# TMS320C2834x
# External Interface (XINTF) Flash Programming Utilities

**SDFlash XINTF Flash Programming Algo Version 1.0**

## This download includes algorithm files that can be used with SDFlash to program a flash memory device (AT49BV802D or similar) connected to the XINTF of a C2834x device.

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

**Document updated:**
**August 5, 2009**

# Contents:

# 1. Revision History

**Version 1.0**

❑ This version is the first release of the TMS320C2834x SDFlash XINTF Programming Utilities.

# 2. Release Notes

❑ This release of the SDFlash C2834x XINTF Flash Programming Algo is based on utility functions written to write and read from an external flash memory device through the XINTF of the C2834x device. The source code for the utility functions is provided as-is.

❑ 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 device JTAG port. During programming, SDFlash will have complete control of the device. No user application code can be running in parallel.

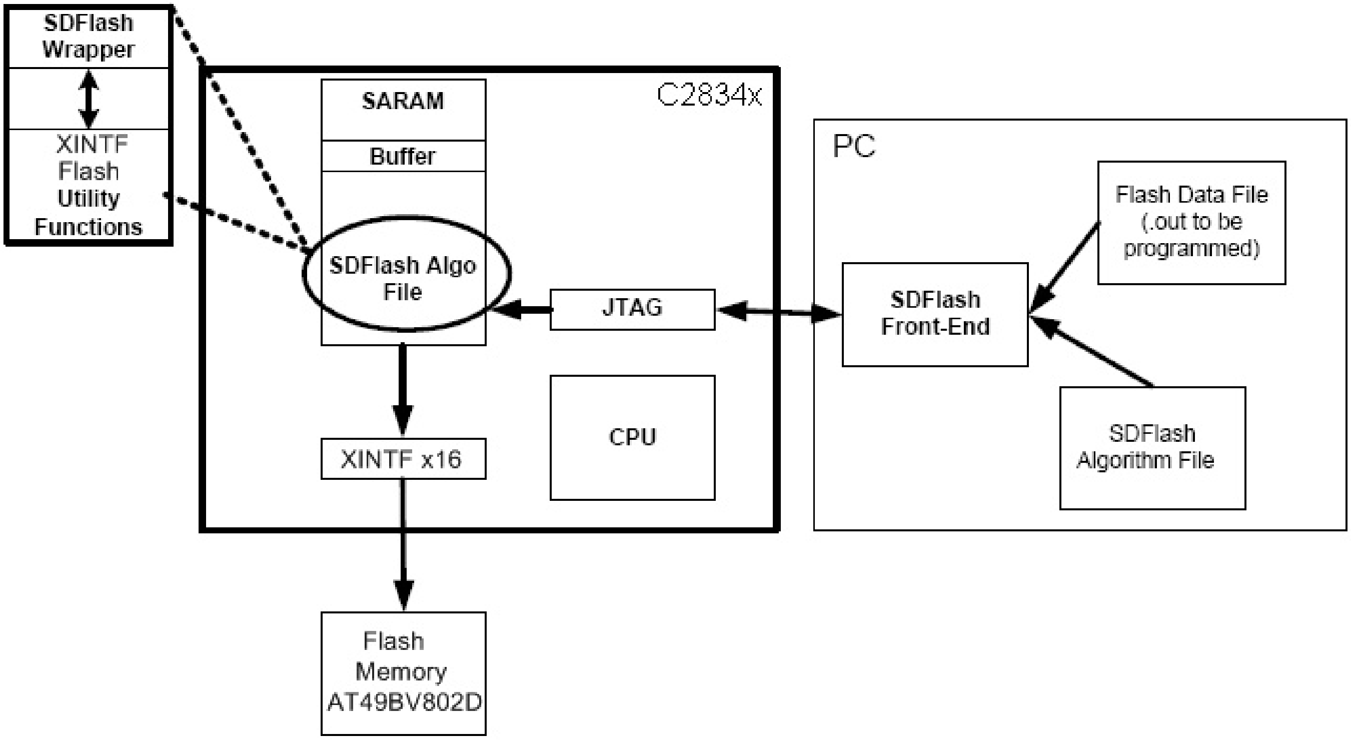
# 3. Known Limitations

❑ Support is currently provided for the Atmel AT49BV802D 8-megabit flash memory. The utility functions can be modified by the customer to support other flash memories if required.

