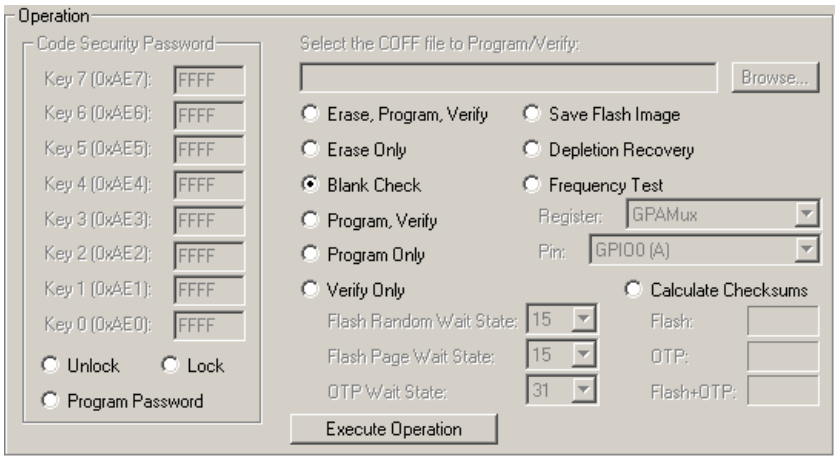

FIGURE 16

Program, Verify Operation

The Program, Verify flash operation performs the following actions:

- Program the COFF or HEX file specified in the Operation area into flash memory.
- Verify flash memory against the COFF or HEX file specified in the Operation area.

The required parameters for the Program, Verify flash operation are the COFF or HEX file and the Verify operation Wait State parameters.

- For information on how to specify a COFF or HEX file, refer to section *Specifying the COFF or HEX file,* p. 28.
- For information on how to specify the Verify Operation parameters, refer to section *Specifying the Flash Operation Wait States (F28xx only),* p. 28

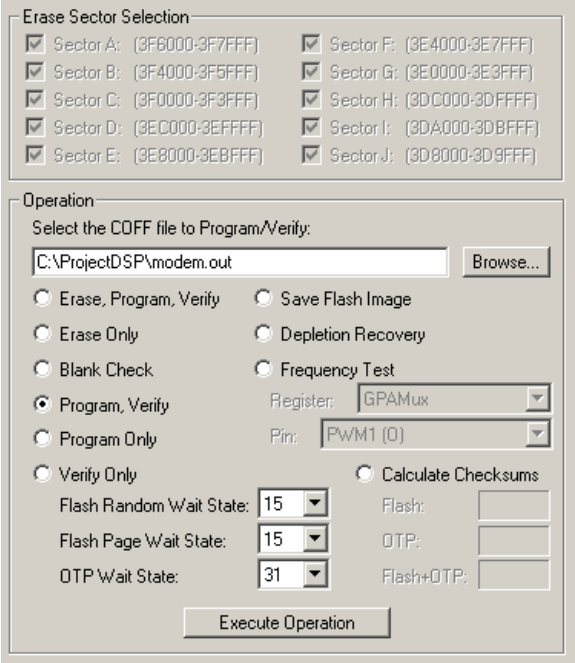

FIGURE 17

Once you have specified all the required parameters, press the Execute Operation button to start the Program, Verify flash operation.

> **Caution:** You must never halt, restart, or reset the CPU while the Program, Verify flash operation is in progress. Doing so may damage flash memory.

Program Only Operation

The Program Only operation should be used only on sectors that have previously been erased. If you do not know if the sector is erased, select the Erase Only operation and press the Verify button and the programmer will tell you. Alternatively, execute the Erase Only operation followed by the Program Only operation.

The Program Only flash operation programs the specified COFF or HEX file into the flash memory. The required parameter for the Program Only operation is a valid COFF or HEX file.

- For information on how to specify a COFF or HEX file, refer to section *Specifying the COFF or HEX file,* p. 28.

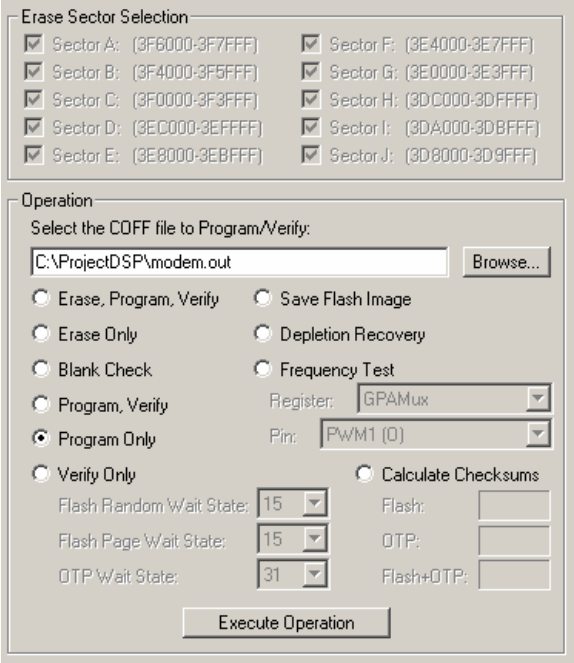

FIGURE 18

Once you have specified the COFF or HEX file, press the Execute Operation button to start the Program Only flash operation.

> **Caution:** You must never halt, restart, or reset the CPU while the Program Only flash operation is in progress. Doing so may damage the flash memory.

Verify Only Operation

The Verify Only flash operation verifies flash memory against the COFF or HEX file specified in the Operation area.

The required parameters for the Verify Only flash operation are a COFF or HEX file and the Verify operation Wait State parameters (FIGURE 19).

- For information on how to specify a COFF or HEX file, refer to section *Specifying the COFF or HEX file,* p. 28.
- For information on how to specify the Verify Operation parameters, refer to section *Specifying the Flash Operation Wait States (F28xx only),* p. 28

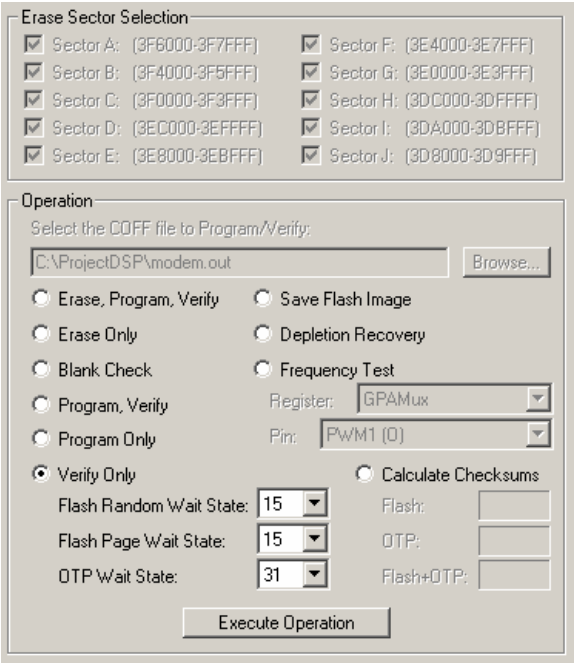

FIGURE 19

Once you have specified all the required parameters, press the Execute Operation button to start the Verify Only flash operation.

