

# **Using the TMS320VC5506/C5507/C5509/C5509A USB Bootloader**

*Mathew George, Jr. (Joe) and Clay Turner*

## **ABSTRACT**

Bootloading the TMS320VC5506/C5507/C5509/C5509A digital signal processor (DSP) through the on-chip universal serial bus (USB) peripheral is part of the standard bootloader provided on the device. This document describes the procedures for physically connecting the DSP to a USB host, invoking the USB bootloader on the DSP, generating the correct boot table file, and downloading the boot table from the host to the DSP via USB.

You can perform the host-side functions using an existing driver provided by TI third-party provider, Thesycon, or write a custom driver. Information on both of these options is provided.

For general information about the TMS320VC5506/C5507/C5509/C5509A bootloader and other available boot modes, see *Using the TMS320VC5503/C5506/C5507/C5509/C5509A Bootloader* ([SPRA375](#)).

Project collateral and source code discussed in this application report can be downloaded from the following URL: <http://www.ti.com/lit/zip/SPRA840>.

## **Contents**

1	Introduction.....	2
2	DSP USB Bootloader Resources.....	2
3	Connecting the C5506/C5507/C5509/C5509A USB Port to the Host .....	3
4	Selecting the USB-Boot Mode .....	3
5	Creating the Binary Boot Table.....	4
6	USB Boot Using the Thesycon (USBIO) Driver and Demo Application.....	5
7	Example Boot Table: boot_img.bin .....	12
8	Summary of USB Boot-Load Steps.....	12
9	Customizing the USB Bootload.....	12
10	References .....	12

## **List of Figures**

1	DSP USB Peripheral Connection to the Host.....	3
2	Boot Table Structure .....	4
3	USB Demo Application Initial Dialog Box.....	6
4	Output Window in Step 6 .....	7
5	USB Demo Application in Step 8 .....	8
6	USB Demo Application in Step 9 .....	9
7	Output Window Following Step 9 .....	10
8	USB Demo Application in Step 11 .....	11
9	Output Window for Step 11 .....	11

## **List of Tables**

1	USB Bootloader Resource Summary.....	2
---	--------------------------------------	---

## 1 Introduction

The DSP supports a standard bootloader that provides the capability to load application code to run, following a reset of the DSP. The bootloader is implemented in on-chip read-only memory (ROM) and supports several boot modes, allowing the application to be loaded from external memory, from the host port interface, or from any of several peripherals. Included among these is the option to load the desired application via the USB peripheral. This document covers the use of the USB bootload option only. For information about use of the other boot modes and general information on the bootloader, see *Using the TMS320VC5503/C5506/C5507/C5509/C5509A Bootloader* ([SPRA375](#)).

This application report provides information specific to the USB-boot mode and covers procedures for physically connecting the DSP to the USB host, invoking the bootloader on the DSP, generating the correct boot table file, and downloading the boot table from the host to the DSP.

Detailed knowledge of USB protocol and the USB peripheral on the C5506/C5507/C5509/C5509A is not necessary to use the USB-boot mode and, consequently, is not covered here. For those interested in more detail on the operation of USB, several resources are available. The complete USB Specification is available at <http://www.usb.org>. Information on the operation and programming of the USB peripheral is available in the *TMS320C55x DSP Peripherals Reference Guide* ([SPRU317](#)) and the *TMS320C55x CSL USB Programmer's Reference Guide* ([SPRU511](#)). Other related references are listed in [Section 10](#).

In this document, all references to TMXVC5506/C5507/C5509/C5509A are equivalent to TMS320VC5506/C5507/C5509/C5509A, unless otherwise specified.

## 2 DSP USB Bootloader Resources

The bootloader code is stored in on-chip ROM on the DSP and is available for use following reset. The USB-boot mode requires some hardware and software resources, as specified below in [Table 1](#).

**Table 1. USB Bootloader Resource Summary**

Bootloader USB Specifics	Start (byte address)	End (byte address)	Frequency	Value
USB VID (vendor identification)				0x0451
USB PID (product identification)				0x9001
Endpoint address				0x06
Endpoint type				Bulk Out
<b>Boot Mode Pin Settings</b>				
BOOTM[3:0] <sup>(1)</sup>				0010b
<b>Input Clock</b>				
CLKIN/X2			12.0 MHz	
<b>Memory Usage</b>				
Bootloader reserved ROM	0xFF0000	0xFFFFF		
USB bootloader reserved RAM				
• C5506/C5507	0x01F800	0x01FFFF		
• C5509/C5509A	0x03F800	0x03FFFF		

<sup>(1)</sup> BootM[3:0] and the GPIO 0, GPIO[3:1] are the same pins.

