

TMS320F2802x SDFlash JTAG Flash Programming Utilities

**This document describes the V1.0 release of the
F2802x SDFlash programming algorithms.**

These algorithms are based on the following API releases:

TMS320F2802x Flash API V1.00 in the 2802x boot ROM

**This download includes SDFlash algorithm files used to interface
the TMS320F2802x FLASH APIs to the SDFlash JTAG flash utility.**

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

**Document updated:
March 6, 2009**

SDFlash Algo Disclaimer

Please check the Spectrum Digital website for future updates.

The SDFlash Algorithm files in this download access the flash API included in the boot ROM of F2802x devices.

The SDFlash algorithm uses the following Flash Application Program Interface (Flash API) symbol library to access the API in the boot ROM:

- ❑ **Flash2802x_API_BootROMSymbols_V100.lib**

Symbol library for accessing the Flash API in the boot ROM

- ❑ **Flash2802x_API_V100a.lib**

Identical to the API in the boot ROM API except for a fix to the top level of the `Flash2802x_Program()` function. This software library is only needed for the program function.

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 TMS320F2802x testing.
- ❑ Improvements in algorithm performance and functionality.
- ❑ SARAM based flash API algorithms if required.

It is the user's responsibility to check for future updates to the SDFlash Algo Files based on any updates to the F2802x Flash API and to use the latest version available for their F2802x silicon.

Any SARAM based updates to the API or updates to the boot ROM symbol library will be posted on the Texas Instruments Inc website (www.ti.com) and can also be obtained by contacting Texas Instruments' support. Updates to the SDFlash algorithm files will also be posted on the Spectrum Digital website (www.spectrumdigital.com).

Contents:

| | | |
|-----|--|----|
| 1. | Revision History | 4 |
| 2. | Release Notes | 5 |
| 3. | Known Limitations | 5 |
| 4. | API Revision vs. Silicon Revision..... | 6 |
| 5. | SDFlash Overview..... | 7 |
| 6. | Quick Start Guide | 8 |
| 7. | PLL and CPU Clock Rate Configuration | 15 |
| 8. | Code Security Module (CSM) Password Considerations | 18 |
| 9. | SDFlash User Options..... | 19 |
| | 9.1. Specify Which Sectors to Erase: Erase User Option 1 | 19 |
| | 9.2. Configure Flash and OTP wait states: Verify User Option 1 and Verify User Option 2..... | 20 |
| | 9.3. CPU Frequency and PLL Multiplier Configuration Toggle Test: User Option 3 | 21 |
| 10. | Depletion Recovery Algorithm..... | 23 |
| 11. | Troubleshooting Tips..... | 24 |
| | 11.1. DSP Reset Fails..... | 24 |
| | 11.2. All Operations..... | 24 |
| | 11.3. Erase Fails | 25 |
| | 11.4. Programming Fails..... | 26 |
| | 11.5. Verify Fails | 26 |
| | 11.6. Programmed Application Fails To Run | 26 |
| 12. | API Error Codes | 28 |

1. Revision History

This is the revision history for the F2802x algorithm builds for SDFlash. Note that the SDFlash algorithm build version is separate from the SDFlash front-end utility version.

V1.0

- Initial release.

2. Release Notes

- ❑ The flash APIs used in this release are in the boot ROM of the F2802x devices.
- ❑ Some traditional programming utilities have separate operations for “Clear” and “Erase”. These two operations have been combined into one operation referred to only as “Erase”.
- ❑ **Note: The CSM will be permanently locked if the CSM password locations are loaded with all 0x0000 and the device is secured. During the combined “Erase” function, a sector clear 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 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.
- ❑ **Note: The F28022, F28024, F28025, and F28026 devices are not available during TMX sampling.**

3. Known Limitations

- ❑ SDFlash only programs sections linked to page 0 (i.e. program) memory. For F2802x users 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 it.
- ❑ 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. API Revision vs. Silicon Revision

The silicon revision can be determined by the lot trace code marked on the top of the package. Refer to the device specific errata for information on how to determine your device's silicon revision from the device symbolization. In addition, the silicon revision can be read from the REVID register located at address 0x883 as shown in the tables below.

