Application Report
SPRABL7D–September 2011–Revised July 2018

Using the TMS320C5545/35/34/33/32 Bootloader

ABSTRACT

This application report describes the features of the on-chip ROM for the TMS320C5545/35/34/33/32 device. Included is a description of the bootloader and how to interface with it for each of the possible boot devices, as well as instructions for generating a boot image to store on an external device. Instructions are provided for burning the boot image onto the C5535 eZdsp (TMDX5535EZDSP) and C5545 BoosterPack (BOOST5545ULP) boards.

Collateral and source code discussed in this document can be downloaded from:

http://www.ti.com/lit/zip/sprabl7.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

1.1 On-Chip ROM

The on-chip ROM contains several factory-programmed sections:

- Algorithm tables to be referenced by drivers to reduce application data size
- Application programming interface (API) tables for referencing ROM API functions in applications
- Bootloader program

Table 1. ROM Memory Map

Starting Byte Address	Contents
FE_0000h	LCD Table
FE_0860h	WMA Encode Table
FE_9FA0h	WMA Decode Table
FE_D4C4h	MP3 Table
FE_F978h	Equalization Table
FE_FB14h	API Table
FE_FE9Ch	Bootloader Code (and other built-in API functions)
FF_FEFCh	Bootloader ID
FF_FF00h	Bootloader Interrupt Vector Table

1.2 Bootloader Features

The major features of the bootloader are:

- Support for unencrypted boot images from external devices (see Section 1.3.1).

The bootloader also has the following features:

- Port-addressed register configuration during boot
- Programmable delay during register configuration

The bootloader is always invoked after reset. The function of the bootloader is to transfer user code from an external source to RAM. Once the transfer is completed, the bootloader transfers control to this user code.

1.3 Supported External Devices

The bootloader is responsible for bootloading the code from an external device.

NOTE: SARAM31 (byte address 0x4E000 – 0x4FFFF) is reserved for the bootloader.

1.3.1 Unencrypted External Devices

The following external devices support unencrypted booting:

- 16-bit SPI EEPROM
- 24-bit SPI Serial Flash
- I2C EEPROM
- SD/SDHC/eMMC/moviNAND
- UART Host
- USB Host

2 Bootloader Operation

2.1 Bootloader Initialization

When the bootloader begins execution, it performs some initialization before attempting to load code:

- All peripherals are idled (the bootloader un-idles peripherals as it uses them).
- CPU Clock setup
  - If CLK_SEL = 0, the bootloader powers up the PLL and sets its output frequency to 12.288 MHz (multiply 32 768 Hz RTC Clock by 375).
  - If CLK_SEL = 1, the bootloader bypasses the PLL and uses CLKIN. Note that CLKIN is expected to be 11.2896 MHz, 12.0 MHz, or 12.288 MHz.
- The low-voltage detection circuit is disabled to prevent trim setup (next step) from causing an unnecessary reset.
- The bootloader reads the trim values from the e-fuse farm and writes them into the analog trim registers.

2.2 Boot Devices

Each device has a fixed order in which it checks for a valid boot image on each supported boot device. The device order is 16-bit SPI EEPROM, 24-bit SPI serial flash, I2C EEPROM, and SD/SDHC/eMMC/moviNAND. The first device with a valid boot image will be used to load and execute user code.

If none of these devices has a valid boot image, the bootloader will modify the CPU Clock setup as follows:

- If CLK_SEL = 0, the bootloader powers up the PLL and sets its output frequency to 36.864 MHz (multiply 32 768 Hz RTC Clock by 1125).
- If CLK_SEL = 1, the bootloader powers up the PLL and sets it to multiply CLKIN by 3.

This CPU clock setup change is required to meet the minimum frequency needed by the USB module.

Next the bootloader goes into an endless loop checking for data received on either the UART or USB. If a valid boot image is received from either device, it will be used to load and execute user code. If no valid boot image is received, the bootloader will simply continue to monitor these two devices. During this endless loop, if the time since the trim setup exceeds 200 ms, then the bootloader will re-enable the low-voltage detection circuit. This is done to prevent leaving the low-voltage detection disabled for an extended period of time.

