Application Report
SPRACI5–December 2018

Updating Firmware on Security Enabled TMS320F2837xx or TMS320F2807x Devices

David Foley, Pramod Prabhakara, Salvatore Pezzino, Vivek Singh

ABSTRACT

This application report briefly discusses the basic DCSM protections and how to update the firmware in the Flash when the DCSM protections are enabled.

Contents

List of Figures

List of Tables

Trademarks

Delfino, Piccolo, Code Composer Studio are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Introduction

The Delfino™ TMS320F2837xD, Delfino TMS320F2837xS, and Piccolo™ TMS320F2807x are powerful 32-bit floating-point microcontroller unit (MCU) designed for advanced closed-loop control applications such as industrial motor drives; solar inverters and digital power; electrical vehicles and transportation; and sensing and signal processing. While the hardware efficiency of these microcontrollers enables powerful applications, it is the code that creates the specific and unique application. It is important to protect your code against copycats that leverage your time and expertise for their own gain. The Dual Code Security Module (DCSM) built into these devices helps you to create a secure solution against those that would attempt to clone your code. One of those protections is limiting the ability to update the Flash contents on the device to an authorized source.

The simple concept of the DCSM separates code into three areas: an unsecure space and two secure zones. Each of the secure zones is enabled with a password and security policy programmed into the one-time programmable (OTP) area of the Flash. The policy governs, among other items, which flash sectors and RAM blocks (memory) belong to that zone and whether or not the information contained in that memory is marked execute-only. Any code or data residing in a memory is said to belong to the same zone as the memory. If a memory belonging to a locked zone is marked execute-only, code can be executed from that memory, but not read (or written, if RAM). If a memory belonging to a locked zone is not marked execute-only, code belonging to that zone can also read it (and write to it, if RAM). Regardless of the execute-only setting, a flash sector belonging to a zone can be reprogrammed by code belonging to the same zone. Code running from the other zone or unsecured space cannot read, write, or reprogram a zone's memory while that zone is locked. Table 1 shows the permissible memory accesses.

Table 1. Permissible Memory Accesses

	Access Permission (R/W/X) [1]				
Code Running From	**Zone 1**	**Zone 1 EXEONLY**	**Zone 2**	**Zone 2 EXEONLY**	**Unsecure**
Zone 1	R/W/X	-/-/X	-/-/X	-/-/X	R/W/X
Zone 2	-/-/X	-/-/X	R/W/X	-/-/X	R/W/X
Unsecure	-/-/X	-/-/X	-/-/X	-/-/X	R/W/X

(1) Key: R = Read; W = Write; X = Execute

NOTE: Code executing from either secure memory or unsecured memory can branch and begin executing from either secure memory or unsecure memory.

In the TMS320F2837xD devices, there are two CPU subsystems with local securable memory. Each contains an independent DCSM. In the TMS320F2837xS and TMS320F2807x devices, there is one CPU subsystem and corresponding DCSM. For illustration, this document uses the TMS320F28379S as an example. The other devices are easily extensible. The TMS320F28379S memory block diagram is shown from a security point of view in Figure 1. Some memories are hardcoded to a security zone or unsecure space; others are selectable via the settings programmed into the Flash OTP.