In the F240x/F24x version of Flasher_C2000, the Verify Only operation is replaced by the Verify with File operation. (This version's Verify Clear operation verifies if all bits in the selected sectors were cleared, that is, set to zero.)

Save Flash Image Operation

The Save Flash Image operation saves the contents of the flash memory into a HEX or binary file. The required parameters for this operation are a HEX/BIN file name, and the Sector Selection parameters (FIGURE 20).

- For information on how to specify sectors, refer to section *Erase Sector Selection,* p. 27.

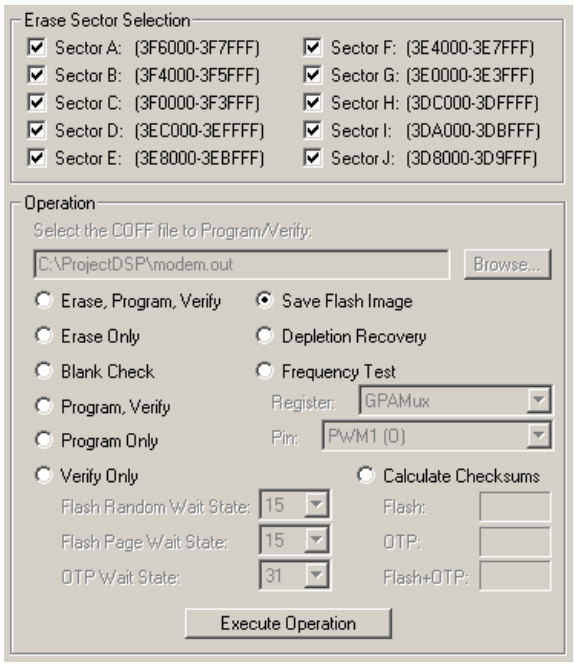

FIGURE 20

Once you have specified all the required parameters, press the Execute Operation button to start the Save Flash Image operation. You will be asked to browse for the file to be saved. It is possible to save the flash image in either binary or hex format. *Appendix F – Using HEX Files* (p. 43) provides additional information about the hex file formats supported. Saved HEX files can be used when programming or verifying the flash.

Depletion Recovery Operation (F28xx and F24x only)

The depletion recovery algorithm searches all sectors on the device for those that are in depletion and attempts recovery. This operation is available only on the F28xx and old F24x devices.

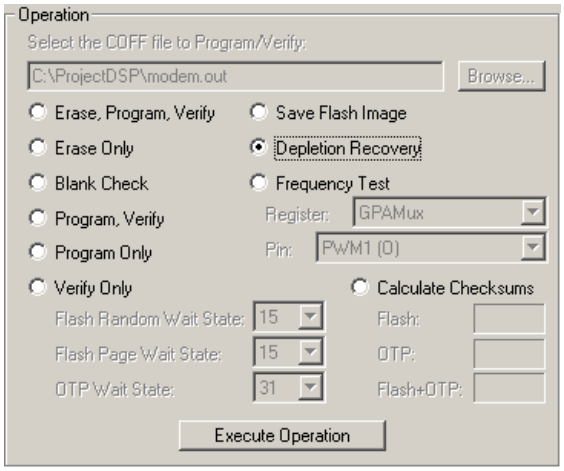

FIGURE 21

How Does Depletion Occur?

Depletion may occur when the Erase operation is interrupted and not allowed to complete. It is common for a device that become depleted to fail to erase afterwards, have its CSM password corrupted, or both. Therefore, the erase algorithm should be allowed to terminate of its own accord.

If the passwords are in an unknown state, the depleted device cannot be recovered. If, however, the CSM passwords are known and the device can be unlocked, then the depletion recovery algorithm stands a chance to recover the depleted part of the flash.

The current maximum timeout for the algorithm is approx 35 seconds per sector in depletion. Unless Erase has been performed multiple times on multiple sectors and interrupted, typically, only one sector is in depletion. If a longer timeout can be tolerated, depletion recovery can be used multiple times.

There is no guarantee that the Depletion Recovery operation will be able to bring a sector out of depletion within a reasonable amount of time. The deeper in depletion the part is, the longer its recovery. The Flash API Erase function has been implemented to erase the flash in a way that avoids deep depletion. However, if the CPU is halted during an erase pulse for a long period of time, the part may suffer a deep depletion that may not be recoverable in an acceptable time frame.

This algorithm cannot recover the part if the flash passwords are unknown. For example, if power is lost during the erasure of sector A where the CSM passwords are located, then the device may be permanently locked and inaccessible to the recovery algorithm.

Frequency Test Operation (F28xx only)

This operation is only available on the F28xx devices.

The Frequency Test flash operation toggles the specified GPIO Register pin to confirm proper clock configuration. The Frequency Test flash operation requires that you monitor the specified GPIO Register pin with an oscilloscope. Complete instructions for the Frequency Test can be found in the TMS320F2808, TMS320F2806, and TMS320F2801 Flash API (SPRC193) and the TMS320F2810, TMS320F2810, and TMS320F2812 Flash API (SPRC125) documentation.

> **Caution:** It is strongly recommended that you run the Frequency Test before erasing or programming every new batch of parts or boards. If this test fails (usually due to a crystal frequency mismatch), do not proceed to erase or program the flash until the problem is corrected, or flash damage may occur.
>
> The flash programming process includes delay loops. If the CPU runs at a speed that does not match these loops, the flash may not be erased or programmed thoroughly.

One of the required parameters for the Frequency Test flash operation is the GPIO Register pin toggle (FIGURE 22).

To specify the GPIO Register pin for F281x devices, begin by selecting the GPIO Register from the Register drop down list. Then from the Pin drop down list, select the GPIO Register pin to be toggled.