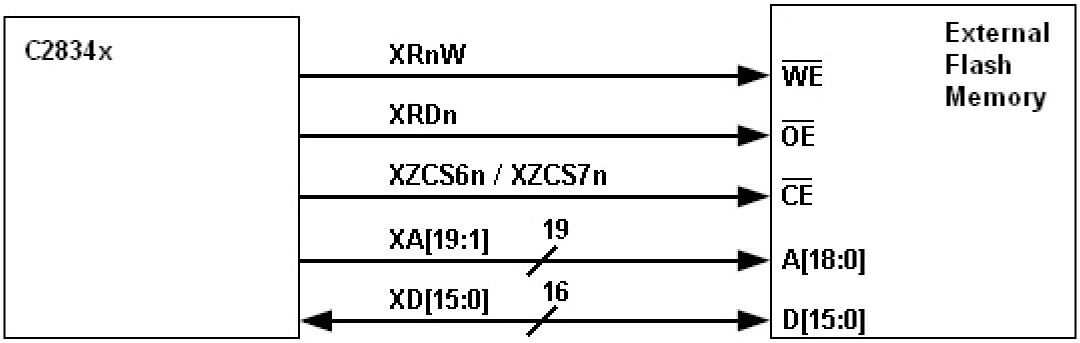
# 4. SDFlash Overview

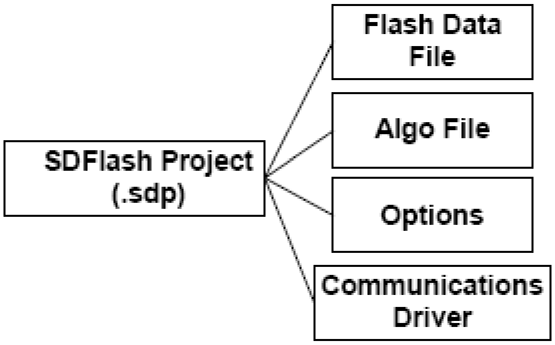
SDFlash is a generic front-end application owned by Spectrum Digital Inc (www.spectrumdigital.com). This application provides a generic interface to the JTAG communications channel that can be used to support programming of nonvolatile memory.

In order to perform the required operations, SDFlash downloads an algorithm file onto the C2834x device. This algorithm file is an executable (.out) file for the C2834x device. The SDFlash algorithm file consists of an SDFlash wrapper and C2834x-to-AT49BV802D flash programming library functions (for reading from, writing to, and erasing external memory through the XINTF). 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 C2834x-to-AT49BV802D flash programming utility functions to perform operations on the external flash memory. 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.



Ensure that your development board has the required hardware connections and circuitry for your flash memory chip. The flash memory should be connected to XINTF zone 6 and the boot mode pins set to “Jump to XINTFx16” mode. The connection between the C2834x device and the flash memory chip are shown below:





For SDFlash to program the external flash memory, it must have the following information:

- ❑ Location of the XINTF flash algorithm file
- ❑ Location of the flash data file – that is the data (.out file) to program into the flash device.
- ❑ 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 interface. A sample SDFlash project has been provided in this download.

Section 5 of this document will guide you through the SDFlash setup and show you how to use the provided sample SDFlash project to create your own project to program the external flash memory through the XINTF.

# 5. Quick Start Guide

The following is a step-by-step guide for using the SDFlash utility to program an external flash chip via the XINTF on your C2834x device.

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

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

These directories may be different for your particular installation.

**5.1. Make sure the target and emulator are setup properly:**
If you are using an XDS510USB/XDS510/XDS510pp+/SPI515 emulation controller, use the SDConfig utility. SDConfig is part of the default emulation installation and will be installed in your *<CCS base>\specdig\sdconfig* directory

**5.2. Install the SDFlash flash support utility.**
A version of SDFlash is included with your Code Composer Studio install. SDFlash will typically be installed in your <CCS base>\specdig\SDFlash directory.