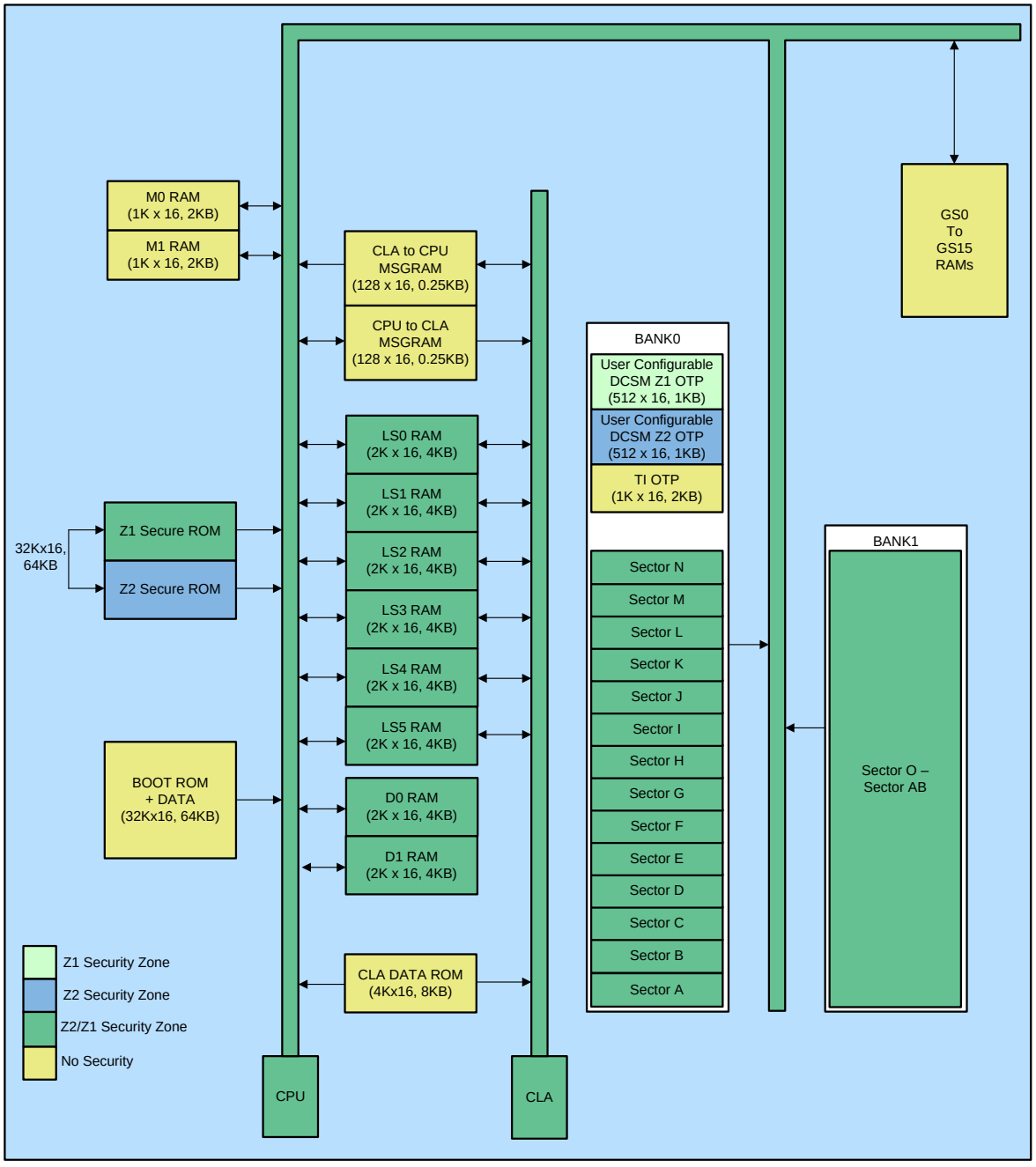

Figure 1. TMS320F28379S Security View Block Diagram

When a zone is locked, reprogramming the flash sectors belonging to that zone is protected. Code running from the other zone or unsecure space cannot erase or program a sector belonging to a secure zone. Once a zone is unlocked using password match flow (PMF), there are two methods that can be used to reprogram it during a firmware update. The first is by simply writing the password for that zone into its KEY register, thereby unlocking the zone. Once unlocked, any sector belonging to that zone can be reprogrammed by any code on the device, meaning basic flash programming utilities can be used. Since this approach unlocks the entire zone for both reprogramming and even reading and writing, it should only be employed in a trusted environment. For guidance on programming an unlocked or unsecure devices, see the *Serial Flash Programming of C2000 Microcontrollers*.

The other method available is to embed the flash programming utilities into the zone's flash during initial programming. This code can then be copied to a RAM block owned by that zone during a firmware update and run from within the zone. Since code running from within a zone can reprogram any Flash memory in that zone, the password for that zone need not be written into the KEY register and the zone will stay locked in the view of the other zone and unsecure space. This application report explains the above process of programming flash code while the zone is locked. Also there is an example [(1)] code provided in C2000Ware for this method.

