Application Report
SPRAAQ3–September 2007

Custom Bootloader Options via One-Time Programmable (OTP) Memory

Christine Peng and Hareesh Janakiraman

ABSTRACT

The *Boot to One-Time-Programmable (OTP)* memory mode in the TMS320x280x devices provides the necessary hooks to support custom bootloaders. This is very useful in a situation where a bootloader option, required by a specific customer application, is not already supported as one of the *built-in* options in the Boot read-only memory (ROM). This application report includes information and examples on how to program a custom bootloader and boot via the OTP memory.

Project collateral and source code discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/SPRAAQ3.

Contents

List of Tables

1 Introduction

The current TMS320x280x Boot ROM supports a limited number of Boot mode options. If other Boot modes are desired, it is possible to program the custom Boot code to OTP and then select the *Boot to OTP* option with the appropriate general-purpose input/output (GPIO) pins. Sample Boot code software, which can be programmed into the OTP, is provided with this application report. The provided sample Boot code software allows you to:

- Boot from both 16-bit and 24-bit addressable electrically erasable programmable read-only memory (EEPROM) and/or Serial Flash.
- Boot properly through the eCAN interface as a workaround to the errata mentioned in the *TMS320F280x, TMS320C280x, and TMS320F2801x DSP Silicon Errata* (SPRZ171).

This application report explains the process of programming a custom bootloader into OTP and then booting to OTP. As an example, the sample 16/24-bit addressable Serial bootloader code is used to demonstrate the process. Although the examples and text in this application report refer to the TMS320x280x generation of devices, they apply to any C2000™ DSP device with a *Boot to OTP* mode in the Boot ROM.

2 Programming the OTP With a Custom Bootloader

This section demonstrates the steps required to configure your custom bootloader code for *Boot to OTP* mode and how to program the OTP. For example purposes, the steps are catered to the TI custom serial bootloader provided with this application report. For your own custom bootloader, other modifications can be required.

1. Include the common Boot ROM files in the project.

   Source Files:

   - Init_Boot.asm – This file manages the bootloader stack pointer and calls the custom Boot code. For this particular example, it calls the SPI_Boot() function located in SPI_Boot.c. You can change the following lines in Init_Boot.asm to fit the Boot function in your custom bootloader.

     ```
     ref _SPI_Boot ; refers to SPI_Boot() in SPI_Boot.c
     LCR _SPI_Boot
     ```

   - Shared_Boot.c – This file includes shared functions used by the normal TI Boot ROM code to move data via the peripheral. You can choose to use these functions or create your own.
   - Your custom bootloader code (SPI_Boot.c, for purposes of this example).

   Other:

   - 280x_Boot.h – Included in the project via #include in source files – This file includes specific defines used by Shared_Boot.c and SPI_Boot.c. If you do not use these defines in your custom bootloader, this include is unnecessary.
   - 280x header files, specifically:
     - DSP280x_GlobalVariableDefs.c
     - DSP280x_Headers_nonBIOS.cmd
     - The previous two files, in conjunction with the 280x header files, automatically ensure the appropriate header and source files are included for those peripherals specific to your bootloader.

   For more information on including header files, see the *DSP280x/2801x Header Files Quick Start Readme* packaged with the 280x header files.

2. Configure a linker command file to link the bootloader code to OTP and add it to the project.

   Generate a linker command file, as follows, such that all code (.text) is linked to OTP memory (280x_boot_rom_lnk.cmd in the included examples) . The stack and static variables are located in M1 memory at location 0×400, where the normal Boot ROM stack and static variables are stored.

   Note that the .InitBoot code section is linked to the beginning of OTP memory at 0×3D7800, while the rest of the code is linked to subsequent memory. This ensures that when *jump to OTP* occurs at 0×3D7800, the appropriate code is executed first (in this case, the Init_Boot.asm file). Include this .cmd linker file in your project.

   ```
   MEMORY
   {
   PAGE 0 :

         SPIBOOTINIT : origin = 0x3D7800, length = 0x000020
         SPIBOOT     : origin = 0x3D7820, length = 0x000200
   PAGE 1 :
            EBSS      : origin = 0x400, length = 0x002
            STACK     : origin = 0x402, length = 0x200
   }

   SECTIONS
   {
            .InitBoot    : load = SPIBOOTINIT,  PAGE = 0
            .text        : load = SPIBOOT,      PAGE = 0
            .stack       : load = STACK,        PAGE = 1
            .ebss        : load = EBSS,         PAGE = 1
   }
   ```

3. Set the required jumpers for the *Boot to OTP* mode.
   The required jumper settings for each Boot mode are shown in Table 1.

Table 1. 280x Boot Mode Settings

GPIO18	GPIO29	GPIO34	Mode
1	1	1	Boot to flash 0x3F7FF6
1	1	0	Call SCI-A bootloader
1	0	1	Call SPI-A bootloader
1	0	0	Call I2C bootloader
0	1	1	Call eCAN-A bootloader
0	1	0	Boot to M0 SARAM 0×000000
0	0	1	Boot to OTP 0×3D7800
0	0	0	Call parallel bootloader

   For more detailed information regarding configuring the Boot mode selection pins, see the device-specific documentation for your hardware platform.
   For more information on the 280x Boot modes, see the device-specific Boot ROM Reference Guide.
4. Program the device with the built code.
   This can be done by using the SDFlash tool available from Spectrum Digital's website (www.spectrumdigital.com). In addition, the C2000™ on-chip Flash programmer plug-in for the Code Composer Studio™ can also be used.
   These tools are updated to support new devices as they become available. Please check for updates.
5. Debug the information. Load the project in Code Composer Studio by selecting *File -> Load Symbols -> Load Symbols Only*
   It is useful to load only the symbol information when working in a debugging environment where the debugger cannot or need not load the object code, such as when the code is in ROM or Flash. This operation loads the symbol information from the specified file.

