TMS320F2810, TMS320F2811 and TMS320F2812 SDFlash JTAG Flash Programming Utilities

SDFlash Algo Version 4.1

This download includes SDFlash algorithm files used to interface the FLASH API to the SDFlash JTAG flash utility.

These algorithms use the following Flash APIs

- **TMS320F2810 V2.10**
- **TMS320F2811 V2.10**
- **TMS320F2812 V2.10**

SDFlash is a product of Spectrum Digital Inc. (www.spectrumdigital.com)

Document updated:
August 04, 2005

SDFlash Algo Disclaimer

The SDFlash Algorithm files in this download include the TMS320F2810, TMS320F2811, and TMS320F2812 Flash Application Program Interface (Flash API) libraries:

- **Flash2810_API_V210.lib**
- **Flash2811_API_V210.lib**
- **Flash2812_API_V210.lib**

Texas Instruments Inc. (TI) reserves the right to update or change any material included with this release. This includes:

- **The API functional behavior based on continued TMS320F2810, TMS320F2811, TMS320F2812 testing.**
- **Improvements in algorithm performance and functionality.**

It is the user's responsibility to check for future updates to the SDFlash Algo Files based on any updates to the TMS320F2810, TMS320F2811, or TMS320F2812 Flash API and to use the latest version available for their TMS320F2810, TMS320F2811, or TMS320F2812 silicon.

Updates to the API will be posted on the Texas Instruments Inc website (www.ti.com) and can also be obtained by contacting a local TI representative or the TI Product Information Center.

Updates to the SDFlash algorithm files based on these APIs will be posted on the Spectrum Digital website (www.spectrumdigital.com.)

Contents:

1. Revision History

This is the revision history for the TMS320F2810, TMS320F2811, and TMS320F2812 algorithm builds for SDFlash. Note that the SDFlash algorithm build version is separate from the SDFlash front-end utility version.

Changes from Algorithm V4.0 to V4.1

- Updated to use the V2.10 release of the TMS320F2810, TMS320F2811 and TMS320F2812 Flash APIs.

Changes from Algorithm V3.0 to V4.0

- Updated to use the V2.00 release of the TMS320F2810, TMS320F2811 and TMS320F2812 Flash APIs.
- Added a separate SDFlash project for depletion recovery. Refer to section 10.

Changes from Algorithm V2.1 to V3.0

- Updated to use the TMS release of the F281x Flash API's (V1.00) and added support for the F2811 device.
- Minor changes to the procedure to customize the operating frequency based on the Flash API.
- Changed the install directory name to tif281x_<version> to indicate the F281x devices.

Changes from Algorithm V2.0 to V2.1

- Corrected erase algorithm voltage setting.
- Added ISR stub functions to catch ITRAP routines. This change improves SDFlash operation when the device is set to "boot to flash" mode.

Changes from Algorithm V1.0 to V2.0

- Support for F2810/12 TMX Rev C silicon only. Version 1.0 supported Rev A silicon.
- Added custom CPU Frequency and PLL multiplier support.
- Added One Time Programmable (OTP) block programming support.
- Added user specified wait states for the verify operation.
- Attempting to program a RAM location is now handled properly. In V1.0, this could result in programming an unknown location in flash. This condition is now caught by the algorithms and will result in program failure.
- Leaving user options blank no longer results in garbage data being fed to the algorithms.

2. Release Notes

- This release of the SDFlash algorithms is based on the TMS320F2810, TMS320F2811, and TMS320F2812 Flash API V2.00 (SPRC125) release. This API release is available as a stand-alone download on the TI website and can be used to develop custom programming solutions.

- Some traditional programming utilities have separate operations for "clear" or "pre-condition" and "erase". These two operations have been combined into one operation referred to only as "erase".

  Note: The CSM will be <u>permanently</u> locked if the CSM password locations are loaded with all 0x0000 and the device is secured. During the erase API function, a sector clear (program all bits to 0x0000) is immediately followed by an erase operation without resetting the device. This will help avoid permanently locking the CSM. Do not program the CSM passwords with all 0x0000.

- SDFlash does not support the XDS560 scan controller.

- You should not run SDFlash and Code Composer Studio at the same time. This results in two different applications trying to control the DSP. During programming, SDFlash will have complete control of the device. No user application code can be running in parallel.

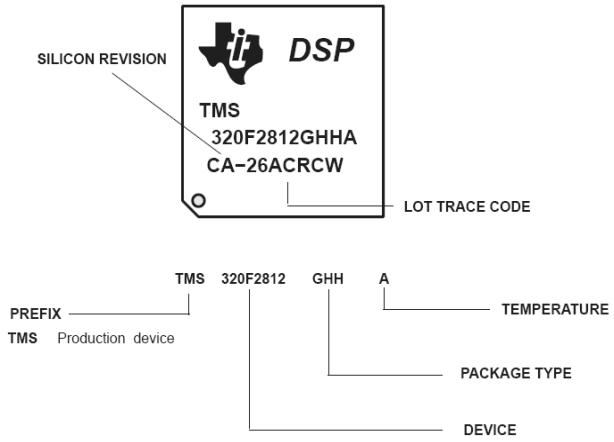
3. Known Limitations

- SDFlash only programs sections linked to page 0 (i.e. program) memory. It is important to make sure that sections such as .switch, and .const/.econst section are linked to page 0 and not page 1 for SDFlash to program these sections.

- When specifying User Options for SDFlash functions **do not** use a leading 0x in front of the number. Enter only the hex value with no leading 0x. For example: correct: 03FF incorrect: 0x03FF

4. SDFlash vs. Silicon Revision

The version of the SDFlash algorithm build must be correct for the version of silicon being programmed. The silicon revision can be determined by the lot trace code marked on the top of the package. The figure below provides an example of the TMS320F281x markings. Some prototype devices may have markings different from those illustrated. Refer to the *TMS320F2810\F2811\F2812, TMS320C2810\C2811\C2812 DSP Silicon Errata* (literature number SPRZ193) for information on how to determine your device's silicon revision.



Second Letter In Prefix of Trace Lot Code [1]	REVID (Addr 0x883)	Obsolete SDFlash JTAG Algorithms These Algorithms are Obsolete and No Longer Recommended [2]	Recommended SDFlash JTAG Algo
Blank	0x0000	V1.0 (tif28x_v1)	**tif28x_v1.zip**
A	0x0001	V1.0 (tif28x_v1)	**tif28x_v1.zip**
B	0x0002	none	**none**
C	0x0003	V2.1 (tif28x_v2_1) V3.0 (tif281x_v3_0) V4.0 (tif281x_v4_0)	**V4.1 (tif281x_v4_1) or Later**
D	0x0003	V3.0 (tif281x_v3_0) V4.0 (tif281x_v4_0)	**V4.1 (tif281x_v4_1) or Later**
E	0x0005	V3.0 (tif281x_v3_0) V4.0 (tif281x_v4_0)	**V4.1 (tif281x_v4_1) or Later**
F	0x0006	V3.0 (tif281x_v3_0) V4.0 (tif281x_v4_0)	**V4.1 (tif281x_v4_1) or Later**
G	0x0007	V4.0 (tif281x_v4_0)	**V4.1 (tif281x_v4_1) or Later**
Later	>= 0x0008	V4.0 (tif281x_v4_0)	**V4.1 (tif281x_v4_1) or Later**

Notes:

1) Silicon Revisions B, D and F were TI only internal test revisions.

2) There are no reliability issues with devices prior to revision G that are programmed with v1.0, v2.1 or v3.0 of the SDFlash JTAG algorithms. Erasing at zero degrees C with v1, v2.1 or 3.0 algorithms will, however, yield higher than expected fallout.

The following SDFlash algorithms are obsolete and should not be used:

- **Version 4.0: tif281x_v4_0.zip**

  Version 4.0 of the SDFlash JTAG algorithms was available for download during the week of July 18th 2005. This release was based on the TMS320F2810, TMS320F2811 and TMS320F2812 Flash API V2.00. Using V2.00 of the Flash API to program the OTP will disturb erased bits within sector J of the main flash array. For this reason, V2.00 of the API is now considered obsolete. Users who downloaded V4.0 of the SDFlash JTAG algorithms should migrate to V4.1.