SDFlash Algo release information for F2802x:

| Second Letter In Prefix of Trace Lot Code | REVID (Addr 0x883) | Boot ROM Rev | Flash API Rev | Obsolete SDFlash JTAG Algorithms | Recommended SDFlash Version | Flash API Boot ROM Symbol Library |
|---|--------------------|--------------|---|----------------------------------|-----------------------------|-----------------------------------|
| Blank | 0x0000 | V1a | V1.00 (in ROM) V1.00a SARAM library (for Flash2802x Program function only) | none | V1.0 | V1.0 |

Note:

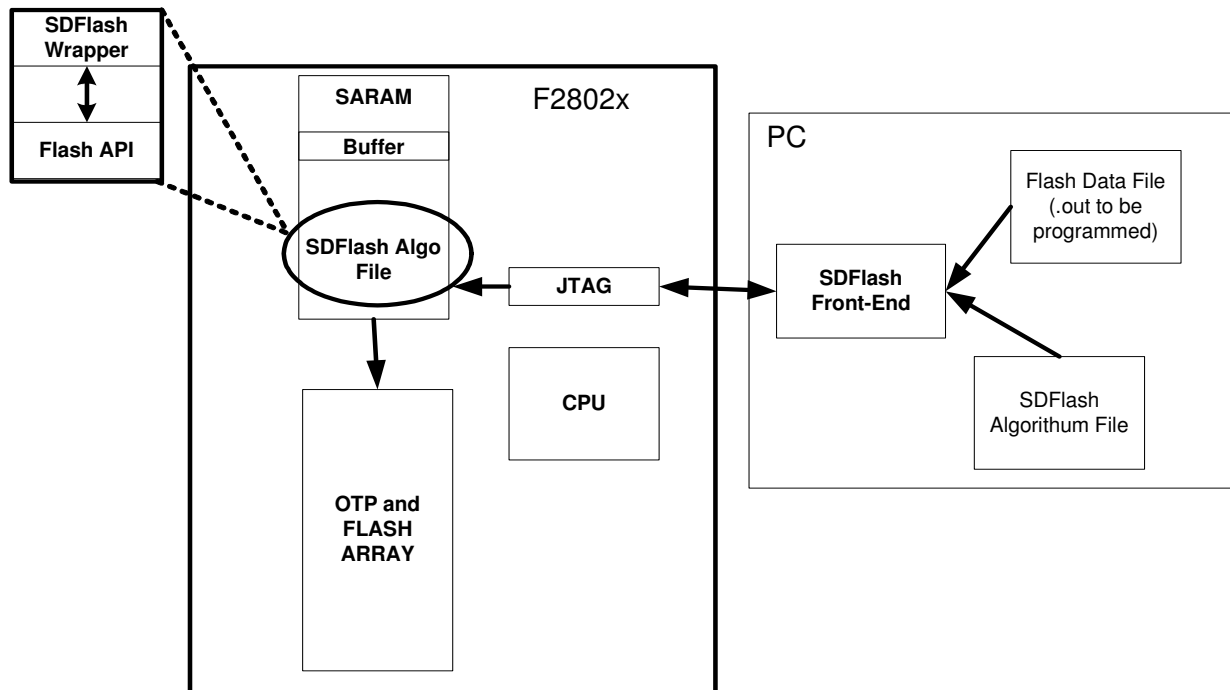
For future silicon revisions, TI anticipates that no functional changes will be required to these APIs (and hence no changes should be required to SDFlash algorithms).