The bootloader code is stored in ROM at byte address 0xFF0000. This code runs following reset, if the USB-boot mode is selected (by the configuration of the BOOTM[3:0] pins). The USB boot code reserves byte addresses (see [Table 1](#)) of on-chip RAM for its own use during the bootload. You should not map any part of the application to be loaded to this reserved space. All other internal and external memory space is available to the application. Note that the application may use the reserved space after the bootload has completed, but the image to be loaded cannot use this space.

Code Composer Studio is a trademark of Texas Instruments.  
All other trademarks are the property of their respective owners.

When the USB boot code runs, the USB peripheral is configured, and the DSP is ready to receive the boot table (the boot table is covered in [Section 5](#)) from the host. The USB bootloader configures Endpoint 6 Out as a bulk endpoint. The host should send the boot table to this endpoint. The bootloader uses vendor identification (VID) 0x0451 and product identification (PID) 0x9001 during enumeration with the host operating system. The host operating system needs the VID and PID to load the appropriate driver, which is essential to establish communication between the host application and the bootloader.

### 3 Connecting the C5506/C5507/C5509/C5509A USB Port to the Host

Connecting the USB peripheral to the host only requires connection of the USB differential data driver pins. The recommended connection is shown below in [Figure 1](#). The positive differential signal on the USB peripheral is called DP. The negative differential signal is called DN.

The USB specification requires that the equivalent output impedance of these pins be between 29 and 44  $\Omega$ . The series resistors  $R_s(DP)$  and  $R_s(DN)$  are needed to ensure that this impedance requirement is met. The board designer should consult the device data sheet for the appropriate series resistor values.

The 1.5k  $\Omega$  resistor on the PU pin tells the USB host (or hub) that a full-speed device is present on the bus. The PU pin is internally connected to the  $USBV_{DD}$  supply on the DSP by a software-controllable switch (CONN bit of USBCTL register). Setting this bit causes the USB host to detect the presence of the DSP on the bus and begin the enumeration process. The USB bootloader sets this bit when it is ready for the enumeration.

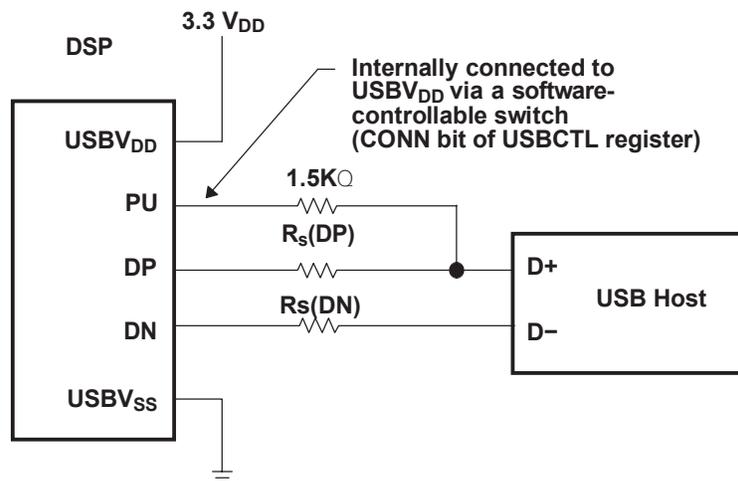


Figure 1. DSP USB Peripheral Connection to the Host

### 4 Selecting the USB-Boot Mode

On the standard DSP bootloader, the application can be loaded from a variety of sources. The bootloader determines which port or peripheral will be the source of the data by polling the BOOTM[3:0] pins at reset. The configuration of these pins determines which boot mode will be executed. To select the USB-boot mode, set BOOTM[3:0] = 0010b at reset. The bootloader will then configure the USB peripheral appropriately and set the CONN bit in the USBCTL register to 1, to indicate to the host (or hub) that the DSP is now connected to the USB bus. The host will then recognize the DSP as a USB boot device, complete the enumeration process, and begin the download of the application code.