To specify a GPIO Register pin for other or newer F28xx devices, select the GPIO Register pin to be toggled from the Pin drop down list. (The GPIO register list will not be available.)

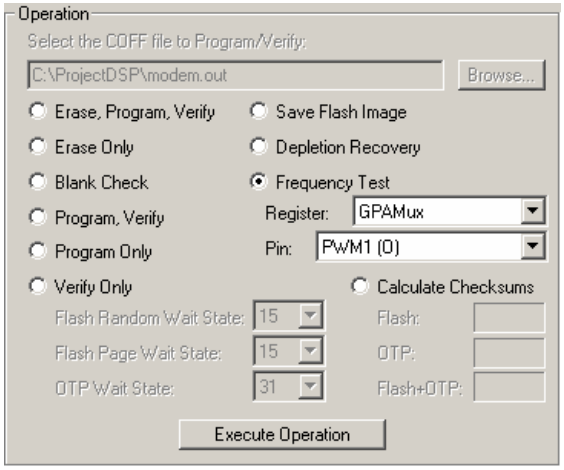

FIGURE 22

Once you have specified a Register and Pin, press the Execute Operation button to start the Frequency Test flash operation. The Execute Operation button title changes to Halt Frequency Test.

The Frequency Test flash operation causes the target execute an endless loop, allowing you to measure the frequency on the specified GPIO Register pin with an oscilloscope. The measured frequency should be 10KHz. If it is not, verify that your clock configuration is correct.

To halt Frequency Test, press the Halt Frequency Test button.

Calculate Checksums Operation

The Calculate Checksums flash operation (FIGURE 23) calculates checksums for the following types of memory:

- Flash memory.
- One Time Programmable (OTP) memory (F28xx only).
- Sum of Flash memory and OTP memory (F28xx only).

The flash memory and OTP memory checksums are calculated on the target by adding together all 16-bit memory locations in the flash or in the OTP range. The checksum for the sum (combination) of the flash and the OTP memory is calculated on the host.

There are no required parameters for the Calculate Checksums flash operation.

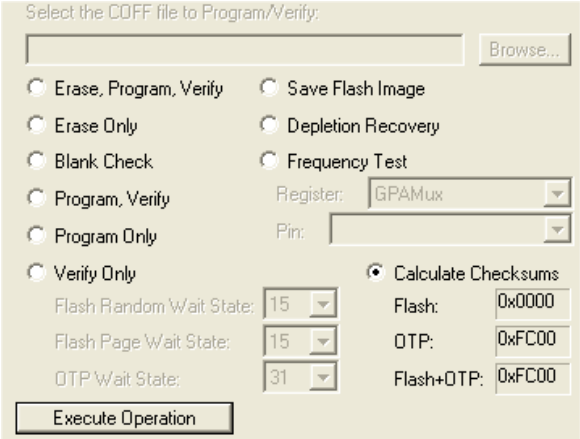

FIGURE 23

To compute the checksums, select Calculate Checksums and press the Execute Operation button.

When the Calculate Checksums flash operation completes, the results are displayed below the Calculate Checksums radio button.

The checksum is calculated as a 16-bit sum of all the values in the flash area (the main flash or OTP). For example, when calculating the checksum on an empty F28035 device (all flash values 0xFFFF), the checksum will be as follows (TABLE 4):

	CHECKSUM	WHY
Flash	0x0000	256K × 0xFFFF = 0x3FFF**C0000** (only 0x0000 used)
OTP	0xFC00	0x400 × 0xFFFF = 0x3FF**FC00** (only 0xFC00 used)

TABLE 4

Caution: The checksum calculations in the Code Composer Studio plugin may be incorrect. This is because some of the flash programming algorithms supplied with CCStudio calculate the checksum outside of the user-accessible flash area. Since the flash includes the ADC calibration data (in the part of the flash that is not accessible to the end-user), the plugin may generate checksums for the same code that vary from device to device. Flasher-C2000 properly calculates the checksum in the user-accessible flash range.

Verify Checksums Operation

The Verify Checksums flash operation is available only from the command line interface with the use of the Flash2000.exe application (see *Appendix A – Flasher Command Line Interface*, p. 31). The operation calculates the checksums for the sum of the flash memory and One Time Programmable (OTP) memory (F28xx only), and then compares them against reference values generated by the -VerifyChecksum command. The operation is deemed successful when the calculated checksums match their corresponding reference values, and failed otherwise.

The syntax of the –VerifyChecksum values is explained in TABLE 5 and TABLE 6.

SYNTAX	COMMENTS
-VerifyChecksum "HHHH HHHH"	Compute both the main flash and OTP checksums
-VerifyChecksum HHHH,HHHH	Same as above (a newer format without double quotes)
-VerifyChecksum HHHH	Compute the main flash checksum only.
-VerifyChecksum ,HHHH	Compute the OTP checksum only (starts with a comma)

TABLE 5

Examples:

EXAMPLE	COMMENTS
-VerifyChecksum 1234,ab56	Verify the main flash and OTP checksums
-VerifyChecksum 1234	Verify the main flash checksum.
-VerifyChecksum ,ab56	Verify the OTP checksum.

TABLE 6

Execute CPU Control Command Script File Operation

The operation Execute CPU Control Command Script File is only available from the command line provided by the Flash2000.exe application (see *Appendix A – Flasher Command Line Interface*, p. 31). This operation allows you to execute a series of low level CPU control commands from within a command script file specified with the –Exec option, as shown in the following example.

```
-Exec Cmds.txt -CpuDef "Algos\TMS320F2406A\TMS320F2406A.CPU" -BoardCfg
"Single_24xx.dat" -JtagDev "0" -PLL 10.000x2.00 -Log 2 -ReadBack algo
```

TABLE 7 list the currently available CPU control commands.

COMMAND	DESCRIPTION
RESET	Resets the target DSP
GO	Executes a program on the target device. Same as RUN.
STOP	Stops execution of a program on the target device.
PAUSE <nsec>	Waits for the specified number of seconds <nsec>. Same as WAIT or SLEEP or DELAY.
SD <addr> = <val>	Write the specified 32-bit double word value <val> at address <addr> in the target memory.
SW <addr> = <val>	Write the specified 16-bit word value <val> at address <addr> in the target memory.

TABLE 7

The command set in TABLE 7 can be extended to accommodate the specific needs of the end user requirements.