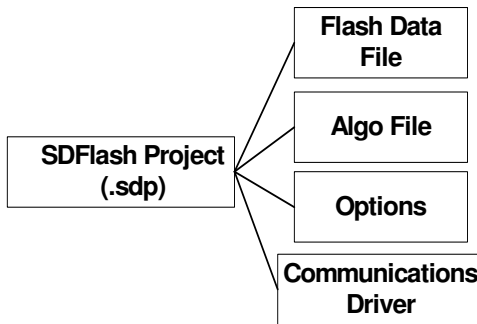
Should API changes occur that affect the programming of the flash, it is the user's responsibility to update any application that uses the API (programmers, embedded software, etc) to insure proper long-term operation of the flash. TI will test these APIs on future silicon revisions as soon as possible when such devices become available. Updates that only add features are not required.

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. For the F2802x, the SDFlash algorithm file consists of an SDFlash wrapper and the F2802x 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. 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 through the SDFlash GUI interface. A sample SDFlash project for each F2802x 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 and flash the F2802x device.

For example F2802x code that executes from the flash, refer to the following available from the TI website:

- Device appropriate C/C++ *Header Files and Peripheral Examples*
- Running an Application from Internal Memory on the TMS320F281x DSP (SPRA958)* This application note is written for the 281x family of DSPs but can be applied to the 2802x family as well.

6. Quick Start Guide

The following is a step-by-step guide for using the SDFlash utility to program your F2802x device.

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

| | |
|-----------|---|
| <CCS> | default Code Composer Studio install directory: " C:\CCStudio_v3.3" |
| <SDFlash> | default SDFlash directory <CCS>\specdig\SDFlash |

These directories may be different for your particular installation.

6.1. Make sure the target and emulator are setup properly:

- If you are using an XDS510/XDS510pp+/SPI515 emulation controller, use the SDConfig utility. SDConfig is part of the default emulation installation and will be installed in your <CCS>\specdig\sdconfig directory

6.2. Install the SDFlash flash support utility.

SDFlash is a generic utility supplied by Spectrum Digital Inc. to interface to user written 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 F2802x SDFlash algorithm files.

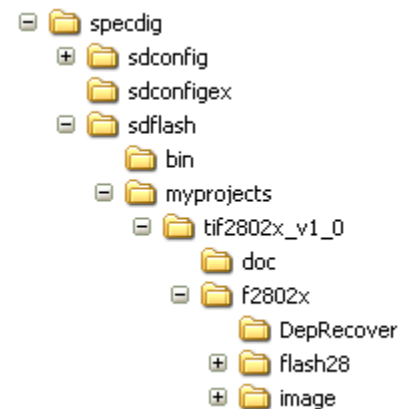
CAUTION

The current version of the algos is for TMX material for initial sampling.
Please check Spectrum Digital's website for future updates.

6.4. Unzip the F2802x SDFlash algorithm files into the myprojects subdirectory of SDFlash. For a typical install this will be the <CCS>\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.

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 have been included for use as project templates.

Using *File->Open Project* in SDFlash, browse to and load the appropriate sample SDFlash project. These sample projects program a flash image that is an address = data pattern. The CSM password is not programmed. For a typical installation, these files will be found in the following location:

SD 510PP/510PP+ emulators:

```
F28022: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28022.sdp
F28023: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28023.sdp
F28024: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28024.sdp
F28025: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28025.sdp
F28026: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28026.sdp
F28027: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28027.sdp
```

SD 510USB emulators:

```
F28022: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28022usb.sdp
F28023: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28023usb.sdp
F28024: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28024usb.sdp
F28025: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28025usb.sdp
F28026: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28026usb.sdp
F28027: <SDFlash>\myprojects\tif2802x_v1_0\f2802x\SampleF28027usb.sdp
```

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 V3.3 base directory "C:\CCStudio_v3.3". For example:

| | |
|----------------|--|
| <CCS> | default Code Composer V3.3 install directory: "C:\CCStudio_v3.3" |
| <SDFlash> | default is <CCS>\specdig\SDFlash |
| SDFlash binary | default is <CCS>\specdig\SDFlash |
| Flash projects | default is <CCS>\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 may 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*

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 can be found in the <CCS>\drivers\ directory.