See Section 3 for a description of the valid boot image formats.

The following subsections describe details for each supported boot device.

2.2.1 16-Bit SPI EEPROM

The bootloader supports booting from an SPI EEPROM with the following requirements for the external device:

- The device must support at least a 500-kHz SPI clock.
- The device must be connected to SPI CS0 and act as an SPI slave.
- The device uses two bytes (16 bits) for internal addressing (up to 64KB).
- The device must have the capability to auto-increment its internal address counter to allow sequential reads from the device.
- The device may be connected to either of the two available pin-mappings for SPI. The bootloader attempts to communicate on each SPI pin-mapping, one at a time.

2.2.2 24-Bit SPI Serial Flash

The bootloader supports booting from an SPI Flash with the following requirements for the external device:

- The device must support at least a 500-kHz SPI clock.
- The device must be connected to SPI CS0 and act as an SPI slave.
- The device uses three bytes (24 bits) for internal addressing (up to 16MB).
- The device must have the capability to auto-increment its internal address counter to allow sequential reads from the device.
- The device may be connected to either valid pin-mapping for SPI (there are two distinct pin-mappings available). The bootloader attempts to communicate on each SPI pin-mapping, one at a time.
- Any and all write-protect features must be disabled if re-authoring is needed. The Bootloader will not attempt to disable these features.
- Figure 1 must match this waveform: read command (03h opcode) followed immediately by 3 byte address, followed immediately by data output by SPI Flash (no additional clock cycles between any of the three stages).

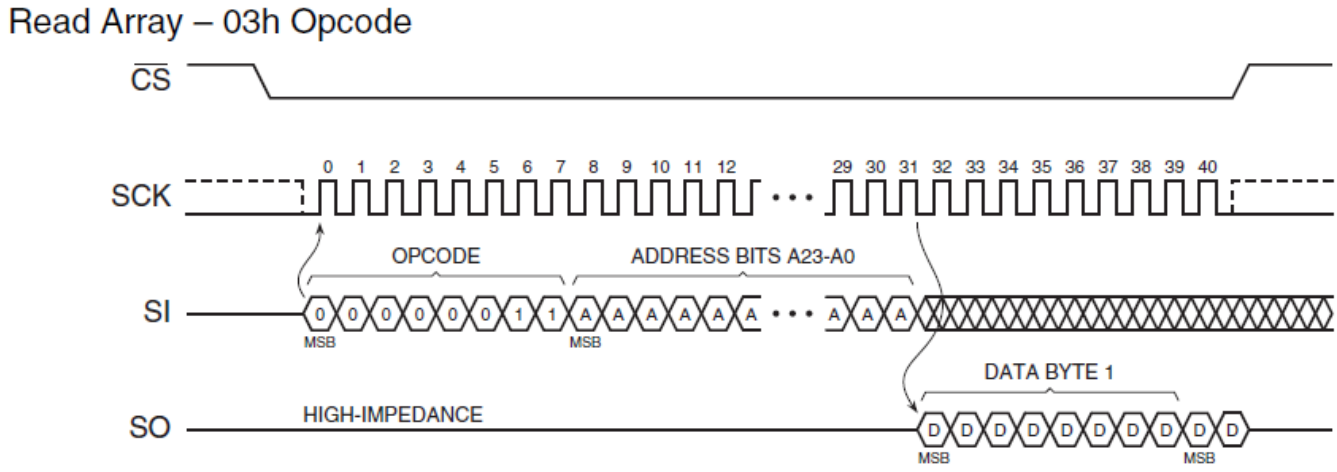

Figure 1. C5535 SPI Flash Boot Waveform

2.2.3 I2C EEPROM

The bootloader supports booting from an I2C EEPROM with the following requirements for the external device:

- The device must support the *fast* I2C specification (400 kHz).
- The device must respond to slave address 0x50 (7-bit address).
- The device uses two bytes for internal addressing (up to 64KB).
- The device must have the capability to auto-increment its internal address counter to allow sequential reads from the device.