- **Version 3.0: tif281x_v3_0.zip**

  This release was based on the TMS320F2810, TMS320F2811 and TMS320F2812 Flash API V1.00. This API will not program or erase the flash as of Rev G silicon and will report one of the following error codes:

Erase:	STATUS_FAIL_ERASE (error code # 22)
Program:	STATUS_FAIL_ZERO_BIT_ERROR (error code # 31)
Verify:	Verify will still operate as expected

  The attempt to erase and program the flash will have no effect and no change will have been made to the contents of the flash.The only exception to this is if an attempt is made to program all zeros (0x0000) into the flash. In this case, the V1.00 API will incorrectly report that the programming operation successfully completed, however no change will have been made to the contents of the flash. This is a very unusual case and typically does not occur in a customer's system.

- **Version 2.0: tif28x_v2_1.zip**

  These APIs operated on REV C silicon only.

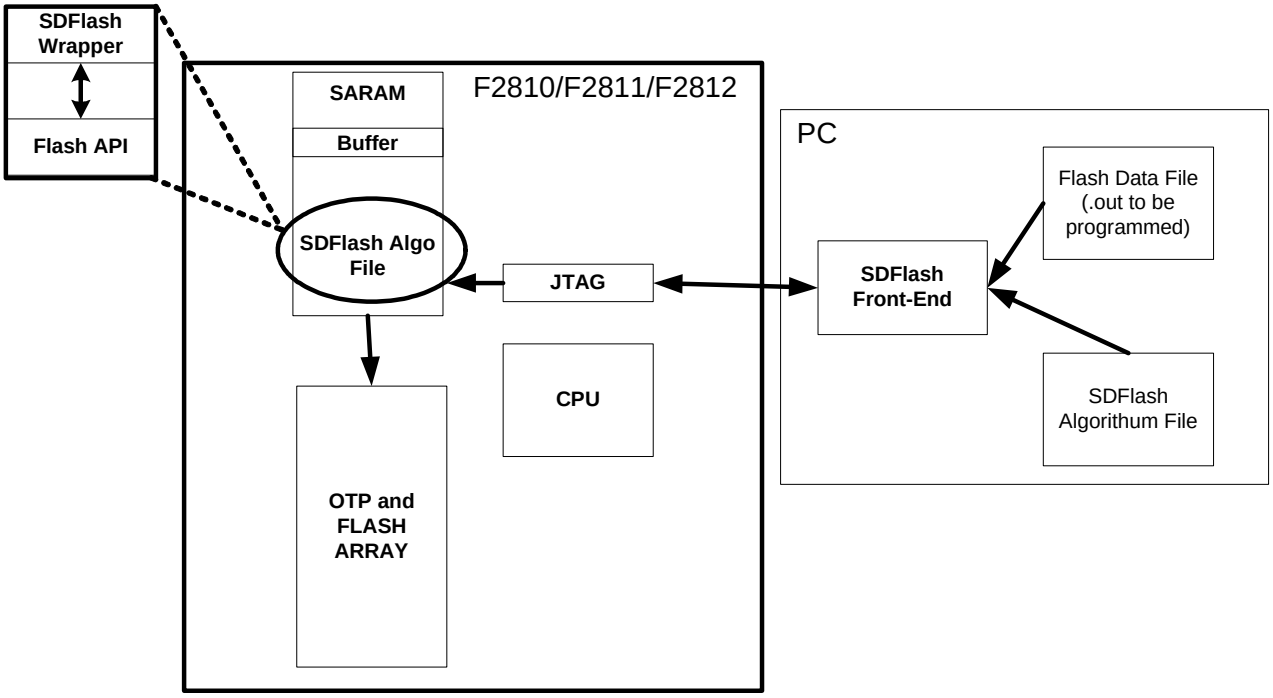
  For future silicon revisions, TI anticipates that no functional changes will be required to the Flash API that the SDFlash algorithm uses in this release. Should API changes occur that affect the programming of the flash, it is the user's responsibility to update to the new SDFlash algorithm files. TI will test these APIs on future silicon revisions as soon as possible when such devices become available. SDFlash algorithm builds for use with SDFlash can be downloaded from the Spectrum Digital (SD) website (www.spectrumdigital.com).

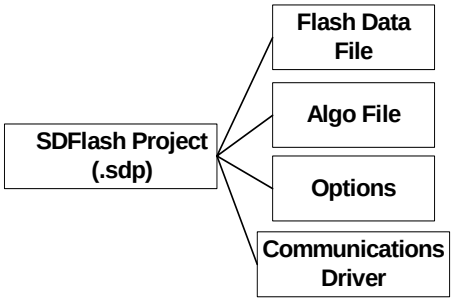
5. SDFlash Overview

SDFlash is a generic front-end application owned by Spectrum Digital Inc (www.spectrumdigital.com). This application provides a generic interface to the JTAG communications channel that can be used to support flash programming.

In order to erase or program flash, SDFlash downloads a flash algorithm file onto the DSP. This algorithm file is an executable (.out) file for the DSP being programmed and is executed on the DSP target. The SDFlash algorithm file consists of an SDFlash wrapper and the device specific Flash API library. The SDFlash wrapper has well defined standard functions and variables that are accessed by SDFlash over the JTAG channel. These functions in turn make calls to the Flash API Library to perform operations on the flash array. The Flash API library used by SDFlash can be downloaded from the TI website: *TMS320F2810, TMS320F2811 and TMS320F2812 Flash API (literature number SPRC125).*

Note: because the SDFlash interface is separate from the algorithm file, the version of the SDFlash interface will differ from the version of the algorithm file.





For SDFlash to erase or program the flash on a device, it must have the following information:

- Location of the SDflash algorithm file.
- Location of the flash data file – that is the data (.out file) to program into the device.
- Any user defined options.
- Which JTAG driver to use
- Information about the JTAG scan chain

All of this information is stored in an SDFlash project file (.sdp) that can be edited though the SDFlash GUI. A sample SDFlash project for each device has been provided in this download.

Section 6 of this document will guide you through the SDFlash setup and show you how to use the provided sample SDFlash projects to create your own project to flash your device. For example code that executes from the flash, refer to the following available from the TI website:

- *C281x C/C++ Header Files and Peripheral Examples* (SPRC097)
- *Running an Application from Internal Memory on the TMS320F281x DSP* (SPRA958)

6. Quick Start Guide

The following is a step-by-step guide for using the SDFlash utility to program your device.

This quick start guide will refer to the following default directory locations:

<CCS base>	typical Code Composer Studio install directory: "c:\ti"
<SDFlash base>	default SDFlash directory <CCS base>\specdig\SDFlash

The default directories may be different for your particular installation. For example, the CCS base directory for some Code Composer Studio installs will be C:\CCStudio_v3.1.

In this case, the SDFlash project will need to be updated to reflect the directory structure of your install as described in step 6.7.

6.1. Run the SDConfig utility to make sure the target and emulator are setup properly.

The latest version of the SDConfig utility is included in Spectrum Digital's Code Composer emulation driver install package for C2000. This installation can be downloaded from the Spectrum Digital (**www.spectrumdigital.com**) website in the *CodeComposerDrivers->C2000* download section if it is not already installed on your system.

SDConfig will typically be installed in your <CCS base>\specdig\sdconfig directory

6.2. Install the SDFlash flash support utility.

The SDFlash utility is distributed as component of the standard Spectrum Digital emulation drivers. In the past, SDFlash was distributed as a standalone install. If you already have an older version of SDFlash or emulation drivers installed, you should un-install the older version first and install the latest drivers. The Spectrum Digital C2000 emulation drivers are at the following URL: h*ttp://emulators.spectrumdigital.com/c2000/*

SDFlash will typically be installed in your `<CCS base>\specdig\SDFlash` directory.

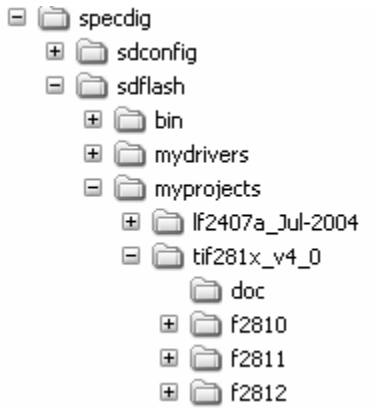
SDFlash is a generic utility supplied by Spectrum Digital Inc. to interface to user written flash algorithms. In this case, Texas Instruments Inc has supplied the algorithm file. Users should check the SD website for updates to this utility.

6.3. Download the latest SDFlash algorithm files for TMS320F2810, TMS320F2811 and TMS320F2812.

The latest SDFlash algorithm .zip file can be downloaded from the Spectrum Digital website at the following URL: *http://emulators.spectrumdigital.com/utilities/*. In addition, check Spectrum Digital's website for future updates to the algorithm.