default for PP emulators 510PP+/SPI515 etc: <CCS>\drivers\sdgo28x.dvr
 default for USB emulators (510PP+): <CCS>\drivers\sdgo28xusb.dvr

- Emulator Address/ID:* default is 378 for the 510pp, it is 510 for the USB driver. 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>\cc\bin\BrdDat directory. The default board file is <SDFlash>\myprojects\tif2802x_v1_0\ccBrd028x.dat
- Processor name:* default is cpu_0

Erase Tab: *Algorithm File:*

40MHz F2802x Devices:

<SDFlash>\myprojects\tif2802x_v1_0\flash28\config1\debug\SDFlashConfig1_Erase.out

60MHz F2802x Devices:

<SDFlash>\myprojects\tif2802x_v1_0\flash28\config1\debug\SDFlashConfig2_Erase.out

- Timeout:* leave as 3000
- User Options 1:* Erase sector mask default is 000F.
See section 9 for more information.
- For all other boxes the default is blank.*

Program Tab: *Algorithm File:*

40MHz F2802x Devices:

<SDFlash>\myprojects\tif2802x_v1_0\flash28\config1\debug\SDFlashConfig1_Program.out

60MHz F2802x Devices:

<SDFlash>\myprojects\tif2802x_v1_0\flash28\config1\debug\SDFlashConfig2_Program.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:

F28022: <SDFlash>\myprojects\tif2802x_v1_0\flash28\image\debug\image16k.out

F28023: <SDFlash>\myprojects\tif2802x_v1_0\flash28\image\debug\image32k.out

F28024: <SDFlash>\myprojects\tif2802x_v1_0\flash28\image\debug\image16k.out

F28025: <SDFlash>\myprojects\tif2802x_v1_0\flash28\image\debug\image32k.out

F28026: <SDFlash>\myprojects\tif2802x_v1_0\flash28\image\debug\image16k.out

F28027: <SDFlash>\myprojects\tif2802x_v1_0\flash28\image\debug\image32k.out

- Timeout:* leave as 3000
- For all other boxes the default is blank.*

Verify Tab: *Algorithm File:*

40MHz F2802x Devices:

<SDFlash>\myprojects\tif2802x_v1_0\flash28\config1\debug\SDFlashConfig1_Verify.out

60MHz F2802x Devices:

<SDFlash>\myprojects\tif2802x_v1_0\flash28\config1\debug\SDFlashConfig2_Verify.out

- Timeout:* leave as 3000
- User Options 1:* Flash wait states: default is 0F0F
- User Options 2:* OTP wait states: default is 001F
- For all other boxes the default is blank.*

- 6.6. **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 F2802x flash, use this project as a template and change the *Flash Data File* on the **Program Tab**.

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

The SDFlash routines use the internal oscillator by default.

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

Config1 : PLLCR = 0x0008 (PLL x8/2 mode), CPU frequency = $10 * 8 / 2 = 40\text{MHz}$

Config2 : PLLCR = 0x000C (PLL x12/2 mode), CPU frequency = $10 * 12 / 2 = 60\text{MHz}$

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 config 1 algorithm is configured for:

PLLCCR = 0x0008 (PLL x8/2 mode), and CPU frequency = $10 \cdot 8 / 2 = 40\text{MHz}$

As supplied, the config 2 algorithm is configured for:

PLLCCR = 0x000C (PLL x12/2 mode), and CPU frequency = $10 \cdot 12 / 2 = 60\text{MHz}$

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

- 6.8. **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.
- 6.9. **Reset the device: *Device->Reset*.** You will get a pass/fail message in the output window.
- 6.10. **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 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 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 9.3 is ok.