The command script file with CPU control commands is a plain text file. When editing it, remember that a line of the command script file must contain only one CPU control command. Empty lines are skipped. Lines that begin with a semicolon character are ignored as comments.

Linker Auto-initialization Options and Flash Memory

A variable can end up in Flash Memory in either of these situations;:

1. The default compiler options places the variable in a section that is mapped to Flash Memory in the linker command file.
2. You explicitly place the variable into a section that is mapped to Flash Memory by using the compiler's DATA pragma.

If you use the -c linker option to automatically initialize variables at run time, which is the default, then the run-time library attempts to write to Flash Memory and silently fails. Always make sure that data sections in your program are located in RAM areas.

If you use the -cr linker option to automatically initialize variables at the load time, Code Composer Studio initializes the variable properly when it loads the COFF file. However, such a program will not run from the flash after a power-up, as there is no debugger present to load data sections to the RAM.

Erase Sector Selection

The Erase Sector Selection area of the On-Chip Flash Programmer window allows individual flash sectors to be selected for the Erase, Program, Verify, Erase Only, and Save Flash Image operations. Erasing a sector sets all its bits to one. To program the flash correctly and reliably, it is imperative to erase the sectors before any application data is programmed into them.

Note that if none of the Erase, Program, Verify, Erase Only or Save Flash Image operation radio buttons is selected, the entire Erase Sector Selection area is disabled.

The sector memory range is displayed next to sector name. Sectors that are not supported by a particular DSP are disabled. For instance, as the TMS320F2808 device does not have Sectors E - J, the Erase Sector Selection looks like this (FIGURE 24):

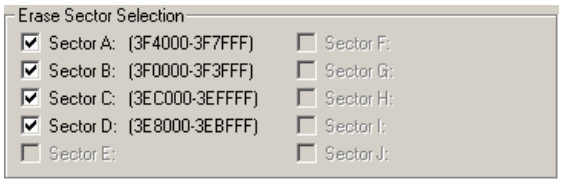

FIGURE 24

To erase one or more sectors with the use of the Erase, Program, Verify, or Erase Only flash operation, check the sector checkboxes corresponding to those sectors.

On the F240x devices, the Clear/Erase Sector Selection area is limited to 4 sectors only (FIGURE 25).

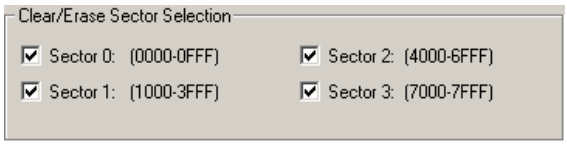

FIGURE 25

As the flash on the F24x devices have one flash sector, the Clear/Erase Sector Selection area also has one checkbox only.

Specifying the Flash Operation Wait States (F28xx only)

The following F28xx flash operations require Wait States parameters:

- Erase, Program, Verify
- Program, Verify
- Verify Only

For explanation on flash wait states, see *TMS320x281x System Control and Interrupts Reference Guide (SPRU078)* by Texas Instruments.

Flasher-C2000 validates the wait state values it receives before starting the Verify flash operation. The wait state values are valid if they meet the following constraints:

- The Flash Page Wait State value is greater than zero if Flash Pipeline Mode is enabled.
- The Flash Random Wait State value is greater than, or equal to, the Flash Page Wait State value.

I the wait state values are invalid, Flasher-C2000 displays an error dialog and does not start the requested flash operation.

Specifying the COFF or HEX file

The following flash operations require a COFF or HEX file parameter:

- Erase, Program, Verify
- Program, Verify
- Program Only
- Verify Only

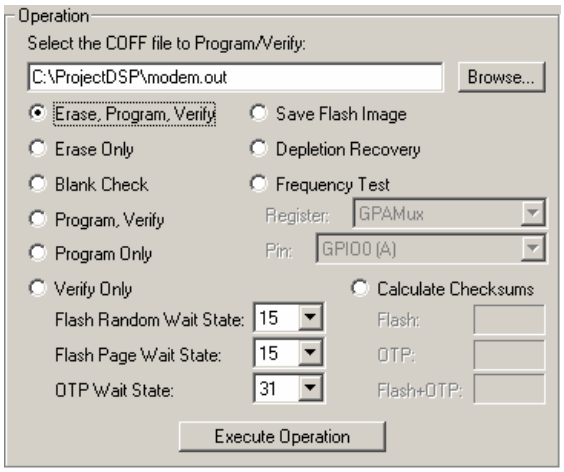

FIGURE 26

Flasher-C2000 supplies the name of the file to program or verify based on the information from the last programming session. The programmer modifies the file name if you:

- Enter a complete path to the COFF or HEX file into the edit box.
- Click on the Browse button and choose the COFF or HEX file.
- Load Flasher-C2000 settings using the Load Settings button.

Before you press the Execute button, double-check your COFF/HEX file selection.

Only linked COFF output files (*.out) can be programmed into the flash. If you choose an object file (*.obj) or an output file with unresolved references, Flasher-C2000 displays an error message and cancels the flash operation.

Flasher-C2000 supports two types or HEX files:

- Files with extension .hex — 16-bit hex files, as generated by the hex2000 utility.
- Files with extension .he8 — 8-bit HEX files.

See *Appendix F – Using HEX Files* (p. 43) for more information about the supported hex file formats.

Secure Flash upgrading in the field

Sometimes there is a necessity to program or upgrade flash devices at a customer site in a secure way, that is, without providing the customer with a flash image (usually a COFF file) in order to protect the intellectual property rights of the flash image vendor.

One method to accomplish the task is by delegating a vendor's representative to the customer site. The representative then programs the customer's device with a new flash image and secures the programmed flash device with a password unknown for the customer. This method, obviously, can be costly and inefficient if the number of customers and devices to be serviced is significant.

Flasher-C2000 allows performing such upgrades in a much easier, more economical way outlined below.

- The Vendor prepares a new flash image as a COFF file. The generated file contains a password in order to lock flash device as soon as flash programming is completed.
- The Vendor encrypts the prepared COFF file using the GenSEF.exe application, an optional component of the Flasher-C2000 package that creates an encrypted .SEF file.
- The Vendor prepares a DLL file containing scrambled password to unlock the old, pre-upgrade flash image. The source code of the skeleton DLL is an optional component of the Flasher-C2000 package.
- The Vendor prepares a command batch file with commands for Flash2000.exe application to unlock the flash using DLL file and program flash using the encrypted .SEF file. Flash2000.exe is the standard component of the Flasher-C2000 package.
- Thus encrypted COFF file, password DLL file, and command batch file containing commands to automatically unlock and upgrade the flash device are then sent to the customer.
- The customer executes the delivered command batch file, which automatically upgrades the flash image. The new flash image is locked by the Flasher-C2000 just after programming/verifying procedure is completed. Flasher-C2000 automatically performs flash verification when an encrypted flash image file is specified for flash programming. This action checks that flash device was upgraded without any errors.