2 Programming a Secure Flash Sector

As stated earlier, programming a locked flash sector can be done by two methods out of which one is to simply unlock the zone by writing the password to the zone's KEY register. This is acceptable in a trusted environment.

The second method would be to integrate the Flash API into the zone's flash. This Flash API library is then used during a firmware update by copying it to and running it from a RAM block owned by that zone, thereby, enabling it to erase and program a flash sector belonging to that zone without actually unlocking the zone.

There are two important points that need to be taken care of while accomplishing this:

- Usage of the memcpy function from the rts library. To include the rts library in your Code Composer Studio™ project, right click on the project and choose Properties, then Build → C2000 Linker → File Search Path and include the library (for example, "rts2800_fpu32.lib").
- Integration of Flash API and the flash programming code into the ramfuncs linker section

NOTE: To program the security settings in the User OTP, the zone must be unlocked. This application report does not cover programming the security settings.

2.1 Usage of memcpy Function From the RTS Library

The RTS library provides a memory copy (memcpy) function that enables you to copy a section of code from flash to RAM. This function can be used to load the Flash API, along with other custom functions that are listed further, from flash to RAM. Following is a snippet of an example usage of this function.

```
memcpy(&RamfuncsRunStart, &RamfuncsLoadStart, (size_t)&RamfuncsLoadSize);
```

The RamfuncsLoadStart, RamfuncsLoadSize, and RamfuncsRunStart are symbols that need to be created in the linker command file. RamfuncsLoadSize and RamfuncsLoadStart define the size and the source starting address in flash for the copy function. RamfuncsRunStart is the destination RAM address where the ramfuncs section is to be copied.

Since the memcpy function needs to copy content from secure flash to secure RAM, it must be placed in a secure memory location as shown in the linker command (.cmd) snippet below:

```
.memcpy     : > FLASHN PAGE = 0
{
--library=rts2800_fpu32.lib<memcpy.obj> (.text)
}
```

2.2 Integration of Flash Programming Code Into RAM Functions Section

The Flash API can be integrated into a CCS project by using the Flash API library available in C2000Ware . Once C2000Ware is installed, the .lib can be found in this directory:

```
<C2000Ware_Version>\libraries\flash_api\<device_name>\lib\*
```

Example:
C:\ti\c2000\C2000Ware_1_00_05_00\libraries\flash_api\f2837xs\lib\F021_API_F2837xS_FPU32.lib

Include this .lib file into your CCS project by dragging the file from Windows Explorer.

(1) The example code assumes flash sectors D and N as well as RAM blocks LS3 and LS5 belong to the same security zone as the flash sector targeted for programming.

Next, force the programming API into a linker section called ".TI.ramfunc", which will hold all the functions that need to be executed from RAM's as per the application requirements. This is done by adding the following to your linker command file (.cmd):

```
GROUP
{
   .TI.ramfunc
   { -l F021_API_F2837xS_FPU32.lib}

} LOAD = FLASHD,
RUN  = RAMLS03,
LOAD_START(_RamfuncsLoadStart),
LOAD_SIZE(_RamfuncsLoadSize),
LOAD_END(_RamfuncsLoadEnd),
RUN_START(_RamfuncsRunStart),
RUN_SIZE(_RamfuncsRunSize),
RUN_END(_RamfuncsRunEnd),
PAGE = 0
```

The linker will link the load address (the location it will be placed when loaded, or programmed) of the Flash API library to a secure flash sector (FLASHD) and link the run address (the locations used for function calls and other run time address requirements) to secure RAM (RAMLS03). It then populates the exact address/size values into the following linker variables for use in the memcpy function:

- RamfuncsLoadStart
- RamfuncsLoadSize
- RamfuncsLoadEnd
- RamfuncsRunStart
- RamfuncsRunSize
- RamfuncsRunEnd

The other custom flash programming code (for example, user code that calls the Flash API functions) can also be a part of ramfuncs with the help of #pragma directives as indicated below.

```
//*****************************************************************************
//   Example_CallFlashAPI
//
//   This function will interface to the flash API.
//   Flash API functions used in this function are executed from RAM
//*****************************************************************************
#pragma CODE_SECTION(Example_CallFlashAPI, "ramfuncs");
```

3 Software Flow Chart

Figure 2 shows a software flowchart followed by the example code found in its C2000Ware folder (C2000Ware_Version\device_support\f2837xs\examples). The example code can be referenced to develop your secure firmware upgrade application.