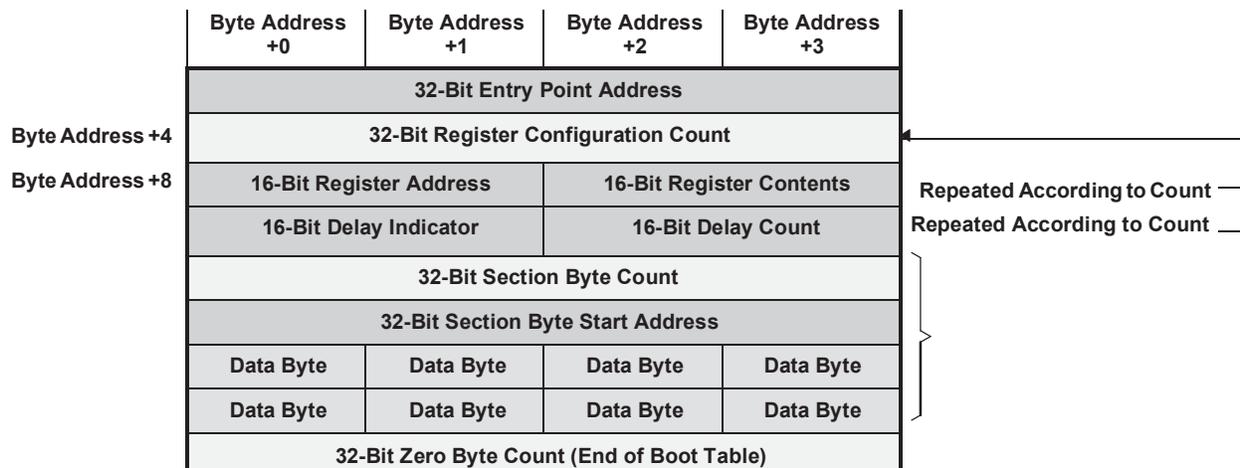
For complete information on the boot modes available on the C5506/C5507/C5509/C5509A and how they are selected, see *Using the TMS320VC5503/C5506/C5507/C5509/C5509A Bootloader* ([SPRA375](#)).

## 5 Creating the Binary Boot Table

The standard DSP bootloader uses a data format called a boot table while loading an application. The boot table includes information about the entry-point address for the application, start addresses and lengths for each of the code sections, and the code content itself. In addition, the bootloader can preset peripheral register contents during the bootload, if desired. The commands to perform this function are also part of the boot table.

The general structure of the boot table is shown in [Figure 2](#). The first 4 bytes contain the entry-point address. This is the program memory address where execution of the application will begin, once the bootload is complete. The next part of the table supports the peripheral register configuration option. If some peripheral registers need to be preset during the boot-load process, the register addresses and contents will be included in the boot table. Delays can also be inserted to allow time for the changes to become effective. The register configuration count, shown in [Figure 2](#), is the number of registers that will be modified.

The next entries in the table are the application code sections. Each section has a start address and a length (in bytes) followed by the actual contents of the section. The end of the boot table is indicated by a section of zero length. For detailed information on the boot table format, see *Using the TMS320VC5503/C5506/C5507/C5509/C5509A Bootloader* ([SPRA375](#)).



The boot table is generated automatically by the hex conversion utility (Hex55.exe). This utility is part of the standard code generation tools. The hex conversion utility converts the common object file format (COFF) output file produced by the linker and creates the desired converted file based on the options specified. A complete description of the functions and operation of the hex conversion utility is covered in the *TMS320C55x Assembly Language Tools User's Guide* ([SPRU280](#)).

## 5.1 Using the Hex Conversion Utility (Hex55.exe) to Generate the Boot Table

**Note:** Hex Conversion Utility version 2.53 or later is required for this method. If the version of Code Composer Studio™ software you are currently using does not yet support this version of the hex conversion utility, it is available in the associated files available for download with this document.

1. Create a command file for the hex conversion utility similar to Example 1. In this example, the hex conversion utility command file is called *serial8bin.cmd*, and the COFF output file for the application is called *yourfile.out*.