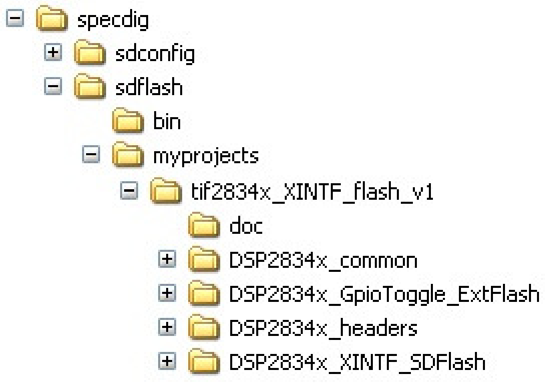
SDFlash is a generic utility supplied by Spectrum Digital Inc. to interface to user written algorithms. In this case, Texas Instruments Inc has supplied the algorithm file. Users should check the SD website for updates to this utility.

**5.3. Download the latest C2834x SDFlash XINTF external flash programming algorithm files.**
Future releases and upgrades will be available for download from the Spectrum Digital website in the *Support Utilities->SdFlashAlgo's*
section.

**5.4. Unzip the XINTF external flash programming 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.



```
⊟ specdig
    ⊞ sdconfig
    ⊟ sdflash
        bin
        ⊟ myprojects
            ⊟ tif2834x_XINTF_flash_v1
                doc
                ⊞ DSP2834x_common
                ⊞ DSP2834x_GpioToggle_ExtFlash
                ⊞ DSP2834x_headers
                ⊞ DSP2834x_XINTF_SDFlash
```

### 5.5. Make any required changes to the XINTF Flash Programming Algorithms.

The flash programming algorithms configure the C2834x device with the following configuration parameters:

| | |
|---|---|
| PLLCR = 19 | SYSCLKOUT = 20*(19+1)/2 = 200 MHz (works on all C2834x devices) |
| XTIMCLK = 1 | XTIMCLK = SYCLKOUT/2 = 200 MHz/2 = 100MHz |
| X2TIMING = 1 | Number of wait states doubled |
| XRDLEAD = 3 | Read lead time = 3*2 + 1 = 7 XTIMCLK cycles = 70ns |
| XRDACTIVE = 3 | Read active time = 3*2 + 1 = 7 XTIMCLK cycles = 70ns |
| XRDTRAIL = 3 | Read trail time = 3*2 + 1 = 7XTIMCLK cycles = 70ns |
| XWRLEAD = 1 | Write lead time = 1*2 + 1 = 3 XTIMCLK cycles = 30ns |
| XWRACTIVE = 3 | Write active time = 3*2 + 1 = 7 XTIMCLK cycles = 70ns |
| XWRTRAIL = 2 | Write trail time = 2*2 + 1 = 5 XTIMCLK cycles = 50ns |
| XSIZE = 3 | 16-bit mode |
| USEREADY = 0 | fixed length accesses |

This configuration and algorithms have been tested on the C28346 with an Atmel AT49BV802D flash memory device. If your system has other requirements you must modify the algorithms for your system.

To change the algorithms, load the SDFlash algo project into Code Composer Studio. This project is located at:

*<SDFlash base>\myprojects\tif2834x_XINTF_flash_v1\DSP2834x_XINTF_SDFlash\ DSP2834x_XINTF_SDFlash.pjt*

To modify the XINTF timing values, edit the file: DSP2834x_*AT49BV802D.c*, which is located in

*<SDFlash base>\myprojects\tif2834x_XINTF_flash_v1\DSP2834x_common\source*

After your modifications, re-build the algorithm file.

### 5.6. Make sure Code Composer Studio is not running and run SDFlash.

### 5.7. Load the supplied SDFlash sample project.

SDFlash uses .sdp project files to store information required to erase a device and program an .out file into a device. A sample 2834x project has been included in *tif2834x_XINTF_flash_v1.zip* for use as a project template.