6.4. Unzip the SDflash algorithm files into the myprojects subdirectory of SDFlash.

For a typical install this will be the <CCS base>\specdig\SDFlash\myprojects directory.



This will automatically create a directory indicating the processor and version of the utilities.

Note: If you had installed an earlier release of SDFlash on your system you may have additional sub-directories to those shown, such as an algo directory. With the release of SDFlash V1.3 the algo directory was replaced with the *myprojects* directory. Presence of this directory will not effect the operation of SDFlash.

6.5. **Run SDFlash**.

6.6. **Load the supplied SDFlash sample project**. SDFlash uses project files to store information required to erase a device and program an .out file into a device. Sample projects for each device have been included for use as project templates.

Using *File->Open Project* in SDFlash, browse to and load the appropriate sample SDFlash project. For a typical installation, these files will be found in the following location:

F2812 eZdsp*:	<SDFlash base>\myprojects\tif281x_v4_1\f2812\SampleF2812eZdsp.sdp
F2812:	<SDFlash base>\myprojects\tif281x_v4_1\f2812\SampleF2812.sdp
F2811:	<SDFlash base>\myprojects\tif281x_v4_1\f2811\SampleF2811.sdp
F2810:	<SDFlash base>\myprojects\tif281x_v4_1\f2810\SampleF2810.sdp

* This project uses the eZdsp locked on-board emulation driver.

6.7. **Modify the SDFlash project (if required) to locate the various elements such as device driver, algorithm file and flash data file.**

If you installed Code Composer Studio (CCS) and SDFlash in the <CCS base> and <SDFlash base> directories shown below, then usually only the *Flash Data File* on the **Program Tab**, and possibly the *Emulator Address/ID* on the **Target Tab** needs to be changed.

By default all flash projects are setup relative to the default TI CCS base directory "c:\ti". For example:

<CCS base>	default Code Composer Studio install directory: "c:\ti"
<SDFlash base>	default is <CCS base>\specdig\SDFlash
SDFlash binary	default is <CCS base>\specdig\SDFlash
Flash projects	default is <CCS base>\specdig\SDFlash\myprojects\<projectname>

NOTE

The location of the SDFlash directories may be different for your particular installation. For example, the <CCS base> directory for some Code Composer Studio installations will be C:\CCStudio_v3.1.

In this case, the SDFlash project will need to be updated to reflect the directory structure of your install.

To change any of the directory paths or project settings from their default values, open the project settings dialog box: *Project->Settings*

The fields that require directory paths in the *Project->Settings* window are summarized below:

Target Tab:	**Erase Tab:**	**Programming Tab:**	**Verify Tab:**
-Driver	-Algorithm File	-Algorithm File	-Algorithm File
-Board File		-Flash Data File	

Target Tab:

- *Driver:* This is the Code Composer Studio™ emulation driver (*.dvr) file that is used to communicate with the target. The driver files are in the <CCS base>\drivers\ directory.
  default for the eZdsp: <CCS base>\drivers\sdgo2812eZdsp.dvr
  default for other 510PP+/SPI515 etc: <CCS base>\drivers\sdgo28x.dvr

- *Emulator Address/ID:* default is 378. This address must match the setting in your SDConfig setup.

- *Board file:* File that provides SDFlash information on how many devices are on the JTAG scan chain. For a single 28x device on the scan chain, the default board file can be used. For systems with more devices on the scan chain, use the board file generated by Code Composer Studio to access your device. This file is found in the <CCS base>\cc\bin\BrdDat directory.
  The default board file is <SDFlash base>\myprojects\tif281x_v4_1\ccBrd028x.dat

- Processor name: default is cpu_0

Erase Tab:

- *Algorithm File:*
  F2810: <SDFlash base>\myprojects\tif281x_v4_1\f2810\flash28\Debug\SDFlash2810.out
  F2811: <SDFlash base>\myprojects\tif281x_v4_1\f2811\flash28\Debug\SDFlash2811.out
  F2812: <SDFlash base>\myprojects\tif281x_v4_1\f2812\flash28\Debug\SDFlash2812.out

- *Timeout:* leave as 3000

- *User Options 1:* F2812/F2811 default is 03FF, F2810 default is 001F.
  See section 9 for more information.

- *For all other boxes the default is blank.*

Program Tab:

- *Algorithm File:*
  F2810: <SDFlash base>\myprojects\tif281x_v4_1\f2810\flash28\Debug\SDFlash2810.out
  F2811: <SDFlash base>\myprojects\tif281x_v4_1\f2811\flash28\Debug\SDFlash2811.out
  F2812: <SDFlash base>\myprojects\tif281x_v4_1\f2812\flash28\Debug\SDFlash2812.out

- *Flash Data File:* This is the .out file that you want to program into the flash. The sample image provided with the algorithm is a simple data = address pattern. The code security module password locations are not programmed by the sample image. That is, the CSM password locations are left erased (all 0xFFFF's) so that the CSM can easily be unlocked.
  If you have your own .out file ready to be programmed you can specify that file as the data file.
  The sample image *Flash Data File* is located at:

  F2810: <SDFlash base>\myprojects\tif281x_v4_1\f2810\image\Debug\imageABCDE.out
  F2811: <SDFlash base>\myprojects\tif281x_v4_1\f2811\image\Debug\imageABCDEFGHIJ.out
  F2812: <SDFlash base>\myprojects\tif281x_v4_1\f2812\image\Debug\imageABCDEFGHIJ.out

- *Timeout:* leave as 3000

- *For all other boxes the default is blank.*

Verify Tab:

- *Algorithm File:*
  F2810: <SDFlash base>\myprojects\tif281x_v4_1\f2810\flash28\Debug\SDFlash2810.out
  F2811: <SDFlash base>\myprojects\tif281x_v4_1\f2812\flash28\Debug\SDFlash2811.out
  F2812: <SDFlash base>\myprojects\tif281x_v4_1\f2812\flash28\Debug\SDFlash2812.out
- *Timeout:* leave as 3000
- *User Options 1:* default is 0F0F
- *User Options 2:* default is 001F
- *For all other boxes the default is blank.*

6.8. **Save the SDFlash project file: *File->Save Project As.*** Once you have made the required changes select ok and save the project using the name of your choice: *File->Save Project As.*

If you changed the *Emulator Address/ID* setting on the **Target Tab**, you should get a message that the current driver was unloaded and a new driver has been loaded. This operation is required to synchronize the SDFlash project settings with SDConfig. If you did not change the *Emulator Address/ID*, then you will not get this message.

You have now created an SDFlash project that can be used anytime you want to erase or program the device using these settings. Should you want to program a different .out file into the F281x flash, use this project as a template and change the *Flash Data File* on the **Program Tab.**

6.9. **Configure the algorithm for the required PLL multiplier and CPU frequency.**

For a custom CPU frequency and PLL multiplier, you must follow the instructions in section 7 to properly configure the algorithms before continuing. As supplied, the algorithm is configured for:

PLLCR = 0x000A (PLL x10/2 mode), and CPU frequency = 150MHz

CAUTION

The erase and program operations MUST be configured for the CPU clock rate (SYSCLKOUT) at which they will run. This configuration is VITAL for proper operation of the algorithm.

As supplied, the algorithm is configured for:

PLLCR = 0x000A (PLL x10/2 mode), and CPU frequency = 150MHz

For a custom CPU frequency and PLL multiplier, you must follow the instructions in section 7 to properly configure the algorithms before continuing.

6.10. **Optional: View which sections are going to be programmed (i.e. loaded) using the *View->Coff/Hex file stats* SDFlash function.**

- Make sure that no RAM locations are marked as load sections.
- Constant sections (i.e. .const/.econst) must be linked to page 0 (i.e. program memory) for SDFlash to program them.
- It is suggested to not program the CSM password locations (0x3F7FF8-0x3F7FFF) during development since flash contents will be changed often.
- If programming the OTP, note that it can only be programmed once. It cannot be erased.

6.11. **Reset the device: *Device->Reset*.** You will get a pass/fail message in the output window.

6.12. **Erase/Program/Verify your device: *Device->-Flash*.** Check or un-check the operation(s) you want to perform then select start. Each checked operation is executed from left to right, with continue on success and abort on fail. Refer to section 11 for trouble shooting tips should a failure occur.

CAUTION

Do not press the SDFlash STOP button during the erase operation.

Pressing STOP will halt the CPU before the Erase algorithm completes. This can leave the Flash in a depleted state or result in unknown Code Security Module passwords and lock the device permanently.