### Example 1. serial8bin.cmd

```
yourfile.out
-boot                ; causes a boot table to be produced
-v5510:2            ; selects the appropriate boot table format
-serial8            ; selects an 8-bit serial boot table format
-b                  ; generates an binary output file
-o yourfile.bin     ; sets the output file name
-map yourfile.bxp   ; generates an output map file with the name shown (optional)
```

The *-boot* option causes the output of the hex conversion utility to be a boot table (instead of a simple data file representing the COFF file contents). The *-v5510:2* option selects the appropriate boot table format. Even though this option refers to the C5510, it applies to the C5506/C5507/C5509/C5509A also. The *-serial8* option tells the hex conversion utility which boot type will be used. The *-b* option causes a binary output file to be produced. The *-o* option sets the desired output file name. You can also optionally create a map file (with the *-m* option) that shows the memory assignments of each code section. This can be useful for debugging purposes, but is not required.

2. Run the hex conversion utility by using a command similar to Example 2 at a DOS prompt (or in a DOS batch file).

### Example 2. Invoking hex55.exe

```
Hex55 serial8bin.cmd
```

The resulting binary file (boot table) can be used by the PC host to bootload the application.

## 6 USB Boot Using the Thesycon (USBIO) Driver and Demo Application

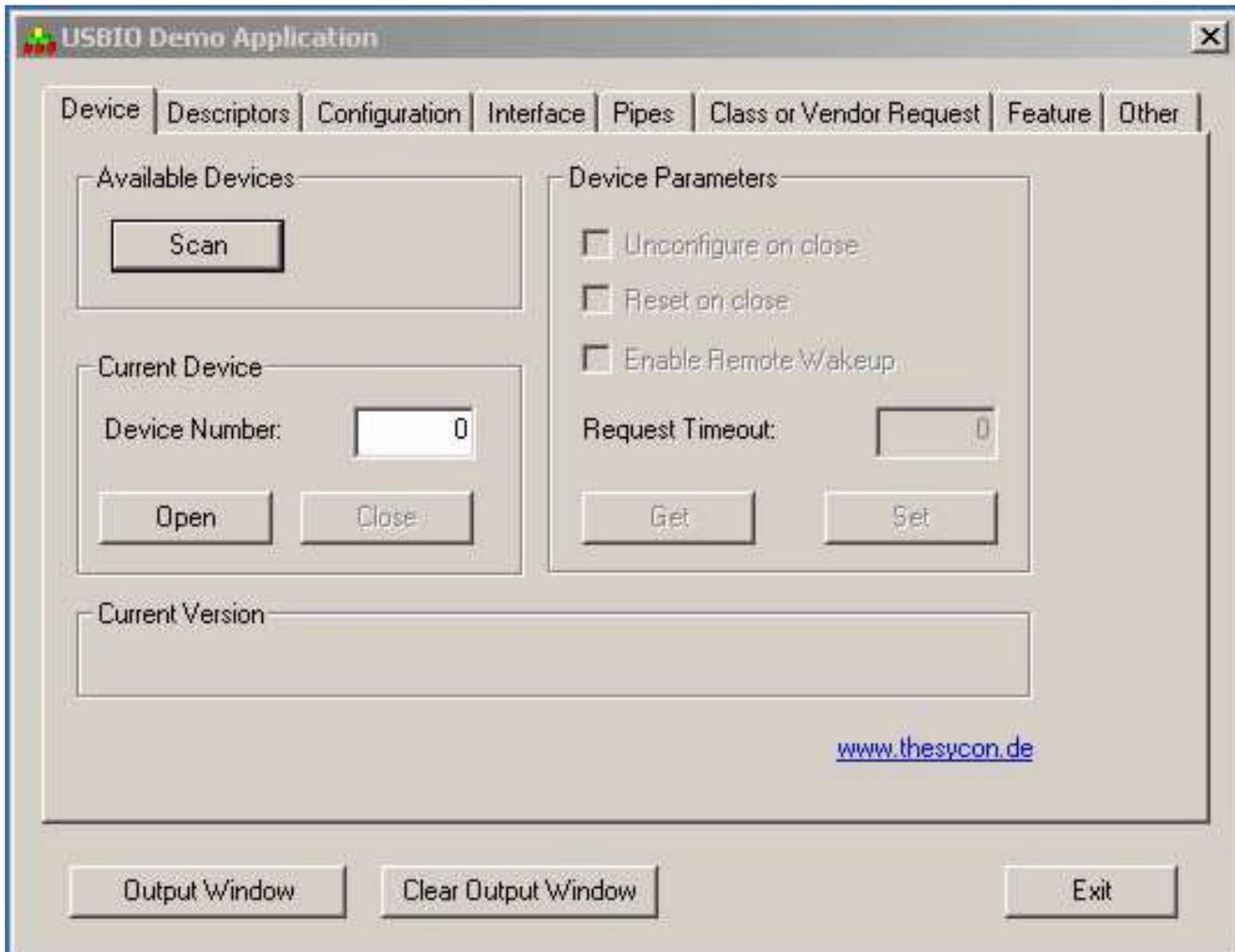
After the binary boot table is generated, an application on the PC (or other USB host) can perform the bootload with the C5506/C5507/C5509/C5509A. If you have a high level of familiarity with USB standard, you can write your own host-side driver if desired. The information necessary is provided in [Table 1](#).