3 TI-Provided Custom Bootloaders

In conjunction with the application report, TI has included two custom bootloaders for the TMS320x280x, which are discussed in this section.

3.1 TMS320x280x Custom Serial Flash Bootloader

The current serial peripheral interface (SPI) bootloader in the Boot ROM boots from 16-bit addressable SPI Serial EEPROM. There is currently no solution available if an application requires the device to Boot from 24-bit addressable memory.

To remedy this, Texas Instruments has provided the bootloader code, which can be programmed into the OTP memory on 280x devices. The */f280x_otp2spi_boot_rom* directory, in the included software package, includes all of the project code and source files required to boot from either 16-bit or 24-bit addressable serial memories.

Note: The code is provided as-is and has been tested on Atmel® SPI Serial EEPROMs (AT25HP256/512 and AT25128/256A) and Atmel 1Mb high-speed SPI Serial Flash memory (AT25F1024A). The functions can be modified to support other memories, if necessary or desired.

For information on how the original SPI-A bootloader functions, see the *TMS320x280x, 2801x, 2804x Boot ROM Reference Guide* (SPRU722). The SPI bootloader included in the software package has been modified slightly from the original SPI-A bootloader to support 24-bit addressable serial memories.

In the modified custom serial bootloader, after the SPI-A outputs a READ command for the serial memory, it no longer sends only 2 bytes of 0×00's (16-bits), indicating a starting address of 0×0000 for a downloadable packet. Instead, it performs the following steps:

1. SPI-A sends three bytes of 0×00's (24-bits) indicating a starting address of 0×000000 for a downloadable packet from the serial memory.
2. The bootloader fetches a byte of data while the SPI-A transmits the third 0×00 (the SPI receives data while it transmits), and checks to see if the received data byte is the least significant bit (LSB) of the key value (0×08AA).
   a. If the data equals 0×AA, the memory is 16-bit addressable (after sending two bytes of 0×00's, the key value is received), and the bootloader fetches one more data byte to check for an 0×08 to ensure the entire key value is correct.
   b. If the data does not equal 0×AA, it is assumed the memory is 24-bit addressable. Three bytes of 0×00's must be sent before the key value is received. The bootloader fetches two more bytes and checks to see that they match the 0×08AA key value for an 8-bit stream.
3. The rest of the SPI bootloader functions exactly as the original SPI Boot ROM code in the 280x devices. For more information, see the *TMS320x280x, 2801x, 2804x Boot ROM Reference Guide* (SPRU722).

3.2 TMS320x280x Custom eCAN Bootloader

As noted in the *TMS320F280x, TMS320C280x, and TMS320F2801x DSP Silicon Errata* (SPRZ171), the eCAN-A boot mode in the current Boot ROM does not work as intended due to a 16-bit R/W employed to check the status of the CCE bit. The boot code has been fixed, and can be programmed into the OTP memory for *Boot to OTP mode*.

The fixed eCAN bootloader project code can be found in the */f280x_otp2can_boot_rom* directory of the included software package. After the bootloader is programmed into the OTP and the GPIO pins are configured for *Boot to OTP* mode, the device can boot from the eCAN-A peripheral via the OTP bootloader the same way it would if the eCAN-A bootloader were in Boot ROM.

For more information on how the eCAN-A bootloader functions, see the *TMS320x280x, 2801x, 2804x Boot ROM Reference Guide* (SPRU722).

4 Conclusion

Booting from a custom bootloader via OTP is a solution for those bootloader options not provided by TI in the Boot ROM code. Two examples have been explained in this application report and included in the attached software package as building blocks for you to begin generating your own custom bootloader. Although this application report specifically targets the TMS320x280x generation of devices, the same general tips can be used for other devices with on-chip OTP memory and a *Boot to OTP* option with minor modifications.

5 References

- *TMS320x280x, 2801x, 2804x Boot ROM Reference Guide* (SPRU722)
- TMS320F280x, TMS320C280x, and TMS320F2801x DSP Silicon Errata (SPRZ171)
- Spectrum Digital's website (www.spectrumdigital.com)

Appendix A Code Listing

A.1 Common Files

The application code's projects consists of the files shown in Table A-1 and Table A-2. This project collateral and source code can be downloaded from the following URL: http://www-s.ti.com/sc/techlit/spraaq3.zip

Table A-1. SPI Serial Memory Bootloader Files Used in This Application Report

Filename	Contents
DSP280x_headers	The DSP280x header files which are utilized by the bootloader code
Init_Boot.asm	280x Boot ROM initialization and exit routines
Shared_Boot.c	280x bootloader shared functions
SPI_Boot.c	280x 16/24-bit addressable SPI boot mode routines
F280x_Boot.h	F280x Boot ROM definitions
f280x_otp2spi_boot_rom.pjt	16/24-bit addressable SPI bootloader project file
F280x_boot_rom_lnk.cmd	F280x Boot ROM linker command file

Table A-2. eCAN-A Bootloader Files Used in This Application Report

Filename	Contents
DSP280x_headers	The DSP280x header files which are utilized by the bootloader code
Init_Boot.asm	280x Boot ROM initialization and exit routines
CAN_Boot.c	280x CAN boot mode routines
Shared_Boot.c	280x bootloader shared functions
F280x_Boot.h	F280x Boot ROM definitions
f280x_otp2can_boot_rom.pjt	280x CAN bootloader project file
F280x_boot_rom_lnk.cmd	F280x Boot ROM linker command file

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