Here is a sample command batch file containing three commands: Unlock, Erase and Program:

```
-Unlock Pwd -CpuDef TMS320F2808.cpu -BoardCfg 28xx.dat -JtagDev "0" -PLL
20.000x10/2

-Erase 0xf -Algorithm FlashAPIInterface2808V3_2.out -CpuDef TMS320F2808.cpu -
BoardCfg 28xx.dat -JtagDev "0" -PLL 20.000x10/2

-Program -File Img.sef -Algorithm FlashAPIInterface2808V3_2.out -CpuDef
TMS320F2808.cpu -BoardCfg 28xx.dat -JtagDev "0" -PLL 20.000x10/2
```

`Pwd` is the name of the DLL file Pwd.dll containing scrambled password to unlock the old contents of the TMS320F2808 flash image. `Img.sef` is the name of the new encrypted flash COFF image file.

Appendix A – Flasher Command Line Interface

The Flash programmer can be run by executing Flash2000.exe with a set of required parameters from the DOS command line.

A complete list of commands and parameters is listed in section *Flasher Commands,* p. 32. To facilitate the process of compiling a set of parameters for a particular flash operation, the current set of parameters is displayed in the Arguments field. The task of the user is simplified by the ability to copy this list of arguments with the use of the right mouse button. You can utilize the list in one of the two following ways:

1. **Create a single operation DOS Batch (.bat) File.**

   To create a DOS batch file to perform a single flash operation, such as Erase, Program, or Verify, run Flahser-C2000 without connecting to the target and select the desired flash operation along with its parameters.

   Then right-click in the Arguments list, choose Select All, and right-click again to select Copy to store the arguments in the clipboard. In your favorite text editor, type "Flash2000.exe" (without the double quotes) and paste the contents of the clipboard to form a list of arguments trailing the command.

   For instance, a batch file erasing the F2808 device might look like this:

   ```
   Flash2000.exe –Erase 0xf –Algorithm FlashAPIInterface2808V3_0.out –Timeout 15
   –CpuDef "Algos\TMS320F2808\TMS320F2808.CPU" –BoardCfg
   "C:\FlashC2000\Single_28xx.dat" –Log 1
   ```

   Such a batch file can perform only one flash operation at a time. To perform several of them from one batch file, use the alternative method described in the next section.

   Access to batch files may be facilitated by placing links to them on the Desktop, or by incorporating the files into the internal test and production software systems.

2. **Create a DOS Batch (.bat) file with a flash command file (.txt)**

   This method allows the user to create a batch command file that performs multiple flash operations. A set of such operations is executed by passing Flasher-C2000 the name of the batch file like this.

   ```
   Flash2000.exe –Use filename.txt
   ```

   Begin by creating a set of commands with their arguments as described in method 1 above.

   Consider the following example of a command file that performs the Erase, Program, Verify, Program Password and Lock operations on the F2808 device.

   ```
   –Erase 0xf –Algorithm FlashAPIInterface2808V3_0.out –PLL 20.000x10/2 –Log 1 –
   ```

```
CpuDef "Algos\TMS320F2808\TMS320F2808.CPU" -BoardCfg
"C:\FlashC2000\single_28xx.dat"

-Program -File "C:\Flasher-C2000\testing\2808_LEDBlinky.out" -Algorithm
FlashAPIInterface2808V3_0.out -CpuDef "Algos\TMS320F2808\TMS320F2808.CPU" -PLL
20.000x10/2 -BoardCfg "Single_28xx.dat"

-Verify int -File "C:\Flasher-C2000\testing\2808_LEDBlinky.out" -PageWS 15 -
RandomWS 14 -OTPWS 30 -Algorithm FlashAPIInterface2808V3_0.out -CpuDef
"Algos\TMS320F2808\TMS320F2808.CPU" -BoardCfg "Single_28xx.dat" -PLL 20.000x10/2

-Password "0xFFF0 0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF" -Algorithm
FlashAPIInterface2808V3_0.out -CpuDef "Algos\TMS320F2808\TMS320F2808.CPU" -
BoardCfg "Single_28xx.dat" -PLL 20.000x10/2

-Lock -CpuDef "Algos\TMS320F2808\TMS320F2808.CPU" -BoardCfg "Single_28xx.dat" -
PLL 20.000x10/2
```

Note that each operation is specified on a separate line. Assuming that the name of this command file is program2808.txt, its commands can be executed with the following DOS prompt command:

```
Flash2000.exe –Use program2808.txt
```

If the flasher program or the command file reside outside of the Flasher-C2000 installation folder, the command must include full file paths.

Flasher Commands

COMMAND	COMMAND PARAMETER(S)
-Use	Name of the .txt command file that contains all programming command options.
-CpuDef	Name of the CPU description file (.cpu).
-Boardcfg	Name of the ccBrd.dat board configuration file (.dat).
-JtagDev	Name or 0-based index of the selected DSP in the ccBrd.dat file (default is 0).
-CCSDrv	Name of the CCS driver (the default is copied from the CPU description file).
-PLL	OSCCLK x PLL_Ratio / PLL_Divisor (text string — the default is copied from the CPU description file).
-Algorithm	Name of the COFF programming algorithm file (.out).
-Erase	Sector bit mask (16-bit hex value), bit 0 is sector A or 0, bit 7 is sector H.[1)]
-Clear	Sector bit mask (hex value), bit 0 is sector 0.[1)]
-ClearCheck	Sector bit mask (hex value), bit 0 is sector 0.[1)]
-Program	The COFF file is specified with the –File command below.
-File	Name of the COFF application file for Program/Verify (*.out); Name of HEX or BIN file for SaveFlashImage (.hex, .bin).
-Lock	For F240x, four 16-bit password keys to be programmed: PWL0 - PWL3. For F28xx, ignored.
-Unlock	For F240x, four 16-bit password keys programmed into device: PWL0 - PWL3. For F28xx, eight 16-bit password keys: Key0 - Key7.
-Password	For F240x, the password keys must be included with the Lock/Unlock commands For F28xx only: eight 16-bit hex values: Key0 - Key7.
-Verify	Int (internal by the DSP - default), Ext (by the host PC).
-PageWS	Number of Flash Page Wait States for Verify Int (decimal value). If not specified,