As an alternative, a sample PC-side USB driver (demo version) and demo application is available from a third party called Thesycon.

The steps to complete the USB bootload using the Thesycon driver are given below:

1. Download and install the Thesycon (USB demo) driver.
2. Set up the DSP to select the USB-boot mode (BOOTM[3:0] pins = 0010b).
3. Power up and reset the DSP, and connect the USB cable to the host PC.

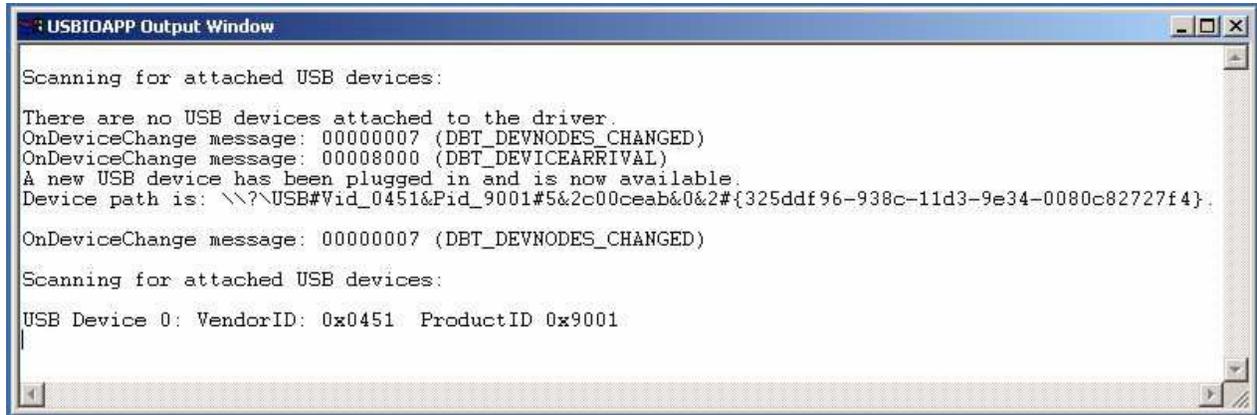
4. Open the USBIO application that came with the Thesycon USB driver. This will produce a window similar to [Figure 3](#).



**Figure 3. USB Demo Application Initial Dialog Box**

5. Open an output window by clicking on the Output Window button.

6. Select the Scan button on the Device tab. In the Output Window, you should see USB device 0:, VendorID: 0x0451 and ProductID: 0x9001. The output window should look similar to [Figure 4](#). This output indicates that the host has successfully enumerated the DSP USB peripheral. If the output window does not display a message similar to [Figure 4](#), check the connections and begin again.



```

: USBIOAPP Output Window
Scanning for attached USB devices:
There are no USB devices attached to the driver.
OnDeviceChange message: 00000007 (DBT_DEVNODES_CHANGED)
OnDeviceChange message: 00008000 (DBT_DEVICEARRIVAL)
A new USB device has been plugged in and is now available.
Device path is: \\?\USB#Vid_0451&Pid_9001#5&2c00ceab&0&2#{325ddf96-938c-11d3-9e34-0080c82727f4}.
OnDeviceChange message: 00000007 (DBT_DEVNODES_CHANGED)
Scanning for attached USB devices:
USB Device 0: VendorID: 0x0451 ProductID 0x9001
  
```

**Figure 4. Output Window in Step 6**

7. Open the device by selecting the Open button.

- Go to the Configuration tab, enter 1 in the Configuration Descriptor Index box and select the Set Configuration button, as shown in [Figure 5](#).