2.2.4 SD

The bootloader supports booting from an SD device with the following requirements for the external device:

- The device must be connected to the eMMC/SD0 interface.
- The SD device must comply with *SD Specifications Part 1 Physical Layer Simplified Specification v1.1* or *v2.0* and formatted in FAT16/32.
- The SD device must use the SD insecure mode (see SD specification). Note that this does not refer to boot image security.

- The boot image must be in the first partition with a filename of “bootimg.bin”.

2.2.5 eMMC/moviNAND

The bootloader supports booting from an eMMC/moviNAND device with the following requirements for the external device:

- The device must be connected to the eMMC/SD0 interface.
- The eMMC/moviNAND device must comply with eMMC Specification v4.3, or later, and formatted in FAT16/32.
- The boot images can be in either the boot partition (with boot-from-partition enabled) or in the user data area formatted in FAT16/32. The boot image must be in the first data partition with filename “bootimg.bin”.
- The bootloader will check for user data partition first. If there is no valid boot image, then it will check for the boot from partition option.

2.2.6 UART

The bootloader supports booting from the UART. The bootloader sets up to receive data using the following UART parameters: 8-bit data, odd-parity, 1 stop-bit, and 57 600 baud rate.

To reduce the probability of a receive error, the external transmitter should set up to use two stop-bits.

Note that this setup may result in data receive errors if CLK_SEL is set to use CLKIN when CLKIN is 11.2896 MHz. The error rate may be reduced by the external device by adding additional time between frames (bytes).

2.2.7 USB

The bootloader supports booting from the USB. The bootloader uses bulk-endpoint 1 (OUT), vendor-id 0x0451, and product-id 0x9010.

2.3 Register Configuration

Once the bootloader detects a valid boot image signature (see Section 3), the first data that is used from the boot image is the optional register configuration data. This data allows the user to set up peripheral port-addressed registers during the boot process and before the code sections are copied. This feature provides the capability to change peripheral registers for specific purposes.

NOTE: Using the register configuration feature to reprogram register settings may cause the bootloader to fail.

Since some register configurations may have an associated latency that must be observed before continuing, a delay feature is also available as part of the register configuration data.

For a description of how to insert register configuration data, including delays, into a boot image, see:

- Section 3.1.1, *Creating a Boot Image*

2.4 Code Sections

After the optional register configuration is complete, the bootloader will copy all of the code sections from the boot image to RAM. Each of these code sections may be actual code or just data. These sections are typically defined by the link-control file.

2.5 Bootloader Completion

After all code sections have been copied, the bootloader will wait to ensure that at least 200 ms have elapsed since the trim setup. Re-enable the low-voltage detection circuit, and then branch to the entry-point address specified in the boot image.

At this point, the bootloader’s task is complete and the user application is executing.

3 Boot Images

The bootloader's primary function is to transfer user code into RAM and then transfer control to this user code. The user code must be formatted into a boot image format supported by the bootloader.

The following sections describe these two boot image formats and how to create boot images.

3.1 Boot Image Format

The boot image format contains the following information:

- All user code/data sections to be loaded to RAM
- Register configuration data for setting up peripheral registers prior to loading code
- The entry-point of the user's application
- A boot signature to the boot image boot signature is 0x09AA..

Table 2 shows the boot image format.