Other conditions that can cause the CPU to halt prior to the completion of the Erase algorithm (e.g., power loss, device reset, PC crash, etc.) can result to the same problem described above.

Pressing STOP when executing the frequency toggle test described in section 9.2 is ok.

6.13. **Optional: Repeat erase/programming for each device to be programmed.** If additional devices are to be programmed, the target can be powered down and a new target connected without closing the SDFlash utility. Once the new target is connected, reset the part (*Device->Reset*) and erase, program, verify (*Device->Flash*) the device as described in 6.11 and 6.12.

6.14. **Optional: View the flash contents using Code Composer Studio™ (CCS).** You can view the programmed flash using CCS, and compare with your source code.

Make sure that the SDFlash utility is closed before starting CCS.

Start CCS and open a memory window to view the flash contents (or use the disassembly window). In addition you can load the CCS project and load the symbols from the .out file.

7. PLL and CPU Clock Rate Configuration

CAUTION

The Flash API used by SDFlash contains several delay parameters that are implemented as software delays. Timing of these delays is VITAL to proper operation. To ensure the proper delays, the flash algorithms must be run at the correct speed.

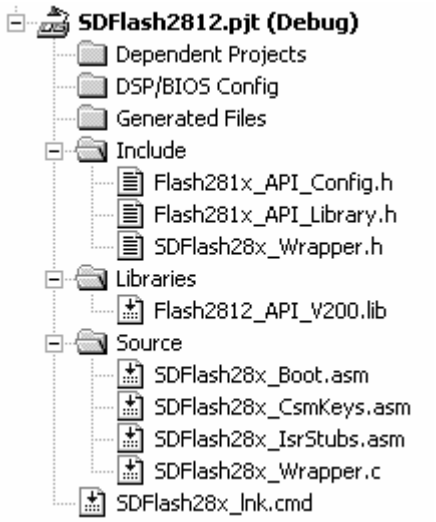
As shipped, the algorithms are configured for a 30 MHz input clock and 150 MHz CPU clock (SYSCLKOUT). To generate the 150 MHz CPU clock, the algorithms initialize the PLL control register (PLLCR) to 0x000A, which gives PLL x10/2 mode. If your hardware is running at a different CPU rate and/or requires a different PLLCR setting, then you must configure the flash programming algorithms as described below:

The following steps describe how to compile a new SDFlash algorithm file for a custom frequency and PLL Control Register setting. This configuration setup meets the requirements specified in the API documentation. Refer to Refer to *TMS320F2810, TMS320F2811 and TMS320F2812 Flash API (literature number SPRC125)* for API specific information.

7.1. Using Code Composer Studio (CCS), load the SDFlash2810.pjt, SDFlash2811.pjt or SDFlash2812.pjt.

This is the CCS project used to build the SDFlash algorithm file. For a typical install, the project will be in the following directory:

F2810: <SDFlash base>\myprojects\tif281x_v4_1\f2810\flash28\SDFlash2810.pjt
F2811: <SDFlash base>\myprojects\tif281x_v4_1\f2811\flash28\SDFlash2811.pjt
F2812: <SDFlash base>\myprojects\tif281x_v4_1\f2812\flash28\SDFlash2812.pjt



These three projects contain the following files:

SDFlash28x_Wrapper.c
Main functions called by the SDFlash front-end. These functions interface directly to the Flash API.

SDFlash28x_CsmKeys.asm
Source file used to specify the CSM passwords used to unlock the CSM before each operation.

SDFlash28x_Boot.asm
SDFlash boot and exit routines. Do not edit this file.

SDFlash28x_lnk.cmd
Linker command file. Do not edit this file.

SDFlash28x_IsrStubs.asm
ISR stub routines. Do not edit this file.

Flash API library (SPRC125) Files:
Each project contains the Flash API files for the device being programmed. This includes the API library file (example: Flash2812_API_V200.lib), the API include file (Flash281x_API_Library.h) and configuration file (Flash281x_API_Config.h). Refer to *TMS320F2810, TMS320F2811 and TMS320F2812 Flash API (literature number SPRC125)* for API specific information.

7.2. **Specify the required PLLCR (PLL Control Register) value.**
In CCS, open and modify SDFlash28x_Wrapper.h to specify the PLLCR (PLL Control Register) value. Uncomment the line corresponding to the required PLL Control Register (PLLCR) setting. This is done by removing the leading // in front of the correct line. Only one line should be uncommented.

For example: To have the algorithms initialize the PLLCR register to 0x0009 uncomment the second line and comment out the remaining lines as shown:

```
/*-----------------------------------------------------------------
*  SDFlash28x_Wrapper.h
*-----------------------------------------------------------------*/
…
// #define PLLCR_VALUE  0x000A     // SYSCLKOUT = (OSCLK*10)/2
#define PLLCR_VALUE 0x0009         // SYSCLKOUT = (OSCLK*9)/2
// #define PLLCR_VALUE 0x0008      // SYSCLKOUT = (OSCLK*8)/2
 etc …..
```

7.3. **Specify the clock rate of the CPU (SYSCLKOUT) in nanoseconds.** In CCS, open and modify the API configuration file Flash281x_API_Library.h to specify the clock rate of the CPU (SYSCLKOUT) in nanoseconds. This is done by removing the leading // in front of the correct line. Only one line should be uncommented. The file lists a number of commonly occurring clock rates. If your CPU clock rate is not listed, then provide your own definition using the examples as a guideline.

For example: Suppose the final CPU clock rate will be 135 Mhz. This corresponds to a 7.407 nS cycle time. There is no line present for this clock speed, so you should insert your own entry and comment out all other entries:

```
/*-----------------------------------------------------------------
*  Flash281x_API_Config.h
*-----------------------------------------------------------------*/
…
//#define CPU_RATE    6.667L   // for a 150MHz CPU clock speed (SYSCLKOUT)
//#define CPU_RATE    7.143L   // for a 140MHz CPU clock speed (SYSCLKOUT)
#define CPU_RATE    7.407L     // for a 135MHz CPU clock speed (SYSCLKOUT)
//#define CPU_RATE    8.333L   // for a 120MHz CPU clock speed (SYSCLKOUT)
etc …..
```

CAUTION

For flash integrity at operating frequencies, the device should always be programmed at the fastest possible CPU frequency. For example, if the CLKIN frequency is 30 MHz program the device at 150 MHz rather then 15 MHz or 30 MHz.

The flash utilities are not designed to function properly below 10 MHz.

7.4. **Rebuild the algorithms in CCS by selecting *Project->Rebuild all***

7.5. **Exit Code Composer Studio**

7.6. **Run the CPU frequency and PLL multiplier configuration toggle test described in section 9.2 from SDFlash to verify the configuration.**

When run, this test will toggle a selected GP I/O pin at a known frequency. This will allow you to confirm that the algorithms are properly configured for the CPU frequency and PLL multiplier you specified.

CAUTION

It is strongly recommended that you test the CPU frequency and PLL configuration using the configuration toggle test described in section 9.2 before erasing or programming any parts.

If this test fails, DO NOT PROCEED to erase or program the flash until the problem is corrected, or flash damage can occur.

The SDFlash algorithm file should now be configured for your hardware's frequency requirements.

8. Code Security Module (CSM) Password Considerations

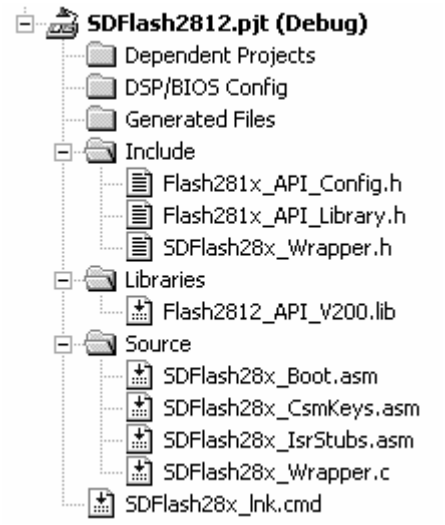
The F281x SDFlash algos must unlock the Code Security Module (CSM) before an erase, program, or verify operation. As supplied, the password locations are assumed to be all erased (all 0xFFFFs). During the code development phase, it is suggested that the CSM passwords be left erased (0xFFFFs) for ease of use. If you program new passwords into the CSM password locations (0x3F7FF8-0x3F7FFF) and then later need to re-program the part, you will need to configure the flash algorithms so that the CSM can be unlocked. Refer to *TMS320x281x System Control and Interrupts Peripheral Reference Guide*, literature #SPRU078, for details on the proper operation of the CSM.