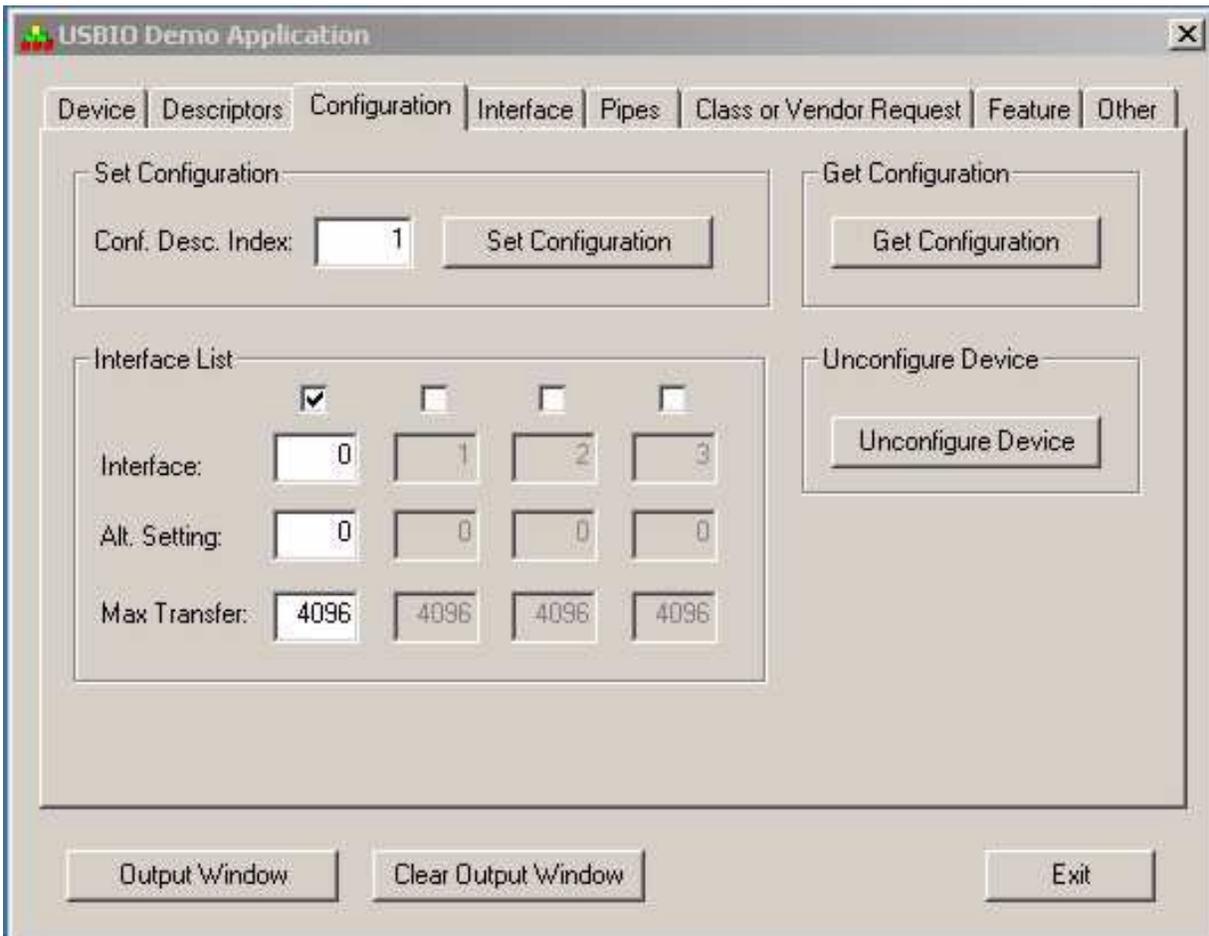