Table 2. Boot Image Format

Word	Content	Valid Data Entries
1	Boot Signature (16-bits)	0x09AA
2	Entry Point (32 bits)	Byte address to begin execution (MSW)
3		Byte address to begin execution (LSW)
4	Register Configuration Count (16 bits, N = count)	1 to 2^{16} - 1
5	Register Config #1 Address in I/O space	Repeated according to register configuration count. Register configuration address is any valid register in I/O space. Address 0xFFFF is reserved as a delay indicator to create delay in between register writes or at end of register writes.
6	Register Config #1 Value or delay count	
7	Register Config #2 Address in I/O space	
8	Register Config #2 Value or delay count	
	. . .	
	Register Config #N Address in I/O space	
4+2N	Register Config #N Value or delay count	0 to 2^{16} - 1
5+2N	Section 1 word count (16 bits) Size is the number of valid (non-pad) data words in block M = (size + 2) rounded up to nearest multiple of 64-bit boundary	1 to 2^{16}-1
6+2N	Destination MSW address to load Section 1 (32 bits)	16-bit word address MSW
7+2N	Destination LSW address to load Section 1 (32 bits)	16-bit word address LSW
8+2N	First word of Section 1 (16 bits)	
	. . .	
5+2N+M	Last word of Section 1, often pad data (padded to 64-bit boundary)	
	. . .	
X	Section X word count (16 bits) Size is the number of valid (non-pad) data words in block N' = (size + 2) rounded up to nearest multiple of 64-bit boundary	1 to 2^{16}-1
X+1	Destination MSW address to load Section X (32 bits)	16-bit word address MSW
X+2	Destination LSW address to load Section X (32 bits)	16-bit word address LSW
X+3	First word of Section X (16 bits)	
	. . .	
X+N'	Last word of Section X, often pad data (padded to 64-bit boundary)	
X+N'+1	Zero word. Note that if more than one source block was read, word X+*N'* shown above would be the last word of the last source block. Each block would have the format shown in the shaded entries.	0x0000

3.1.1 Creating a Boot Image

A boot image can be created using the hex conversion utility (hex55) utility.

For detailed information on the available hex conversion utility output formats, see the *TMS320C55x DSP Assembly Language Tools User's Guide* (SPRU280).

Use the hex conversion utility (hex55.exe) revision 4.3.5 or later. Earlier versions may not support the boot table features correctly.

3.1.1.1 Creating a Boot Image Using hex55

To create a boot table for the application (*my_app.out*) with the following conditions:

- Desired boot mode is 8-bit standard serial boot
- No registers are configured during the boot
- No programmed delays will occur during the boot
- Desired output is binary format in a file called *my_app.bin*

Use the following options on the hex conversion utility command line or command file:

```
hex55 -boot -v5505 -serial8 -b -o my_app.bin my_app.out
```

```
-boot                       ;option to create a boot table
-v5505                      ;use C55x boot table format
-serial8                    ;boot mode is 8-bit standard serial boot
-b                          ;desired output format is binary format
-o my_app.bin               ;specify the output filename
my_app.out                  ;specify the input file
```

3.1.1.2 Creating a Boot Image With I/O Register Configuration

To create a boot table for the application *my_app.out* with the following conditions:

- Desired boot mode is from 8-bit standard serial boot
- Configure the register address 0x1C8C with the value 0x0001
- After the register is configured, wait 256 cycles before continuing the boot
- Desired output is binary format in file a called *my_app.bin*

Use the following options on the hex conversion utility command line or command file.

Note: If using the reg_config option, ensure there is no space between the address and value.

```
hex55 -boot -v5505 -serial8 -reg_config 0x1c8c,0x0001 -delay 0x100 -b -o my_app.bin my_app.out
```

```
-boot                       ;option to create a boot table
-v5505                      ;use C55x boot table format
-serial8                    ;boot mode is 8-bit standard serial boot
-reg_config 0x1c8c,0x0001   ;write 0x0001 to peripheral register at address 0x1C8C.
                             This can be repeated to program multiple registers.
-delay 0x100                ;delay for 256 CPU clock cycles
-b                          ;desired output format is binary format
-o my_app.bin               ;specify the output filename
my_app.out                  ;specify the input file
```

NOTE: Using the register configuration feature to reprogram register settings may cause the bootloader to fail.

3.1.1.3 Section Alignment Restrictions When Using Hex55 Utility

All code sections must be aligned on a word boundary. Sections that are not properly aligned will be flagged by the hex55 utility.

To align a code section, use the *align* command in the linker command file as shown below. Note that if any function included in a code output section has an alignment associated with it (in C via CODE_ALIGN pragma) the whole section will inherit that alignment.

```
.text > ROM PAGE 0 align 2
```

3.1.1.4 DOS Command Line for Generating Boot Image Using Hex55

```
hex55 -boot -v5505 -b -serial8 -o USBKey_LED.bin USBKey_LED.out
```