The SDFlash algorithm uses the CSM keys provided in the SDFlash28x_CsmKeys.asm file to unlock the CSM. Follow these steps to build an SDFlash algorithm file with a new set of CSM passwords.

8.1. **Using Code Composer Studio, load the SDFlash2810.pjt, SDFlash2811.pjt or SDFlash2812.pjt**

For a typical install, the projects will be in the following directories:
F2810: <SDFlash base>\myprojects\tif281x_v4_1\f2810\flash28\SDFlash2810.pjt
F2811: <SDFlash base>\myprojects\tif281x_v4_1\f2811\flash28\SDFlash2811.pjt
F2812: <SDFlash base>\myprojects\tif281x_v4_1\f2812\flash28\SDFlash2812.pjt

These two projects contain the following files:



SDFlash28x_Wrapper.c
Main functions called by the SDFlash front-end. These functions interface directly to the F281x Flash API.

SDFlash28x_CsmKeys.asm
Source file used to specify the CSM passwords used to unlock the CSM before each operation.

SDFlash28x_Boot.asm
SDFlash Boot and exit routines. Do not edit this file.

SDFlash28x_lnk.cmd
Linker command file. Do not edit this file.

SDFlash28x_IsrStubs.asm
ISR stub routines. Do not edit this file.

F281x Flash API V2.00 library (SPRC125) Files:
Each project contains the Flash API files for the device being programmed. This includes the API library file (for example: Flash2812_API_V200.lib), the API include file (Flash281x_API_Library.h) and configuration file (Flash281x_API_Config.h). Refer to *TMS320F2810, TMS320F2811 and TMS320F2812 Flash API (literature number SPRC125)* for API specific information.

8.2. **In CCS, open the file SDFlash28x_CsmKeys.asm** . SDFlash28x_CsmKeys.asm contains the definition of the CSM passwords used by the algorithm to unlock the CSM during the erase, program and verify operations. **Note:** The passwords in this file **will not** be programmed into the CSM password locations. The algorithm uses these passwords only to unlock the CSM prior to an erase, program or verify operation.

8.3. **Modify the passwords in SDFlash28x_CsmKeys.asm to match those already programmed into the CSM password locations**.

8.4. **Rebuild the algorithms in CCS by selecting *Project->Rebuild All***

8.5. **Exit Code Composer Studio**

Code Security Module Considerations

Each step in the process Erase, Program, and Verify separately unlocks the CSM. During development the CSM passwords are typically left erased and thus this is not an issue.

If, however, you are changing the passwords during the Program operation, you will need to supply the new passwords for the verify operation by following the steps above. In other words, configure the SDFlash algorithms to use the old password. Perform the Erase and Program steps (no verify). Then, re-configure the SDFlash algorithms to use the new password (the one you just programmed). Then, go back to SDFlash and selectively perform the verify step.

.

9. SDFlash User Options

The SDFlash utility provides user options for each operation: Erase, Program and Verify. The function implemented by each of the user options is dependent on the algorithm being interfaced to.

The following table shows an overview of which user options are used by the F281x SDflash algorithm V4_1.

	Erase	**Program**	**Verify**
User Option 1	Sector Mask	Not used	Flash bank wait states
User Option 2	Run the configuration toggle test.	Not used	OTP wait states
User Option 3	Not used	Not used	Not used
User Option 4	Not used	Not used	Not used

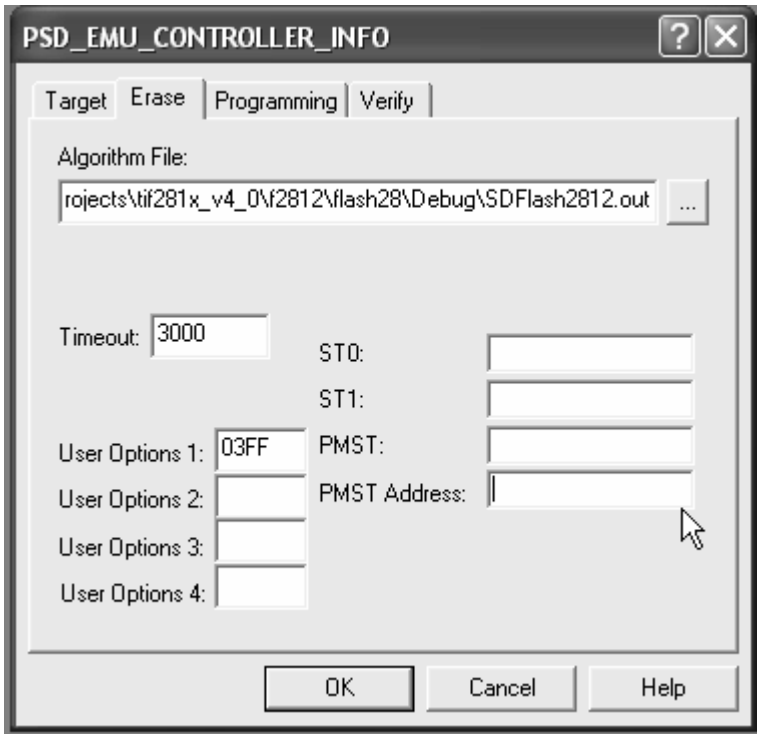
- The previous version (Version 3.0) of the algorithms used the same options shown above.

The following sections describe in detail how to use each option.

CAUTION

When entering the User Options into the SDFlash interface, do not use a leading 0x. Enter only the hex digits as shown in the examples provided.

9.1. Specify Which Sectors to Erase: Erase User Option 1



For the erase operation, User Option 1 allows you to select which sectors will be erased. This information is provided in the form of a mask value where a set bit indicates that the sector will be erased.

Bit 0 = Erase Sector A
Bit 1 = Erase Sector B
Bit 2 = Erase Sector C
Bit 3 = Erase Sector D
Bit 4 = Erase Sector E
Bit 5 = Erase Sector F (F2811/F2812 only)
Bit 6 = Erase Sector G (F2811/F2812 only)
Bit 7 = Erase Sector H (F2811/F2812 only)
Bit 8 = Erase Sector I (F2811/F2812 only)
Bit 9 = Erase Sector J (F2811/F2812 only)
Bit 10-15 = ignored

To modify the sector mask value:

9.1.1. Open the project settings:
Project -> Settings

9.1.2. Click on the Erase tab

9.1.3. Enter a mask value for User Option 1 that corresponds to the sectors you want erased.
Enter the hex number digits only. Do not enter a leading 0x.
For the example shown: Erase User Option 1 = 03FF would erase all sectors on a TMS320F2812 device.

9.1.4. Select OK and save the project: *File->Save Project*

Refer to the following page for a memory map of the sectors available.

CAUTION

The One Time Programmable Block (OTP) can only be programmed once.
The OTP block cannot be erased.

CAUTION

Do not press the SDFlash STOP button during the erase operation.

Pressing STOP will halt the CPU before the Erase algorithm completes. This can leave the Flash in a depleted state or result in unknown Code Security Module passwords and lock the device permanently.

Other conditions that can cause the CPU to halt prior to the completion of the Erase algorithm (e.g., power loss, device reset, PC crash, etc.) can result to the same problem described above.

Pressing STOP when executing the frequency toggle test described in section 9.2 is ok.