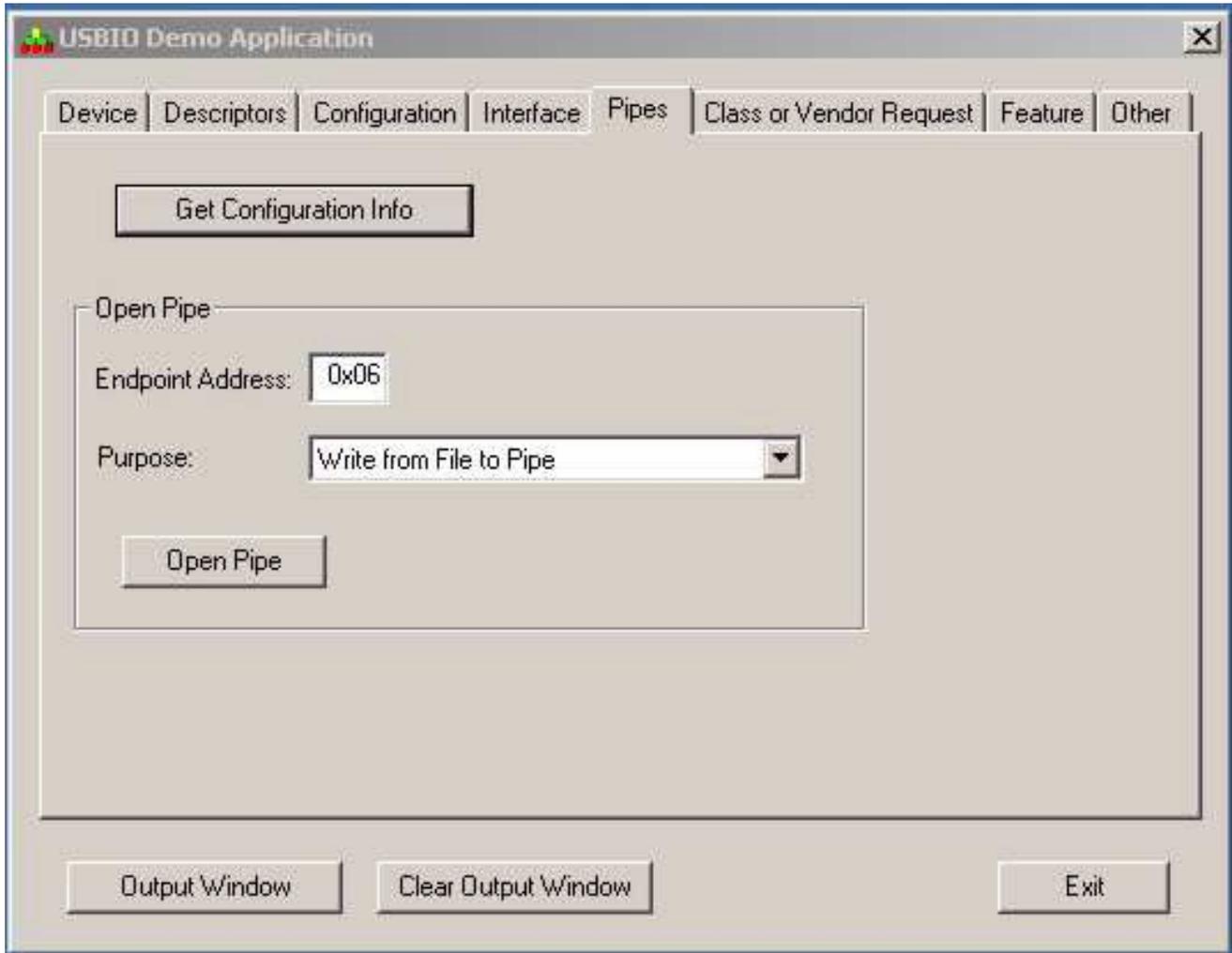