COMMAND	COMMAND PARAMETER(S)
	the defaults are copied from the CPU description file.
-RandomWS	Number of Flash Random Wait States for Verify Int (decimal value), (the default is copied from the CPU description file).
-OTPWS	Number of OTP Wait States for Verify Int (decimal value — the default is copied from the CPU description file).
-OTP	No parameters. This command can be used only for F28xx with –Program or –Verify to specify programming/verifying OTP instead of Flash memory.
-DepletionRecovery	
-BlankCheck	
-SaveFlash	Sector bit mask (16-bit hex value), bit 0 is sector A (sector 0 in F240x)[1].
-Checksum	
-VerifyChecksum	For F28xx reference values of Flash and OTP checksums: flchksum,otpchksum For 240x reference value of Flash checksum: flchksum. Checksum values must be specified as 16-bit hex numbers (with or without 0x). The F28xx requires one hex value, but a comma must be used if only verification of the OTP checksum is required.
-FtestStart	
-PinMask	Pin Mask for Frequency Test (hex value).
-MuxReg	Address of Mux Register for 281x Frequency Test (hex value).
-ToggleReg	Address of Toggle Register for 281x Frequency Test (hex value).
-FtestStop	
-Exec	Name of CPU control command script file.
-Timeout	Timeout for Erase, Program and other operations in seconds (default is copied from the CPU description file) – decimal value.
-Log	Logging level (decimal value 0-6). The default is 1.
-ST1value	Value of the ST1 register to be set (hex).
-Ioaddr	IO address to be written (hex).
-Iovalue	Value to be written at IOaddr (hex).
-ReadBack	None (no reading back flash data), Algo (using algorithm), All (both)
-Emulator	Serial number of the emulator to connect to. The default value 0 allows you to connect to any emulator.

[1] TMS320F24x devices have only one Flash sector (sector 0), and the value of sector mask must be specified as 0x4000.

TABLE 8

Appendix B – 320F24x with Non-Standard Device ID

Each type of the 320F28xx and 320F24xx DSPs has a unique device identifier. Device identifiers for 320F28xx parts are read from the PARTID register; for 320F24xx parts, they are read from the DINR register. Flasher-C2000 provides processor description file (*.CPU) for each supported DSP, which, among other details, contains also value of the device identifier. This allows the Flasher-C2000 to identify connected DSP and verify whether the DSP is the same as selected by user.

However, some old 320F24x DSPs have assigned non-standard device identifiers. In this case value of device identifier read from DINR register does not match value of DINR defined in *.CPU file for the corresponding DSP. To allow programming such old 320F24x parts the Flasher-C2000 allows mapping of the non-standard device identifiers to their corresponding standard values, the mapping is done by means of the file DINR24Map.txt, which has to be placed in the Flasher-C2000 main installation directory. The contents of the file can be prepared by a customer or by Signum Systems Technical Support on the customer request.

The file DINR24Map.txt is a text file that can be created and modified using any text editor. First line of the file must contain text `[MapEntries]`, followed by lines specifying device identifier mappings. Each mapping must be placed on a separate line. Lines beginning with semicolon are regarded as comment lines and thus are ignored by the Flasher-C2000.

A line describing device identifier mapping entry has the following form:

```
Entryn=<dsp-name>   <orig-dinr-value>   <mapped-dinr-value>
```

where `n` is a mapping entry index number >= 0. Subsequent entries must have assigned subsequent indices = 0,1,2 … etc.

Parameter <dsp-name> holds name of the non-standard DSP, used only for logging purposes when LogLevel is >= 2. The name must consist only of letters, digits and underscore characters.

Parameter <orig-dinr-value> holds hexadecimal value of non-standard device identifier, as read from the DINR register.

Parameter <mapped-dinr-value> holds hexadecimal value of standard device identifier to be used instead.

Example DINR24Map.txt file containing two device identifier map entries may look like below:

```
[MapEntries]
Entry0=MyF240         0xC010    0x0010
Entry1=TMP320F240     0x00F0    0x0010
```

Non-standard device identifiers (read from DINR) are 0xC010 and 0x00F0, in both cases they are handled as a regular TMS320F240 having standard DINR defined as 0x0010.

However, there are two important issues that must be remembered by customers who decide to use non-standard device identifier mappings:

- The Flasher-C2000 will use flash programming algorithms prepared for DSP identified by <mapped-dinr-value>. These algorithms may check DINR register and abort programming if unsupported value of DINR is detected.
- Signum Systems does not guarantee and does not take any responsibility for quality of flash programming on DSPs having non-standard values of DINR registers. Flash programming algorithms used by the Flasher-C2000 were prepared for DSPs having standard device identifiers <mapped-dinr-value> and they may not work reliably with the DSPs having non-standard device identifiers.

Appendix C – Error Messages

Problem Connecting to an Emulator

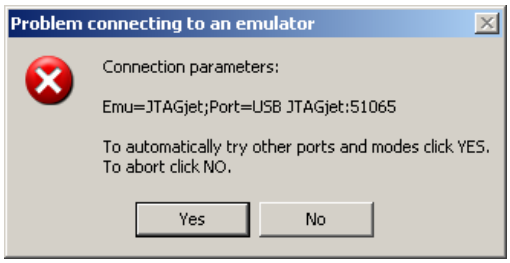

FIGURE 27

Cause: JTAGjet-TMS-C2000 not connected to the PC's USB port.

Please connect the emulator to the USB port of the PC and click the YES button.

License Error

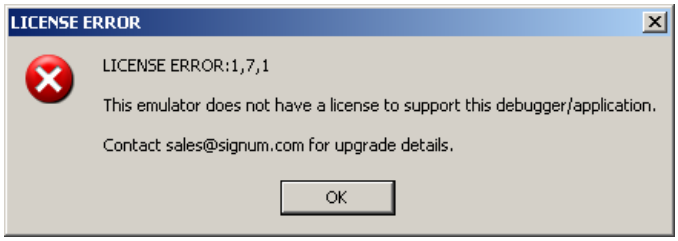

FIGURE 28

Cause: JTAGjet-TMS-C2000 is not enabled or does not have the proper license file to work with the Flasher-C2000.