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

2834x XDS510USB:
*<SDFlash base>\myprojects\tif2834x_XINTF_flash_v1\Sample_DSP2834x_XINTF_ExtFlash.sdp*

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

If you installed 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:\ CCStudio_v3.3" for Code Composer Studio v3.3 For example:

| | |
|---|---|
| *<CCS base>* | default Code Composer install directory: "*c:\CCStudio_v3.3*" |
| *<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>* |

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

**Target Tab:**

- □ *Driver:* This is the Code Composer Studio$^{TM}$ emulation driver (*.dvr) file that is used to communicate with the target. The driver files can be found in the *<CCS base>\drivers\* directory.
  default for XDS510USB: <CCS base>\drivers\sdgo28xusb.dvr
- □ *Emulator Address/ID:*
  default for USB: 510
- □ *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\ tif2834x_XINTF_flash_v1\ccBrd028x.dat*
- □ *Processor name:* default is cpu_0

**Erase Tab:**

- □ *Algorithm File:*
  *<SDFlash base>\myprojects\tif2834x_XINTF_flash_v1\DSP2834x_XINTF_SDFlash\Debug\DSP2834x_XINTF_SDFlash.out*
- □ *Timeout:* leave as 200 or higher
- □ *For all other boxes the default is blank.*

**Program Tab:**

- □ *Algorithm File:*
  *<SDFlash base>\myprojects\tif2834x_XINTF_flash_v1\DSP2834x_XINTF_SDFlash\Debug\DSP2834x_XINTF_SDFlash.out*
- □ *Flash Data File:* This is the .out file that you want to program into the external flash memory.
  For the example project, this file can be found at:
  *<SDFlash base>\myprojects\tif2834x_XINTF_flash_v1\DSP2834x_GpioToggle_ExtFlash\Debug\GpioToggle.out*
  The example application boots from XINTF and then toggles the GPIO pin 0.
  *Timeout:* leave as 200 or higher
- □ *For all other boxes the default is blank.*

**Verify Tab:**

- □ *Algorithm File:*
  *<SDFlash base>\myprojects\tif2834x_XINTF_flash_v1\DSP2834x_XINTF_SDFlash\Debug\DSP2834x_XINTF_SDFlash.out*
- □ *Timeout:* leave as 200 or higher
- □ *For all other boxes the default is blank.*

### 5.9. 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.

Should you want to program a different .out file into the flash, use this project as a template and change the *Flash Data File* on the **Program Tab.**

### 5.10. Reset the device: *Device->Reset*.

You will get a pass/fail message in the output window.

### 5.11. 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.

### 5.12. 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 5.10 and 5.11.

### 5.13. Boot the code from XINTFx16

Power cycle or hardware reset the device. The boot mode options should be set to “Jump to XINTFx16”. The boot loader will branch to address 0x00100000 and your code will execute directly through the XINTF. If you are using the example project, you should see GPIO0 toggle. Consult the 2834x boot ROM guide for more information on boot loader operation (SPRUFN5). Use the linker file in the example project (28346_Flash_AT49BV802D_lnk.cmd) as a template for linking code to external flash. For further support on configuring your application to run from nonvolatile memory, consult “Running an Application from Internal Flash Memory on the TMS320F28xxx DSP” (SPRA958H).

# 6. Modifying the Algorithm Files

As-is, the SDFlash2834x XINTF Flash Programmer Utility algorithm files supports the ATMEL AT49BV802D only.

## 6.1. Modifying the algorithms for a different flash memory device

The “DSP2834x_XINTF_SDFlash.c” file contains the functions called directly by the SDFlash application. This file should require minimal changes to allow the use of a different flash memory chip. All calls to the drivers of the external flash memory are through the interface functions defined in “DSP2834x_ExtFlashAPI.h”. Therefore, to switch to a different flash memory, write drivers for your memory which implement the functions in “DSP2834x_ExtFlashAPI.h”. See the provided file “DSP2834x_AT49BV802D.c” for an example driver that implements these functions.

## 6.2. Modifying the AT49BV802D drivers to suit your application

As previously mentioned, all calls to the AT49BV802D drivers from the SDFlash application come through the functions defined in “DSP2834x_ExtFlashAPI.h”. Therefore, you may modify the AT49BV802D drivers for use in your application without breaking SDFlash as long as the functionality specified in the external flash API is implemented in the correct functions.