- 6.11. **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 6.9 and 6.10.
- 6.12. **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 F2802x 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 40MHz devices (28022, 28023, 28024 and 28025) algorithms are configured for the following (Config1):

- 10 MHz input clock
- x8 PLLCR (0x0008)
- ½ CLKINDIV enabled
- This results in a 40 MHz CPU clock (SYSCLKOUT)

The 60MHz devices (28026 and 28027) algorithms are configured for (Config2):

- 10 MHz input clock
- x12 PLLCR (0x000C)
- ½ CLKINDIV enabled
- This results in a 60 MHz CPU clock (SYSCLKOUT)

If your hardware requires any of the following:

- a different CPU rate (SYSCLKOUT) than stated above
- or requires a different PLLCR setting than stated above
- or requires the PLL to be disabled
- or requires that CLKINDIV is set (1/2 disabled in this case PLLCR must be set to 0x0000)

then you must modify and re-compile the flash programming algorithms for your system's specific requirements as described below:

The following steps describe how to compile a new SDFlash algorithm file for a custom frequency and PLL Control Register (PLLCR) setting.

- 7.1. **Using Code Composer Studio (CCS), load the projects for your device. There is one project per operation and all must be re-compiled for the new PLL configuration. These are the CCS projects used to build the SDFlash algorithm files. The projects are:**

Config 1: 40Mhz

Devices: 28022, 28023, 28024, 28025

Install Dir: <SDFlash>\myprojects\tif2802x_v1_0\config1\flash28

| | |
|-------------------------|-----------------------------|
| SDFlash2802x_Erase.pjt, | SDFlash2802x_Program.pjt, |
| SDFlash2802x_Verify.pjt | SDFlash2802x_DepRecover.pjt |

Config 2: 60Mhz

Devices: 28026, 28027

Install Dir: <SDFlash>\myprojects\tif2802x_v1_0\config2\flash28

| | |
|-------------------------|-----------------------------|
| SDFlash2802x_Erase.pjt, | SDFlash2802x_Program.pjt, |
| SDFlash2802x_Verify.pjt | SDFlash2802x_DepRecover.pjt |

- 7.2. **Specify the required PLLCR (PLL Control register) value.**

In CCS, open and modify SDFlash2802x_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:

```
// #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 Flash2802x_API_Config.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 60 Mhz. This corresponds to a 16.667 nS cycle time. Uncomment this entry and comment out all other entries:


```
#define CPU_RATE 16.667L // for a 60MHz CPU clock speed (SYSCLKOUT)
// #define CPU_RATE 20.000L // for a 50MHz CPU clock speed (SYSCLKOUT)
// #define CPU_RATE 25.000L // for a 40MHz CPU clock speed (SYSCLKOUT)
// #define CPU_RATE 33.333L // for a 30MHz CPU clock speed (SYSCLKOUT)
// #define CPU_RATE 41.667L // for a 24MHz 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 10 MHz and the device will be run at 60 MHz, then program the device at 60 MHz rather than 10 MHz or 20 MHz.

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

7.4. If your system requires any other custom setup such as modifying the CLKINDIV bit then:

Open the SDFlash2802x_Main.c file. The code that is executed on startup to setup the PLL is at the beginning of the main() function within this file. Modify this function as required for your clock requirements.

7.5. Rebuild each of the algorithms in CCS by selecting *Project->Rebuild all*.

CAUTION

It is important to rebuild all of the algorithm files: erase, program, verify and depletion recovery.

7.6. Exit Code Composer Studio

7.7. Run the CPU frequency and PLL multiplier configuration toggle test described in section 9.3 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 9.3 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 F2802x 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 *TMS320x2802x System Control and Interrupts Peripheral Reference Guide*, for details on the proper operation of the CSM.

The algorithm uses the CSM keys provided in the appropriate CSM key assembly file: (SDFlash28x_Erase_CsmKeys.asm, SDFlash28x_Program_CsmKeys.asm, or SDFlash28x_Verify_CsmKeys.asm) to unlock the CSM. Follow these steps to build an SDFlash algorithm file with a new set of CSM passwords.