You must first obtain a license file from the distributor or manufacturer to use the Flasher software.

Please see Appendix D - License Installation (p. 40) for information about obtaining and installing the license files.

Flash API Error Messages

The following Flash API error messages may appear during flash operations that do not succeed. Detailed information is supplied by the Flash API for some errors. This information is defined below:

First Fail Address The first address at which the failure occurred.

Expected Data The target value that the Flash API expected to encounter.

Actual Data The target value the Flash API actually encountered.

Precondition Failure

The Erase operation failed the Precondition step.

Erasing Flash Memory involves three separate steps: Preconditioning, Erasing, and Compacting. This error occurs if Flash Memory fails the Precondition step of an Erase, Program, Verify flash operation or Erase Only flash operation.

First Fail Address The first address which failed to Precondition correctly.

Erase Failure

The Erase operation failed the Erase step.

Erasing Flash Memory involves three separate steps: Preconditioning, Erasing, and Compacting. This error occurs if Flash Memory fails the Erase step of an Erase, Program, Verify flash operation or Erase Only flash operation.

First Fail Address The first address which failed to Erase correctly.

Compaction Failure

The Erase operation failed the Compact step.

Erasing Flash Memory involves three separate steps: Preconditioning, Erasing, and Compacting. This error occurs if Flash Memory fails the Compaction step of an Erase, Program, Verify flash operation or Erase Only flash operation.

First Fail Address The first address which failed to Compact correctly.

Program Failure

The Program operation failed.

This error occurs if the verification step during the program portion of an Erase, Program, Verify flash operation, a Program, Verify flash operation, or a Program Only flash operation fails.

First Fail Address The first address that failed to program correctly.

Expected Data The data that should be programmed.

Actual Data The data the verification step found to be in Flash Memory.

The Flash API goes through two steps during the program portion of any flash operation that programs Flash Memory: Programming and Verifying. After the programming step, the Flash API verifies that what it programmed is actually in Flash Memory. This verification is not the same as the verification in the Erase, Program, Verify flash operation, or the Program, Verify flash operation, or the Verify Only flash operation.

For example, during the Program, Verify flash operation, the Flasher-C2000 sends a block of the selected COFF file to the target to be programmed by the Flash API. After the Flash API programs that block of data, it immediately verifies Flash Memory against the data sent by the Flasher-C2000. After the programming part of the Program, Verify flash operation has completed, the Flasher-C2000 sends the same block of data to the target again and asks the Flash API to verify against the re-sent data. This latter verification is the verify step of the Program, Verify flash operation.

Zero Bit Failure

There was a Zero Bit Error on the Program Flash operation.

Programming Flash Memory requires that it be erased first. Erasing sets all bits in Flash Memory with a value of one. This error occurs if you attempt to set a bit to zero in Flash Memory and it does not currently have a value of one. This error can occur during the Erase, Program, Verify flash operation, the Program, Verify flash operation, or the Program Only flash operation.

For example, if you program a location in Flash Memory with the value 0xFFFE, then bit zero has a value of zero. If you then attempt to program the same location with the value 0xFFFE without first erasing the location, you will get a zero bit error.

First Fail Address The address with the Zero Bit error.

Expected Data The data to be programmed.

Actual Data The data actually in Flash Memory.

Verify Failure

The Verify operation failed.

This error occurs if the Verify portion of an Erase, Program, Verify operation; a Program, Verify operation; or a Verify Only operation fails.

First Fail Address The first address that failed to Verify correctly.

Expected Data The data that should be in Flash Memory.

Actual Data The data the Verification step found to be in Flash Memory.

Timeout waiting for CPU Stop

This error occurs when the DSP does not finish the requested operation in time. As a remedy, try the following:

1. Repeat the operation—subsequent attempts are often likely to succeed.
2. If method 1 does not solve the problem, use the Disconnect from Target button to disconnect and then reconnect. Repeat the operation that timed-out.
3. If method 2 does not solve the problem, you must reset the DSP to clear the error. Press the DSP Reset switch on your board, or power cycle your target board.

Appendix D - License Installation

Emulators (or programmers) purchased with the flash programmer option have a programmer license preinstalled at the factory. The programmer works out of the box.

A license error occurring when you try to connect to the programmer is an indicator that the flash programmer license is missing. You can purchase the license from your local distributor (www.signum.com/distribu.htm) or directly from Signum Systems (sales@signum.com). You will be provided with a .lic license file.

Install the license by placing the .lic license file in the Flash-C2000 installation directory. Your license file must match the serial number of the emulator (or programmer). The base component of the license file name contains the serial number. The emulator's serial numbers is printed on the bottom of the unit, or it can be determined using the EmuDiag program downloadable from www.signum.com/support.htm.

If your computers need to be frequently connected to different flash programmers, copy all your license files to the Flash-C2000 installation folders on all the computers. This will allow you to use any programmer available with any of your computers.

Appendix E - JTAG Settings Adjustment and Test

You can change JTAG clock speed and test the JTAG scan chain with the Test JTAG button (FIGURE 29).

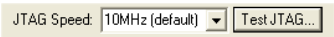

FIGURE 29

It is recommended that you keep the default clock setting of 10MHz. This setting is appropriate for the XDS510 & XDS560 emulators. Select a faster JTAG clock only if you need a very high programming speed, provided the speed is within the allowed range of the target. As a rule of thumb, the DSP clock frequency should be at least 3 times the JTAG clock frequency.

Test JTAG button also allows you to test the JTAG scan chain.

To test if the JTAG chain is connected properly, click the Test button in the JTAG Clock group. The dialog box shown in FIGURE 30 appears.

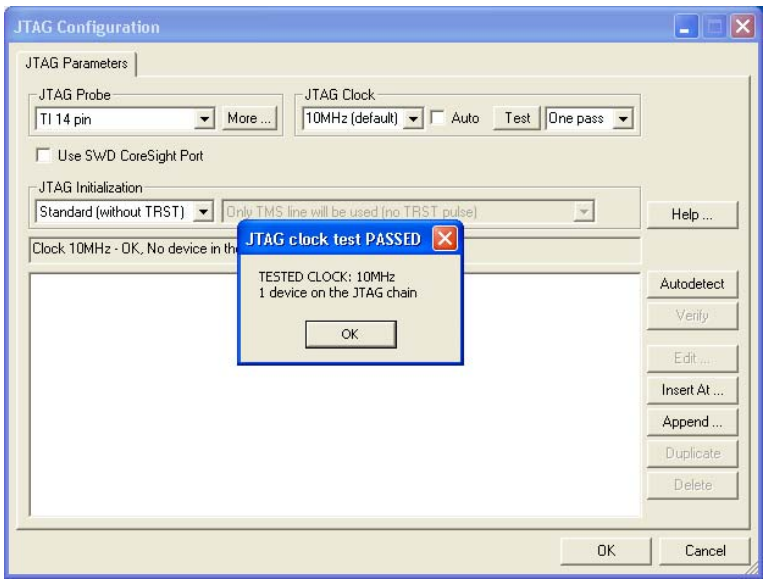

FIGURE 30

Unless the Test button is pressed, clicking OK or closing the tab will perform a clock test. If the basic JTAG clock test fails, do not continue.