TMS320F2810 Sector Mask Value:

Bit 0 = Erase Sector A
Bit 1 = Erase Sector B
Bit 2 = Erase Sector C
Bit 3 = Erase Sector D
Bit 4 = Erase Sector E
Bit 5-15 = ignored

F2810 Sector Addresses:

ADDRESS RANGE	PROGRAM AND DATA SPACE
0x3E 8000 0x3E BFFF	Sector E, 16K x 16
0x3E C000 0x3E FFFF	Sector D, 16K x 16
0x3F 0000 0x3F 3FFF	Sector C, 16K x 16
0x3F 4000 0x3F 5FFF	Sector B, 8K x 16
0x3F 6000	Sector A, 8K x 16
0x3F 7FF6 0x3F 7FF7	Boot-to-Flash Entry Point (program branch instruction here)
0x3F 7FF8 0x3F 7FFF	Security Password (128-Bit)

TMS320F2810 Erase User Option 1 Examples:

0001 Erase only sector A
0003 Erase only sector A and sector B
001F Erase all sectors on an F2810 device

TMS320F2811/TMS320F2812 Sector Mask Value:

Bit 0 = Erase Sector A
Bit 1 = Erase Sector B
Bit 2 = Erase Sector C
Bit 3 = Erase Sector D
Bit 4 = Erase Sector E
Bit 5 = Erase Sector F
Bit 6 = Erase Sector G
Bit 7 = Erase Sector H
Bit 8 = Erase Sector I
Bit 9 = Erase Sector J
Bit 10-15 = ignored

TMS320F2811/TMS320F2812 Sector Addresses:

ADDRESS RANGE	PROGRAM AND DATA SPACE
0x3D 8000 0x3D 9FFF	Sector J, 8K x 16
0x3D A000 0x3D BFFF	Sector I, 8K x 16
0x3D C000 0x3D FFFF	Sector H, 16K x 16
0x3E 0000 0x3E 3FFF	Sector G, 16K x 16
0x3E 4000 0x3E 7FFF	Sector F, 16K x 16
0x3E 8000 0x3E BFFF	Sector E, 16K x 16
0x3E C000 0x3E FFFF	Sector D, 16K x 16
0x3F 0000 0x3F 3FFF	Sector C, 16K x 16
0x3F 4000 0x3F 5FFF	Sector B, 8K x 16
0x3F 6000	Sector A, 8K x 16
0x3F 7FF6 0x3F 7FF7	Boot-to-Flash Entry Point (program branch instruction here)
0x3F 7FF8 0x3F 7FFF	Security Password (128-Bit)

TMS320F2811/TMS320F2812 Erase User Option 1 Examples:

0001 Erase only sector A
0003 Erase only sector A and sector B
03FF Erase all sectors on an F2812 device

9.2. CPU Frequency and PLL Multiplier Configuration Toggle Test: Erase User Option 2

Erase User Option 2 turns on and off the frequency configuration toggle test as described in the *TMS320F2810, TMS320F2811 and TMS320F2812 Flash API (literature number SPRC125)* documentation. This test is used to confirm that the algorithms are properly configured for the CPU frequency and PLL multiplier. Refer to section 7 *f*or information on how to configure the algorithm for the frequency of your CPU.

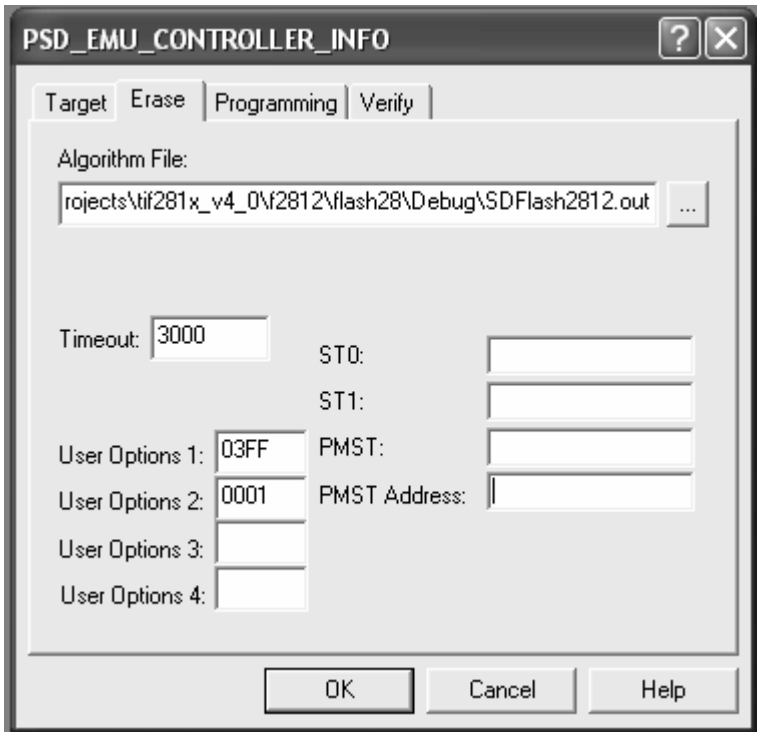
To run this test:

9.2.1. Open project settings: *Project -> Settings*

9.2.2. Click on the Erase tab

9.2.3. For User Options 2, enter the value that corresponds to the pin you want to toggle as shown in the table below.

Enter the hex digits only. Do not enter a 0x in front of the value.



Erase User Option 2	Pin Toggled (100µS cycle time)
blank	Test not run
0000	Test not run
0001	GPIOF14_XF
0002	GPIOA0_PWM1
0003	GPIOF4_SCITXDA
0004	GPIOG4_SCITXDB
0005	GPIOF12_MDXA
0006-FFFF	Test not run

CAUTION

Choose an appropriate pin for your system. Check your board design and board connections to be certain that the pin you have selected for toggling is not being driven by a source other than the DSP, or voltage contention can occur. Also, be certain that whatever the toggling pin is connected to in your system will not encounter difficulty when the pin is toggling (e.g, the device the pin is connected to should be powered-down, held in reset, etc.). A number of different pins are selectable above in order to avoid such problems,

9.2.4. Click Ok and save the project: *File->Save Project*

9.2.5. To begin the toggle test, start an erase operation using the *Device->Flash* menu.

With Erase User Option 2 set to 0001 - 0005, the toggle test will be performed instead of the erase operation and SDFlash will eventually timeout. The timeout period is as specified in the *Timeout* box, in seconds (e.g., 200 seconds). The user can click *Stop* to halt the toggle test sooner.

9.2.6. While the test runs, monitor the selected pin using an oscilloscope. If the algorithms are configured correctly for your CPU rate then the pin will toggle near 10kHz (100µS +/- 10µS cycle time). If the pin is toggling at a different rate, then the algorithms are <u>not</u> configured correctly. Follow the instructions detailed in section 7,

9.3. Configure Flash and OTP wait states: Verify User Option 1 and Verify User Option 2

Verify User Option 1 and User Option 2 allows you to specify wait states used for the flash and OTP memory during the verify operation. This option can be used to increase the speed of the verify operation or allow you to match the wait states used in your application if desired.

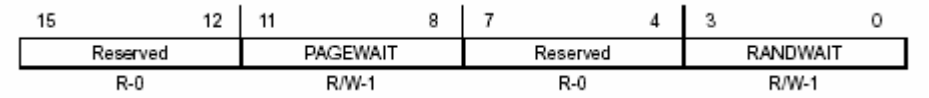
	Verify
User Option 1	Flash bank wait states
User Option 2	OTP wait states
User Option 3	Not used
User Option 4	Not used

CAUTION:

Frequency limits for both the Flash and OTP blocks are documented in the device Data Manual (SPRS174). Refer to the Data Manual for the minimum access time of the Flash and OTP memory. Using a wait state value that is too low will cause the verify operation to fail.

Verify User Option 1 specifies the Flash waitstate register (FBANKWAIT) contents:

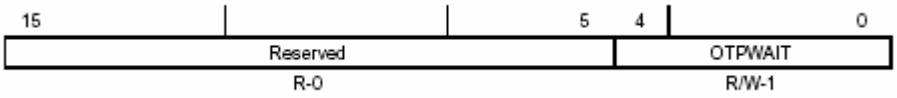
FBANKWAIT Register



The sample SDFlash project uses the default value of 0F0F. This value is also used if the option is left blank.

Verify User Option 2 specifies the OTP wait state register (FOTPWAIT) contents:

FOTPWAIT Register



The sample SDFlash project uses the default value of 001F. This value is also used if the option is left blank.