Using Code Composer Studio (CCS), load the projects for your device. There is one project per operation and all must be re-compiled for the new PLL configuration. These are the CCS projects used to build the SDFlash algorithm files. The projects are:

As supplied, the example projects for 40Mhz devices use configuration 1:

```
<SDFlash>\myprojects\tif2802x_v1_0\config1\flash28\
    SDFlash2802x_Erase.pjt,      SDFlash2802x_Program.pjt,
    SDFlash2802x_Verify.pjt    SDFlash2802x_DepRecover.pjt
```

The example projects for 60Mhz devices use configuration 2:

```
<SDFlash>\myprojects\tif2802x_v1_0\config2\flash28\
    SDFlash2802x_Erase.pjt,      SDFlash2802x_Program.pjt,
    SDFlash2802x_Verify.pjt    SDFlash2802x_DepRecover.pjt
```

- 8.1. **In CCS, open the appropriate CSM key file: SDFlash28x_Erase_CsmKeys.asm, SDFlash28x_Program_CsmKeys.asm, or SDFlash28x_Verify_CsmKeys.asm. These files contain 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.

- 8.2. **Modify the passwords in the appropriate CSM key file to match those already programmed into the CSM password locations.**
- 8.3. **Rebuild the algorithms in CCS by selecting *Project->Rebuild All***
- 8.4. **Exit Code Composer Studio**

Note: Each step in the process Erase, Program, and Verify separately unlocks the CSM. If 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.

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 F2802x SDFlash algorithm V1.0.

| | Erase | Program | Verify |
|----------------------|---------------------------|---------------------------|---------------------------|
| User Option 1 | Sector Mask | Not used | Flash bank wait states |
| User Option 2 | Not used* | Not used | OTP wait states |
| User Option 3 | Run frequency toggle test | Run frequency toggle test | Run frequency toggle test |
| User Option 4 | Enable XCLKOUT | Enable XCLKOUT | Enable XCLKOUT |

* Note on 281x devices this was the frequency toggle test. This has been moved to the user option 3. The frequency toggle test is available under all operations since each operation algorithm file is a separate compile.

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.

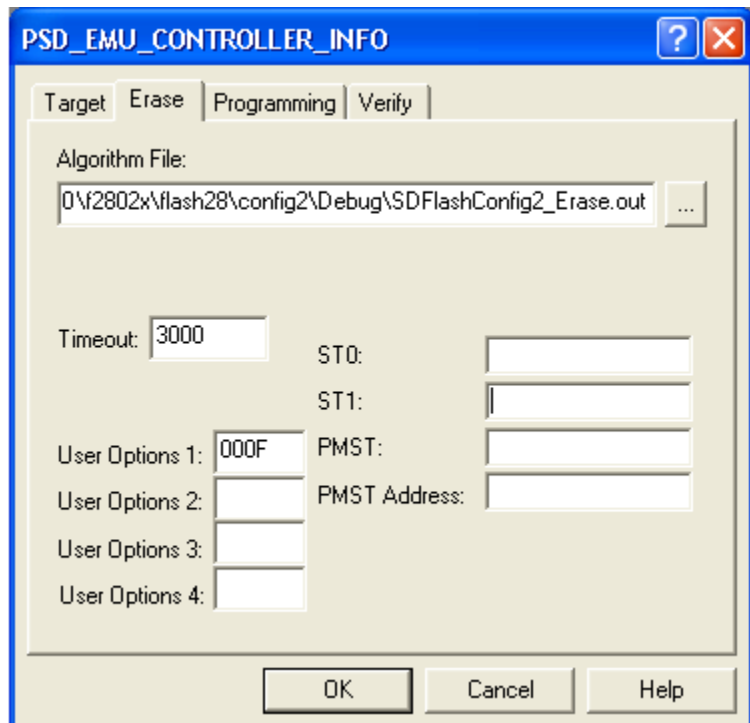
9.1. Specify Which Sectors to Erase: Erase User Option 1

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 4-15 = 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 = 000F would erase all sectors on a 2802x device.



Erase User Option 1 Examples:

| | |
|------|----------------------------------|
| 0001 | Erase only sector A |
| 0003 | Erase only sector A and sector B |
| 000F | Erase all sectors |

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

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. If you feel your device is in a depleted state (cannot be erased) then try the depletion recovery algo.