3.2 Burn a Boot Image

The following instructions detail how to burn a boot image onto the C5535 eZdsp (TMDX5535EZDSP) and the C5545 BoosterPack (BOOST5545ULP). Once a boot image (*.bin) is generated, customers can burn the boot image into the SPI EEPROM (16-bit or 24-bit), I2C EEPROM, SPI serial flash or SD/SDHC/eMMC/moviNAND. It is done by a utility called programmer which runs on each device using an emulator with CCS. First you will need to load the program. This programmer programs the SPI serial flash on both the C5535 eZdsp and the C5545 BoosterPack. It can be downloaded from the Spectrum Digital board support page: http://support.spectrumdigital.com/boards/ezdsp5535/. A more generic programmer project is also available in the TMS320C55x Chip Support Libraries (CSL) – Standard and Low-Power.

For the C5535 eZdsp, ensure SW3 pin 4 in in the "ON" position.

For writing to SPI serial flash, at the “C5535 eZdsp … input <file_path>” prompt, input the path name for the boot image<enter>. The following display will be shown:

SPI Flash…
Erase chip (SPI Flash)…
Open <file-path>
Input file opened
Programming Complete

3.3 Boot From Micro SD/SDHC Card

To boot from micro SD/SDHC card:

1. Ensure there is no valid boot image in any device ahead of SD/SDHC in the order that the bootloader checks for a valid boot image. For more information, see Section 2.2.
2. Format the micro SD/SDHC to FAT16/32.
3. Copy the boot image into the root directory and rename it to “bootimg.bin”.
4. Insert the micro SD/SDHC in the micro SD slot (J6 on C5535 eZdsp, J3 on C5545 BoosterPack).
5. Power cycle the board.

The bootloader will boot from the micro SD/SDHC card.

3.4 Boot From UART

To boot from UART:

1. Ensure there is no valid boot image in any device ahead of UART in the order that the bootloader checks for a valid boot image. For more information, see Section 2.2.

   The Debug USB port (J2 on C5535 eZdsp, J9 on C5545 BoosterPack) should be connected to a PC USB port via USB cable. It will use the virtual COM port (the COM port number varies by system) through TI XDS100 Channel B.

2. If using the C5535 eZdsp, ensure pin 2 of SW3 is in the ON position. If using the C5545 BoosterPack, ensure jumpers on JP2 and JP3 are each between pins 1 and 3, see Figure 2).

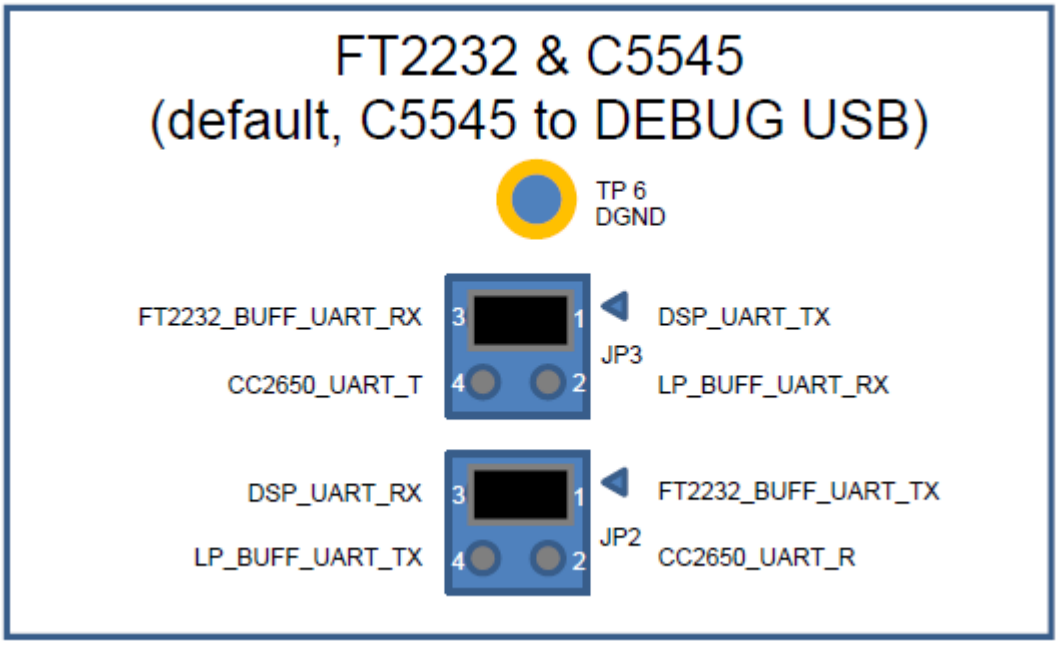

Figure 2. C5545 JP2 JP3 for UART Boot

To see the properties for TI XDS100 Channel B, open Control Panel→System→Hardware→Device Manager→USB Controller→TI XDS100 Channel B→Properties→Advanced.