To set the wait state values:

9.3.1. Open the project settings: *Project -> Settings*

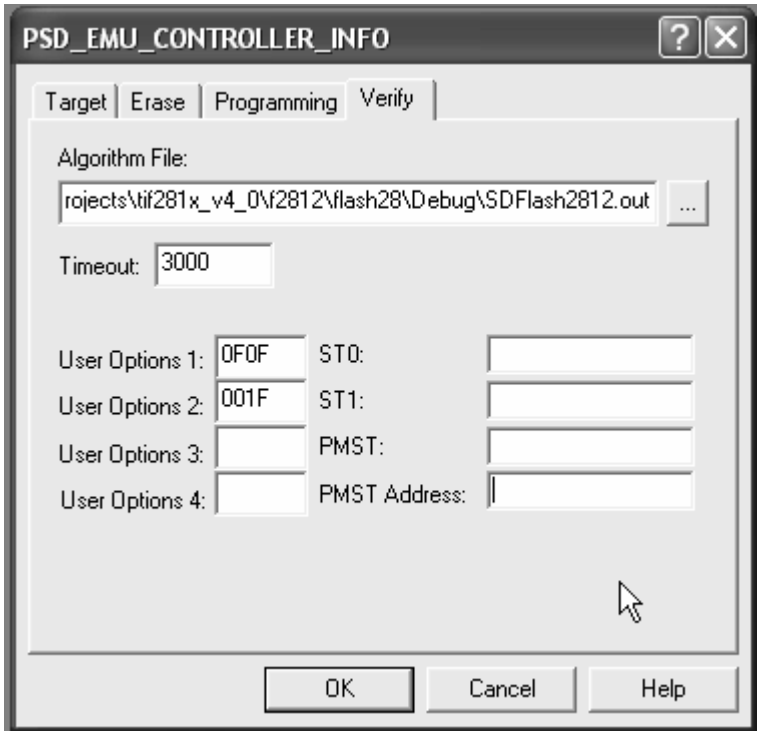
9.3.2. Click on the Verify tab

9.3.3. For User Options 1 enter the value for the FBANKWAIT register

9.3.4. For User Options 2 enter the value for the FOTPWAIT register

9.3.5. Select OK

9.3.6. Save the project: *File->Save Project*



10. Depletion Recovery Algorithm

This release of the SDFlash utilities includes a depletion recovery algorithm. This algorithm is called instead of erase for the following projects:

SampleF2810_DepRecover.sdp
SampleF2811_DepRecover.sdp, and
SampleF2812_DepRecover.sdp projects.

Note

These sample projects are setup for depletion recovery only and do not contain code to perform an erase, program or verify operation. Erase has been replaced by the depletion recovery routine. Program and verify will report a failure but do not perform any operation on the device.

How does depletion occur?

If the erase operation is interrupted and not allowed to complete the device may become depleted. When this happens, the device may then begin to fail to erase. All efforts should be taken to not stop the erase algorithm as this can also affect the CSM passwords. If the passwords are in an unknown state then the device cannot be recovered. If, however, the CSM passwords are known and the device can be unlocked, then the depletion recovery algorithm can be run to try and recover the part.

The depletion recovery algo looks for sectors that are in depletion and attempts to recover them. All sectors on the device are checked.

The current maximum timeout for the algorithm is approx 35 seconds per sector that is in depletion.
Typically only one sector would be in depletion unless erase has been called multiple times on multiple sectors without running to completion. If a longer timeout can be tolerated, the depletion recovery can be used multiple times.

There is no guarantee that this algorithm 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 it will take to recover. The Flash API erase function has been implemented to erase the flash in such a manner that it is not put into deep depletion. However, if the CPU is halted during an erase pulse for a long period of time the part can be put into a deep depletion that may not be recoverable in a time period that is acceptable.

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

11. Troubleshooting Tips

This is a list of solutions to potential failures when using the SDFlash utility. At this point you should have the most recent SDFlash revision installed and the proper flashing algorithms for the DSP you are using should be unzipped in to your SDFlash\myprojects folder. Prior to using the SDFlash, the SDConfig utility must successfully run and verify connection to your emulator and target DSP board. The SDConfig can be launched via the 'SDConfig' icon on your Windows desktop. If you need help with the SDConfig, please refer to the SDConfig.htm file included in your SDConfig directory.

It is recommended you begin by using one of the example projects included with the flashing algorithms associated with the DSP you are using. Before attempting to erase or program the flash, check the Project Settings of the flash project you are using. The project settings can be viewed by clicking the 'Project' menu, then click 'Settings'. Verify that each field that requires a directory path points to the location where that file is actually located. The fields that require directory paths in the Project Settings Window are:

Target Tab:	**Erase Tab:**	**Programming Tab:**	**Verify Tab:**
-Driver	-Algorithm File	-Algorithm File	-Algorithm File
-Board File		-Flash Data File	

If you have installed the SDFlash and Algorithms in a directory different from the default installation directory, the directory paths in the Project Settings will need to be changed.

11.1. DSP Reset Fails

- Check that the target DSP board is properly powered. Check that the DSP is properly clocked (check XCLKOUT using an oscilloscope).
- If using a stand alone JTAG emulator, make sure that it is properly powered.
- Make sure that the SDFlash Project is configured for the correct driver and port address. The SDFlash project must be configured with the correct driver to support the Spectrum Digital JTAG Emulator or eZdsp board you are using.

11.2. All Operations

- Refer to section 12 for a description of the API error codes that will be reported by SDFlash.
- The errors '**File Does Not Exist' or 'Load file failed'** may appear when attempting to open the Flash Panel. The file that the errors are referring to is the 'Flash Data File' located on the Programming tab of the Project Settings. Verify the directory path to the COFF file you wish to load is a valid path. (i.e. that the .out file exists in the directory entered.)
- A successful 'RESET' in SDFlash means the JTAG Emulator and Target board are properly powered and connected. However, take a moment to verify that all cables are connected securely and that the proper power is supplied to the emulator and target board.
- Older versions of the SDFlash algos are based on obsolete Flash APIs and should not be used.
- Run SDConfig to make sure your target and emulator are setup properly.
- Examine the project settings *Project->Settings* and make sure the path names to all files are correct as described in section 6.7.

- The flashing algorithms must be configured to multiply the DSP's input frequency appropriately and not exceed the DSP's maximum operational frequency. The algorithms found on the Spectrum Digital support sites are configured to support Spectrum Digital target boards. If you are using a custom board refer to the directions in this to properly configure the algorithms to support your target configuration.

- Make sure SDFlash can unlock the Code Security Module (CSM). An SDFlash error message of "ERR: Failed to initialize the algorithm" can indicate the CSM as the source of the problem. The algorithm attempts to unlock the CSM before an erase, program, or verify operation. By default, the password locations are assumed to be all erased (FFFFs). Refer to section 8 *Code Security Module (CSM) Password Considerations* if you have changed your passwords from the erased value.

  It is suggested that the CSM passwords be left erased (FFFFs) for initial development. Refer to the *TMS320x281x System Control and Interrupts Reference Guide* (literature number SPRU078) for details on the CSM operation.

- If required, configure the SDFlash algorithms for a custom CPU frequency and PLL multiplier. As supplied, the algorithms are configured for a CPU clock frequency (SYSCLKOUT) of 150MHz and to set the PLLCR register to 0x000A. If your hardware has other requirements you must configure the algorithms as described in section 7 Confirm proper configuration of the algorithms by running the CPU frequency and PLL multiplier configuration toggle test described in section 9.2 *CPU Frequency and PLL Multiplier Configuration Toggle Test: Erase User Option 2,*

- SDFlash should have full control of the device. That is, no user application should be running, no interrupts firing, and CCS should be shut down prior to using SDFlash.

- Make sure that the device has a clean VDD3VFL 3.3V voltage source. In addition VDD3VFL should remain connected, as it is required for read operations as well as programming.

- Check the part using Code Composer Studio (CCS). Using CCS, check the SARAM blocks and make sure that you can unlock the CSM (check by attempting to view the passwords in a memory window. If you view all 0x0000's, the CSM is still locked).

11.3. Erase Fails

- Make sure a valid sector mask is specified for User Option 1 for the erase operation. At least one sector must be specified. Do not use a 0x in front of the mask value – simply enter the hex number without the leading 0x. For example: correct 03FF incorrect 0x03FF

- If the configuration toggle test is selected via erase User Option 2, then the toggle test is executed in place of the erase algorithm. SDFlash will timeout and report an erase error in this case. Refer to section 9.2 *CPU Frequency and PLL Multiplier Configuration Toggle Test: Erase User Option 2* for more information.

- If you programmed new security passwords, the SDFlash algorithm may no longer be able to unlock the CSM using the default passwords. To avoid having to change the CSM passwords for different operations, it is suggested that the CSM passwords be left erased (FFFFs) for initial development. Refer to section 8 *Code Security Module (CSM) Password Considerations* if you have changed your passwords from the default (erased) value.

11.4. Programming Fails

- Check to make sure you are not programming a region of memory outside of the flash. To see what sections SDFlash will program, use the *View->Coff/Hex File Status* utility and look for sections labeled “load” outside of the flash memory region. It may be that the start address is within Flash or OTP but the end address is not.

Refer to the linker section in the *TMS320C28x Assembly Language Tools User's Guide*, literature #SPRU513, for more information on preparing your code for programming. Any loaded section that starts outside or ends outside of the flash region will cause programming to fail.