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 9.3 is ok.

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

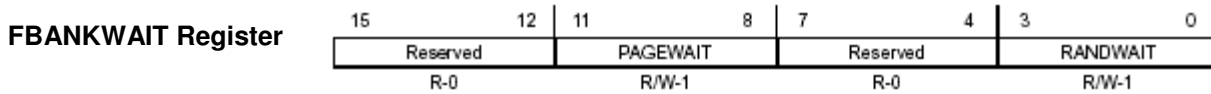
Verify User Option 1 and User Option 2 allows you to specify wait states used for the Flash and OTP memory during the verify operation. This option can be used to increase the speed of the verify operation or allow you to match the wait states used in your application if desired.

| | Verify |
|----------------------|---------------------------|
| User Option 1 | Flash bank wait states |
| User Option 2 | OTP wait states |
| User Option 3 | Toggle test (section 9.3) |
| User Option 4 | Not used |

CAUTION:

Frequency limits for both the Flash and OTP blocks are documented in the device Data Manual. 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:



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

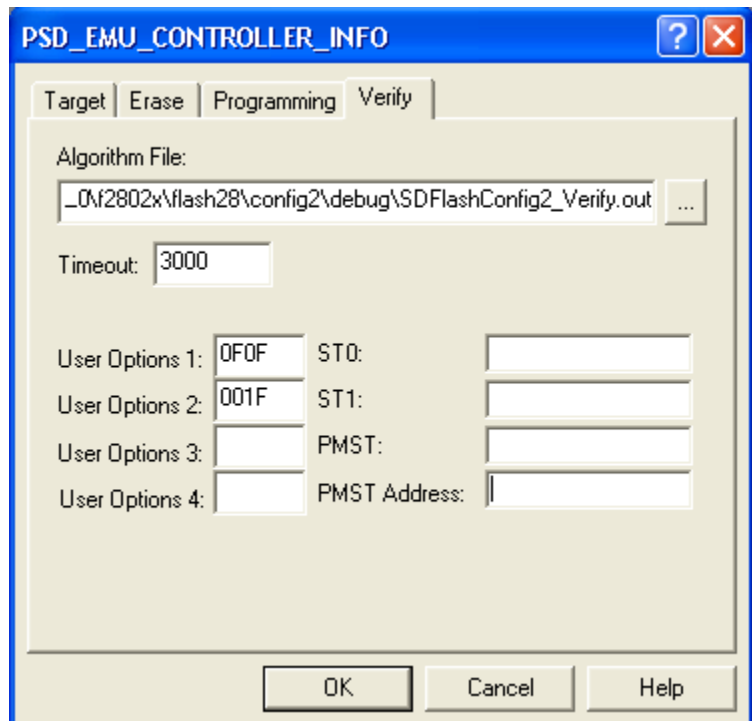
Verify User Option 2 specifies the OTP wait state register (FOTPWAIT) contents:



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:

- 9.2.1. Open the project settings: *Project -> Settings*
- 9.2.2. Click on the Verify tab
- 9.2.3. For User Options 1 enter the value for the FBANKWAIT register
- 9.2.4. For User Options 2 enter the value for the FOTPWAIT register
- 9.2.5. Select OK
- 9.2.6. Save the project: *File->Save Project*



9.3. CPU Frequency and PLL Multiplier Configuration Toggle Test: User Option 3

User Option 3 turns on and off the frequency configuration toggle test. This test is used to confirm that the algorithms are properly configured for the CPU frequency and PLL multiplier. Refer to section 7 for information on how to configure the algorithm for the frequency of your CPU.

The toggle test is available as user option 3 for erase, program and verify because the algorithm file for each of the operations is a separate compile. At initial setup, it is recommended that the frequency configuration of at least the erase and program operations are verified via the toggle test.