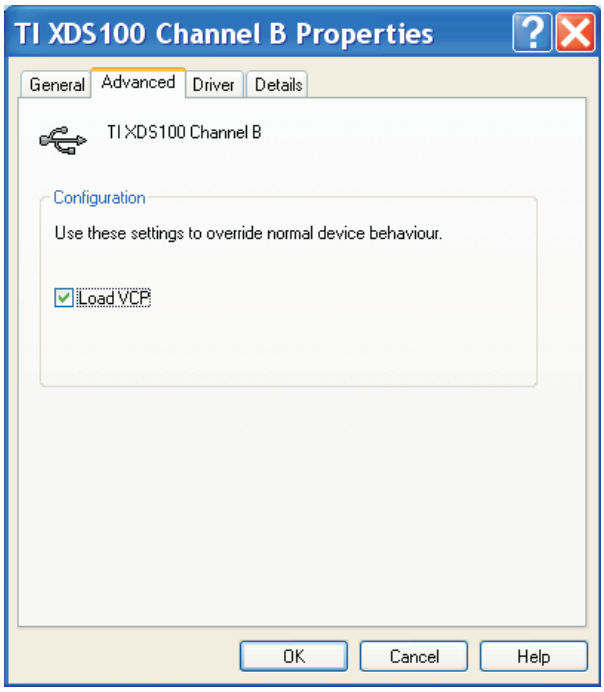

Figure 3. Enable the Virtual COM Port

To see the properties of a particular COM port, open Control Panel→System→Hardware→Device Manager→Ports (COM & LPT)→USB Serial Port (COMxx).

The properties dialog box is shown in Figure 4.

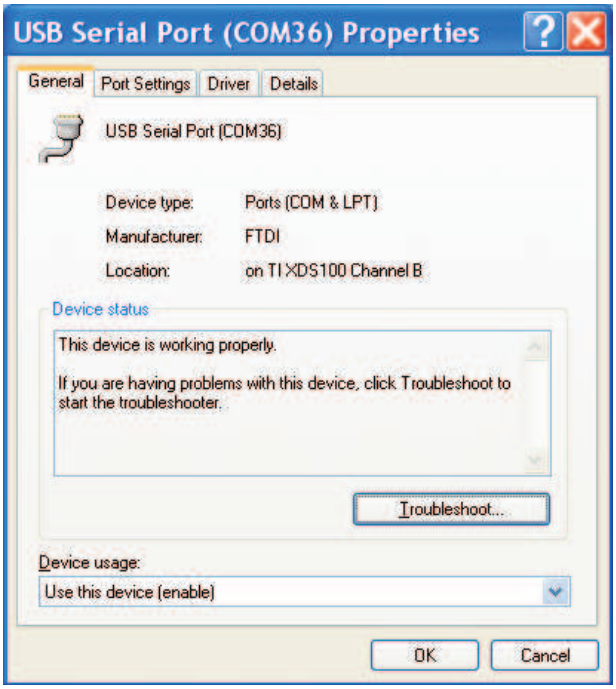

Figure 4. Get the Virtual COM Port Number

3. Run the UartBoot.exe (Figure 5) and select the 57600 in “Baud Rate”.
4. Enter the correct COM port number in “PC COM Port” and check the “Serial Port”.
5. Click “Send File >>”.