To auto-detect and verify the JTAG devices on the JTAG chain, cick **Autodetect**. The target board JTAG geometry is displayed. For instance, boards with a single TMS320F28xx device display the configuration shown in FIGURE 31.

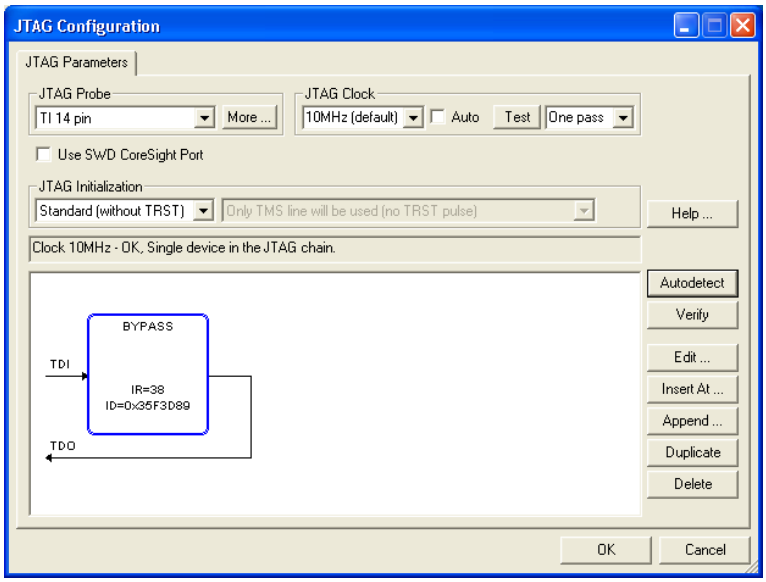

FIGURE 31

> ***Note 1:*** Do not be concerned if *Autodetect displays BYPASS. As many Texas Instruments devices share the same JTAG ID, it is impossible to accurately detect the device based on the JTAG ID alone. The Flasher interrogates the device in more detail and reports the actual device name correctly.*

The F240x/F24x devices are shown as IR=8 with ID = unknown.

> ***Note 2:*** *The results of the Autodetect operation are for informational purposes only and are not used by Flasher automatically. If more devices are detected on the JTAG chain, you must create a correct board configuration file. The Flasher creates the JTAG geometry using the supplied .dat files.*

To begin troubleshooting basic JTAG connection problems, determine the JTAG scan chain with the EmuDiag diagnostic utility. The EmuDiag documentation provides some additional JTAG problem solving tips. EmuDiag is available at www.signum.com\support.htm.

Appendix F – Using HEX Files

Beginning with version 1.15, Flasher-C2000 supports programming, verifying and saving Intel HEX files. Since the F28x and F24x devices are 16-bit word addressable processors (a single addressable memory cell holds a 16 bit value), whereas the Intel HEX file format was designed for byte addressable processors, the Flasher's support of HEX files warrants clarification.

Generating 16-bit (word addressable) HEX file

The **hex2000** tool, which is part of the C2000 compiler package, can be used to create different types of HEX files based on the linker-generated OUT file. The following command creates a 16-bit HEX file (name.hex) out of an OUT file (name.out):

```
hex2000 name.out -i --romwidth=16 -o name.hex
```

The **--romwidth=16** parameter is essential. Without it, the tool would create two byte addressable Intel HEX files with odd and even bytes for each 16-bit word. Such split HEX files are not supported, since Flasher-C2000 is able to program only one file at a time.

Generating 8-bit (byte addressable) HEX file

Byte addressable hex files cannot be easily generated from C2000 OUT files. However, if you program images, checksums, or other data tables not generated by C2000 tools, 8-bit HEX files may prove more convenient.

8-bit/16-bit HEX file differences

Consider the following fragment of a 16-bit HEX file containing sample code at location 0x3F4000. (To visually distinguish different parts of HEX format, hex numbers in each line have been separated with spaces.)

```
:02 0000 04 003F BB
:20 4000 00 767F409D3B1076260000762F00007622761F01C01A1C0008FF6976FD7C803E67 9A
:20 4010 00 761F01C0181CFFF7761A76221812FEFF1812FFFE1A1220001A1240001A120400 B2
```

The same code in an 8-bit HEX file would look like this.

```
:02 0000 04 007E 7C
:20 8000 00 7F769D40103B267600002F76000022761F76C0011C1A080069FFFD76807C673E 5A
:20 8020 00 1F76C0011C18F7FF1A7622761218FFFE1218FEFF121A0020121A0040121A0004 62
```

In a 16-bit hex file, address and offset are 16-bit long, that is, the same as in the C2000 architecture. Values are stored as sequences of 16-bit words.

In an 8-bit hex file, each byte must be addressed. Therefore the address and the offset are 2x of word address. Values are stored as sequence of bytes.

The key differences then are as follows (compare the highlighted elements of the 8-bit code with their counterparts in the 16-bit code.)

- The address is 0x7E8000 instead of 0x3F4000, i.e., twice as large.
- The offset in the third line is 0x8020 instead of 0x4010, i.e., twice as large. (But each line has 32=0x20 bytes in it).
- The data bytes are reversed—0x7F76 becomes 0x767F)—since in 8-bit hex files, there are sequences of bytes, as opposed to 16-bit words. In both cases, the value written to memory is 0x767F.

Selecting file formats

The selection of the format of a file to be programmed, verified, or saved is accomplished through the choice of the filename extension (TABLE 9).

EXTENSION	FILE DESCRIPTION
.out	COFF file.
.hex	16-bit word addressable HEX file (native to C2000).
.he8	Byte addressable HEX file, provided for convenience.
other	Treated as a COFF file.

TABLE 9

❑