To run this test:

- 9.3.1. Open the project settings: *Project -> Settings*
- 9.3.2. Click on the appropriate tab: erase, program, verify

- 9.3.3. For “User Options 3”, enter the value that corresponds to the pin you want to toggle as shown in the table. Enter the value as hex digits only. Do not enter a 0x in front of the value.

For example, 0001 in erase User Options 3 indicates that the toggle test will be run on GPIO01 instead of the erase operation.

If you want to enable XCLKOUT on GPIO18 as well, then set User Options 4 to 0001. If User Options 4 is not 0001, then XCLKOUT will not be enabled.

NOTE: Not all GPIO pins are available on all devices.

| Erase, Program and Verify User Option 3 | Pin Toggled (100µS cycle time) |
|---|--------------------------------|
| blank | Test not run |
| 0000 | GPIO0 |
| 0001 | GPIO1 |
| 0002 | GPIO2 |
| 0003 | GPIO3 |
| 0004 | GPIO4 |
| 0005 | GPIO5... |
| | |
| 000A | GPIO10 |
| ... | ... |
| 0022 | GPIO34 |
| 0023-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.

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

- 9.3.5. To begin the toggle test, start a SDFlash operation using the *Device->Flash* menu.

Uncheck any operations you are not using. For example, if you want to check the frequency configuration of the program algorithm, uncheck the erase option and press start.

With User Option 3 set to 0000 - 0022, the toggle test will be performed instead of the specified operation and SDFlash will eventually timeout. The timeout period is as specified in the *Timeout* box, in seconds (e.g., 3000 seconds). The user can click *Stop* to halt the toggle test sooner.

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 (100 μ S +/- 10 μ S cycle time).

If the pin is toggling at a different rate, then the algorithms are not configured correctly. Follow the instructions detailed in section 7,

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 in the <sdfash>\myprojects\tif2802x_v1_0\2802x\DepRecover directory.

Note: These sample projects are setup for depletion recovery only and do not contain code to perform a program or verify operation. Thus, these operations 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 0.

- ❑ 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 *TMS320x2802x System Control and Interrupts Reference Guide* 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 9.3 *CPU Frequency and PLL Multiplier Configuration Toggle Test: User Option 3*
- ❑ 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 9.3 *CPU Frequency and PLL Multiplier Configuration Toggle Test: User Option 3* 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 9.2 *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.

- ❑ ***C2802x C/C++ Header Files and Peripheral Examples in C***, is available for download from the TI website and provides small example programs for each of the peripherals on the F2802x 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 TMS320F28xx 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 then check that the boot mode option I/O pins of the device are set for “boot to flash” operation. Refer to the device appropriate *Boot ROM Peripheral Reference Guide*, 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 the device appropriate *C/C++ Header Files and Peripheral Examples in C*, 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 the device appropriate *C/C++ Header Files and Peripheral Examples in C*, illustrate how to disable the watchdog before the c initialization phase.
- ❑ 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 the device appropriate *C/C++ Header Files and Peripheral Examples in C*, available for download from the TI website 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. |
| 12 | STATUS_FAIL_ADDR_INVALID | <p>An invalid address (outside of the flash or OTP bank) was passed to the API.</p> <p>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.</p> <p>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.</p> <p>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.</p> |
| 13 | STATUS_FAIL_INCORRECT_PARTID | <p>This error code is new as of V2.00 of the API.</p> <p>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.</p> |

| Status | Definition | Notes |
|---|----------------------------------|---|
| 14 | STATUS_FAIL_API_SILICON_MISMATCH | <p>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.</p> <p>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.</p> |
| 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 | 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. |
| SDFlash Specific Status Messages | | |
| 144 | Missing clock detect bit set | <p>This is a F2802x SDFlash specific error code. It is issued by the SDFlash interface function, not the flash API.</p> <p>This error code indicates that the MCLKSTS bit is set within the PLLSTS register. This bit indicates that the external clock has at some point gone missing. When this is the case, the flash API should not be executed as it will be at the wrong frequency.</p> |