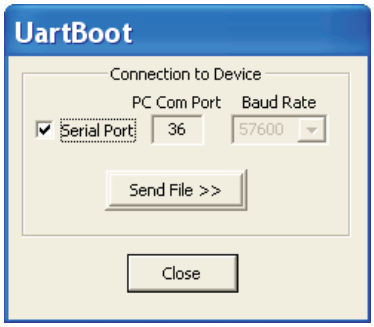

Figure 5. Running UartBoot.exe

In the file selection dialog box (Figure 6):

6. Select the file to upload: demo.bin, in this case.

7. Click "Open" to start uploading.

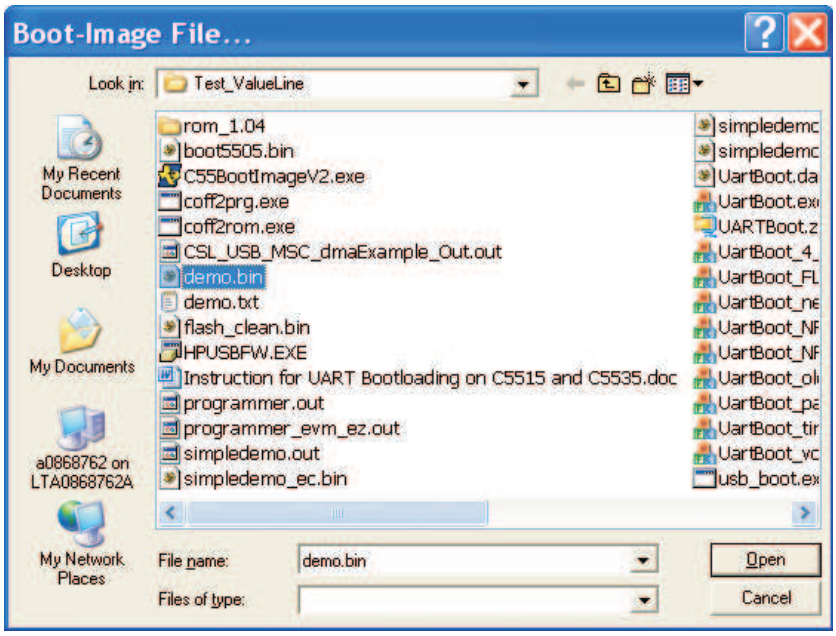

Figure 6. Select the Boot Image File

After the uploading is complete, a message box appears, as in Figure 7.

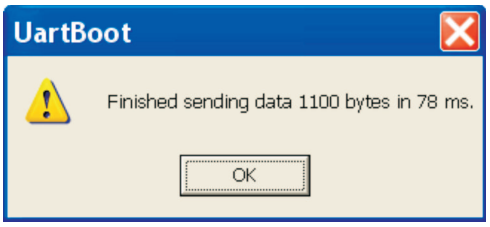

Figure 7. UART Boot is Completed

The demo.bin now runs on the device.

3.5 Boot From USB

To boot from USB:

1. Ensure there is no valid boot image in any device ahead of USB in the order that the bootloader checks for a valid boot image. For more information, see Section 2.2.

   The Client USB port (J1 on C5535 eZdsp, J1 on C5545 BoosterPack) should be connected to a PC USB port via a USB cable.
2. Execute usb_boot.exe and enter option 3 for BootImage Test.
3. Input the boot image path name: demo.bin, in this case.