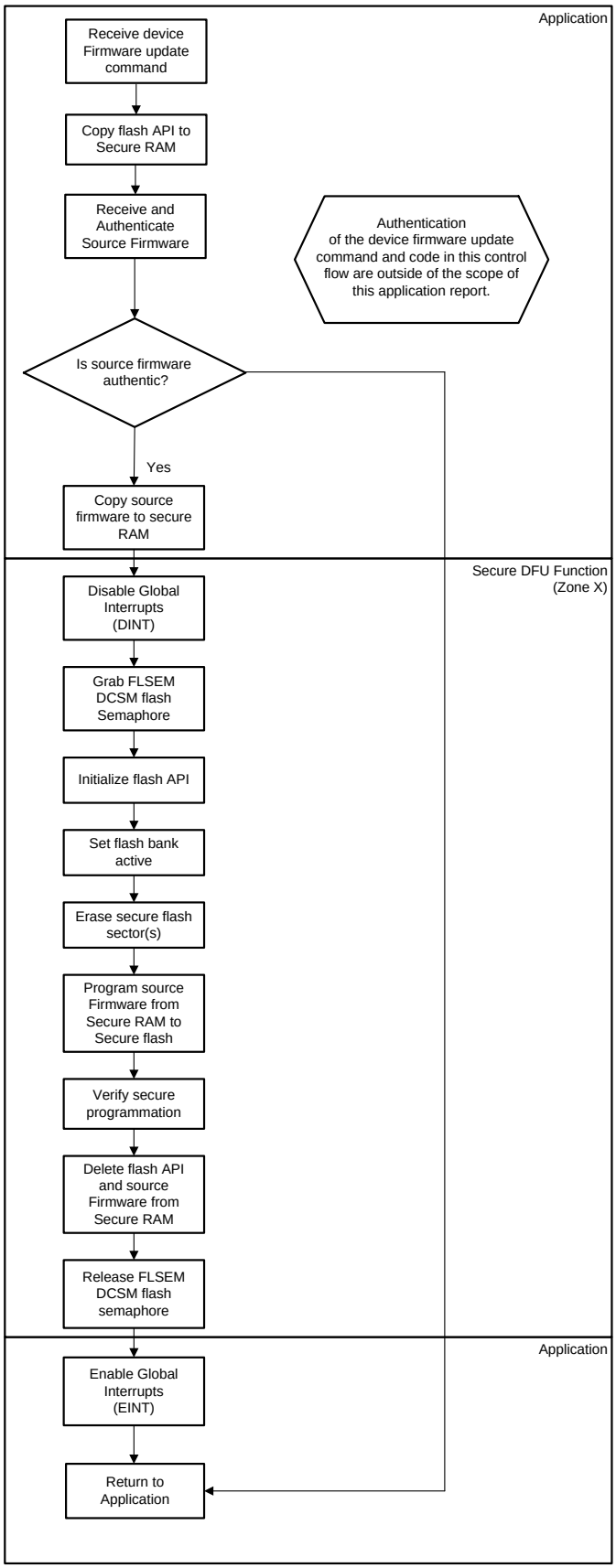

Figure 2. Software Flowchart

By following the procedures mentioned in this application report, secure Flash can be programmed without having to unlock the zone.

4 Summary

This application report discusses the purpose and concepts of the DCSM as well as provides some helpful information to support a more secure firmware update.

5 References

- *Serial Flash Programming of C2000™ Microcontrollers*
- http://www.ti.com/tool/C2000WARE
- http://www.ti.com/product/TMS320F28379S
- *C2000™ Unique Device Number*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265