- Make sure you erased the sectors being programmed prior to programming them. Make sure the sector mask used for the erase function did not include the leading 0x.
  For example: correct 03FF incorrect 0x03FF

11.5. Verify Fails

- If you have programmed new security passwords, the algorithm may no longer be able to unlock the CSM using the default passwords. To avoid having to change the CSM passwords for different operations, it is suggested that the CSM passwords be left erased (FFFFs) for initial development. Refer to section 8 *Code Security Module (CSM) Password Considerations* if you have changed your passwords from the default (erased) value.

  Try increasing the wait states used for the flash during the verify operation as described in section 0 **Configure Flash and OTP wait states: Verify User Option 1 and Verify User Option 2***.*

11.6. Programmed Application Fails To Run

This section offers suggestions if you find that your programmed .out file is not executing properly.

- ***C281x C/C++ Header Files and Peripheral Examples in C*, literature # SPRC097**, is available for download from the TI website and provides small example programs for each of the peripherals on the F281x devices. Also included is an example flash project (in the examples\flash directory) that can be followed as an example program that runs from Flash.

- ***Running an Application from Internal Flash Memory on the TMS320F281x DSP*, literature # SPRA958**, is available for download from the TI website. This application note goes over the requirements for executing an application out of Flash memory. DSP/BIOS and non-BIOS applications are both included.

- Constant sections such as .switch, .const/.econst should be linked to page 0 (program) memory. SDFlash will not program sections linked to page 1 (data) memory. Note: this was not the not the case in *C281x C/C++ Header Files and Peripheral Examples in C V.58*, literature # SPRC097. If you use these examples, move the .econst section to page 0. Refer to the linker section in the *TMS320C28x Assembly Language Tools User's Guide*, literature #SPRU513, for more information on section allocation.

- If using the boot ROM (XMP/MC = low), then check that the boot mode option I/O pins of the device are set for "boot to flash" operation. Refer to *TMS320x281x Boot ROM Peripheral Reference Guide,* literature # SPRU095, for more information.

- The 3.3V flash power pin, VDD3VFL, should remain connected, as it is required for read operations as well as programming.

- The location 0x3F7FF6-0x3F7FF7 in flash should be programmed with a branch instruction to re-direct code flow from the boot ROM to the start of code in flash. The flash example program in *C281x C/C++ Header Files and Peripheral Examples in C*, literature # SPRC097, illustrates how to set this up.

- For programs with a long C initialization routine, the watchdog may reset before the main function is reached and the watchdog disabled or serviced. In this case a small assembly routine can be inserted in the code to disable the watchdog before the branch to _c_int00. The examples in *C281x C/C++ Header Files and Peripheral Examples in C*, literature # SPRC097**,** illustrate how to disable the watchdog before the c initialization phase.

12. API Error Codes

To communicate back to the calling application, the API returns the following status messages. These status values will be reported by SDFlash.

Status	Definition	Notes
0	STATUS_SUCCESS	Operation was successful.
10	STATUS_FAIL_CSM_LOCKED	The API function is unable to access the flash array due to a locked Code Security Module.
11	STATUS_FAIL_REVID_INVALID	The REVID is incorrect for this API. The V2.00 API is built for REVID greater or equal to silicon revision C.
12	STATUS_FAIL_ADDR_INVALID	An invalid address (outside of the flash or OTP bank) was passed to the API. For the program operation, this could be caused if the first address specified is outside of flash/OTP or the number of words to be programmed is such that the last address will be outside of flash/OTP. This error can be returned by the erase and depletion recovery functions if an invalid address is used for the pre-conditioning of the flash. In this case check that the .const section of the API is located in SARAM that can be accessed by the API. This section contains important information that is used by the erase and depletion recovery functions. In the case of this error, none of the values passed to the program function will be programmed. The flash status structure (FLASH_ST) is not updated with any information.
13	STATUS_FAIL_INCORRECT_PARTID	This error code is new as of V2.00 of the API. The expected PARTID did not match the device PARTID. This would indicate that the wrong API was used. For example, using a TMS320F2812 API on the TMS320F2808 device.

API Error codes continued…

Status	Definition	Notes
14	STATUS_FAIL_API_SILICON_MISMATCH	This error code is new as of V2.00 of the API. At the start of each API function, the content of a boot ROM location 0x3FFFB9 is checked to determine if the API is ok to execute on that silicon. In the future, TI can change the content of this boot ROM location if an API becomes obsolete. This will prevent an old API from executing on the new silicon. Version 2.00 of the API looks for the value 0xFFFF. On devices with an XINTF, the XINTFCNF2 register is saved and the boot ROM is enabled (if it was not already). After the check, the XINTFCNF2 register is restored. If this error code occurs, verify that that the proper API version is being used. Check the TMS320C2000 web site at http://www.ti.com/c2000.
Erase Specific Status Messages		
20	STATUS_FAIL_NO_SECTOR_SPECIFIED	Erase had nothing to do because no valid sectors were specified.
21	STATUS_FAIL_PRECONDITION	Erase operation failed because the clear portion of the pre-condition operation failed.
22	STATUS_FAIL_ERASE	Erase operation failed because the sector could not be erased with the maximum allowed number of pulses.
23	STATUS_FAIL_COMPACT	Erase operation failed because the post-conditioning (compaction) failed.
24	STATUS_FAIL_PRECOMPACT	This error code is new as of V2.00 of the API. Erase operation failed because the pre-compaction portion failed. The pre-compaction is applied to all sectors on the device. The FLASH_ST structure will return a fail address corresponding to the first sector fails this step.
Program Specific Status Messages		
30	STATUS_FAIL_PROGRAM	Program operation failed because one or more bits could not be programmed.
31	STATUS_FAIL_ZERO_BIT_ERROR	Program operation failed because one or more bits were already programmed (0) that should have been erased (1). If this happens it could be because the sector was not erased before attempting to program.
Verify Specific Status Messages		
40	STATUS_FAIL_VERIFY	The verify operation failed because one or more bits did not match the reference data. Try increasing the Flash or OTP wait states.

13. Detailed Directory Information

<CC base> is the default Code Composer Studio install directory c:\ti

Board File:
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\ccBrd028x.dat

Driver:
<CC base>\drivers\sdgo28x.dvr

SDFlash Projects Main Directory
<CC base>\specdig\SDFlash\myprojects

F2810/11/12 Algorithms version x
<CC base>\specdig\SDFlash\myprojects\tif28x_vx

F2810/12 Algorithms version x - documentation
<CC base>\specdig\SDFlash\myprojects\tif28x_vx\doc

F2812 Files

F2812 Algorithms and Sample Project Main Directory - version 4
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2812

F2812 SDFlash Sample Projects
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2812\SampleF2812eZdsp.sdp
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2812\SampleF2812.sdp
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2812\SampleF2812_DepRecover.sdp

F2812 Algorithm File for Erase, Program and Verify
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2812\flash28\Debug\SDFlash2812.out

F2812 Algorithm File for Depletion Recovery
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2812\flash28\Debug\SDFlash2812_DepRecover.out

F2812 CCS Project for creating Custom PLL, CPU or Password configuration
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2812\flash28\SDFlash2812.pjt

F2812 Sample Flash Image
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2812\image
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2812\image\debug\imageABCDEFGHIJ.out

F2811 Files

F2811 Algorithms and Sample Project Main Directory - version 4
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2811

F2811 SDFlash Sample Projects
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2811\SampleF2811.sdp
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2811\SampleF2811_DepRecover.sdp

F2811 Algorithm File for Erase, Program and Verify
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2811\flash28\Debug\SDFlash2811.out

F2811 Algorithm File for Depletion Recovery
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2811\flash28\Debug\SDFlash2811_DepRecover.out

F2811 files continued …

F2811 CCS Project for creating Custom PLL, CPU or Password configuration
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2811\flash28\SDFlash2811.pjt

F2811 Sample Flash Image
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2811\image
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2811\image\debug\imageABCDEFGHIJ.out

F2810 Files

F2810 Algorithms and Sample Project Main Directory - version 4
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2810

F2810 SDFlash Sample Projects
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2810\SampleF2810.sdp
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2810\SampleF2810_DepRecover.sdp

F2810 Algorithm File for Erase, Program and Verify
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2810\flash28\Debug\SDFlash2812.out

F2810 Algorithm File for Depletion Recovery
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2810\flash28\Debug\SDFlash2810_DepRecover.out

F2810 CCS Project for creating Custom PLL, CPU or Password configuration
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2810\flash28\SDFlash2810.pjt

F2810 Sample Flash Image
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2810\image
<CC base>\specdig\SDFlash\myprojects\tif281x_v4_1\f2812\image\debug\imageABCDE.out