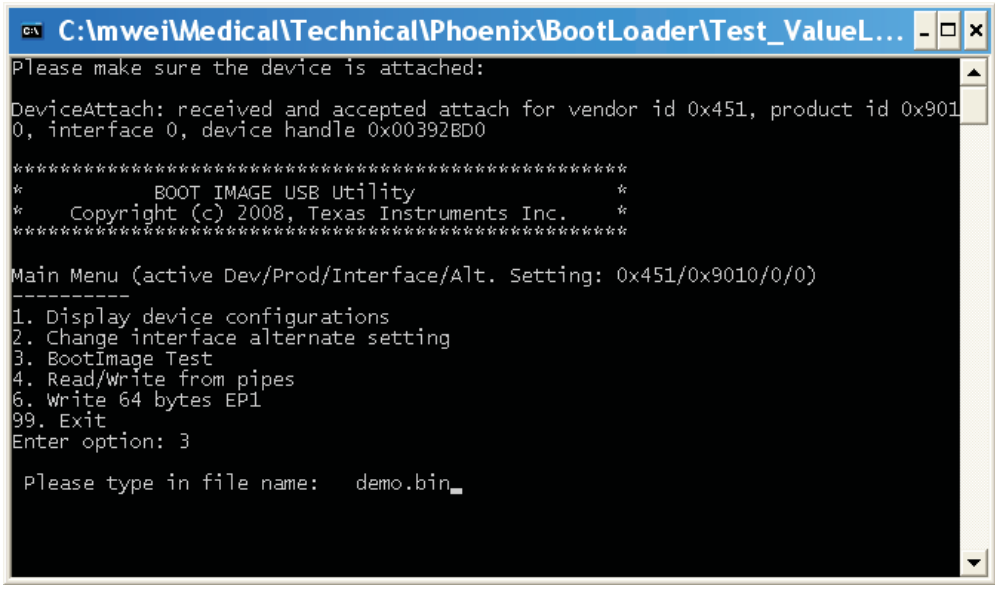

Figure 8. Running usb_boot.exe

After <enter>, the following dialog box appears.

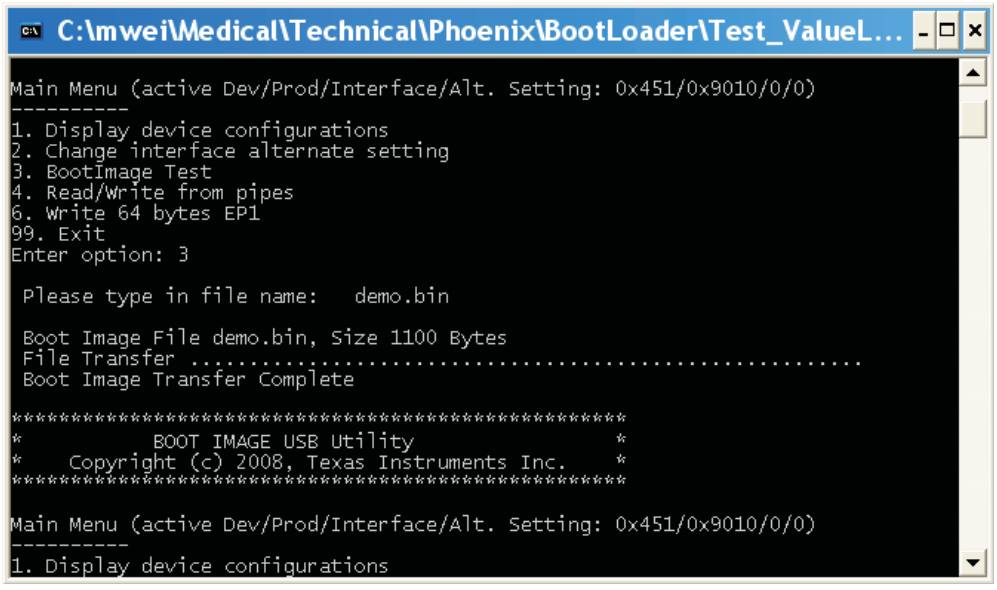

Figure 9. USB Boot is Completed

The demo.bin now runs on the device.

4 References

- *TMS320C55x DSP Assembly Language Tools User's Guide* (SPRU280)

Revision History

Changes from C Revision (April 2016) to D Revision	Page
• Update made to Abstract.	1
• Removed all references of encrypted boot as TI no longer supports it for this family of devices on any new designs.	1
• Removed encrypted boot options. TI no longer supports encrypted boot on these devices.	2
• Added support for the C5545 BoosterPack (BOOST5545ULP) in addition to the C5535 eZdsp (TMDX5535EZDSP).	2
• Updates were made in Section 1.3.1.	2
• Update was made in Section 2.2.2.	4
• Update was made in Section 2.2.4.	4
• Update was made in Section 2.2.5.	5
• Update was made in Section 3.	6
• Updates were made in Section 3.1.	6
• Updates were made in Section 3.1.1.	7
• Updates were made in Section 3.2.	8
• Updates were made in Section 3.3.	8

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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