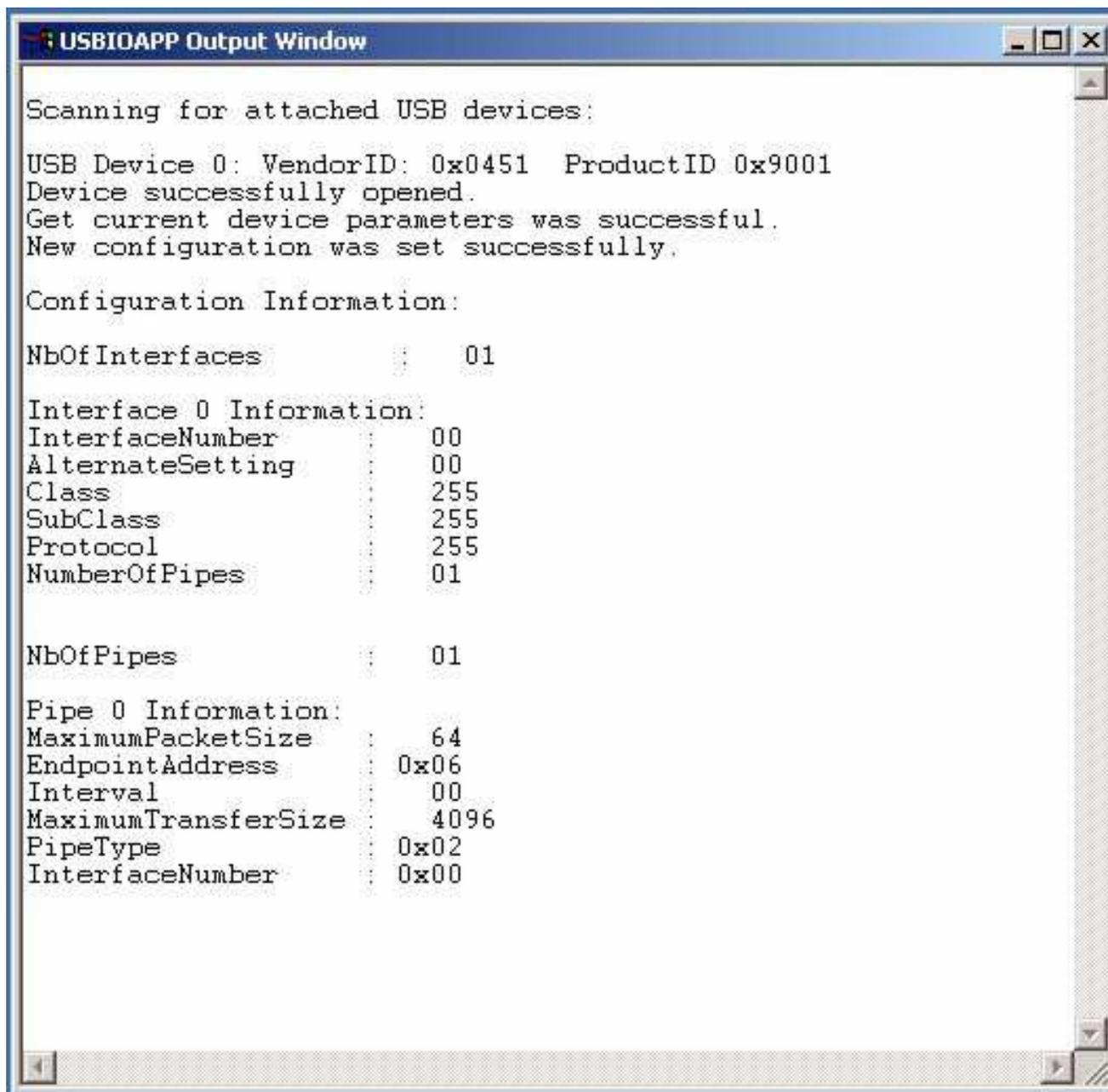
Figure 5. USB Demo Application in Step 8

- Go to the Pipes tab (shown in [Figure 6](#)) and click on the Get Configuration Info button. The output window should appear similar to the one shown in [Figure 7](#).



**Figure 6. USB Demo Application in Step 9**

- Click the Open Pipe button. Enter Endpoint Address 0x06. Select *Write from File to Pipe* in the Purpose field and open a communication pipe to endpoint 0x06 by clicking the Open Pipe button. This will bring up the dialog box, as shown in [Figure 8](#).



```

USBIOAPP Output Window
Scanning for attached USB devices:
USB Device 0: VendorID: 0x0451 ProductID 0x9001
Device successfully opened.
Get current device parameters was successful.
New configuration was set successfully.

Configuration Information:
NbOfInterfaces      :    01

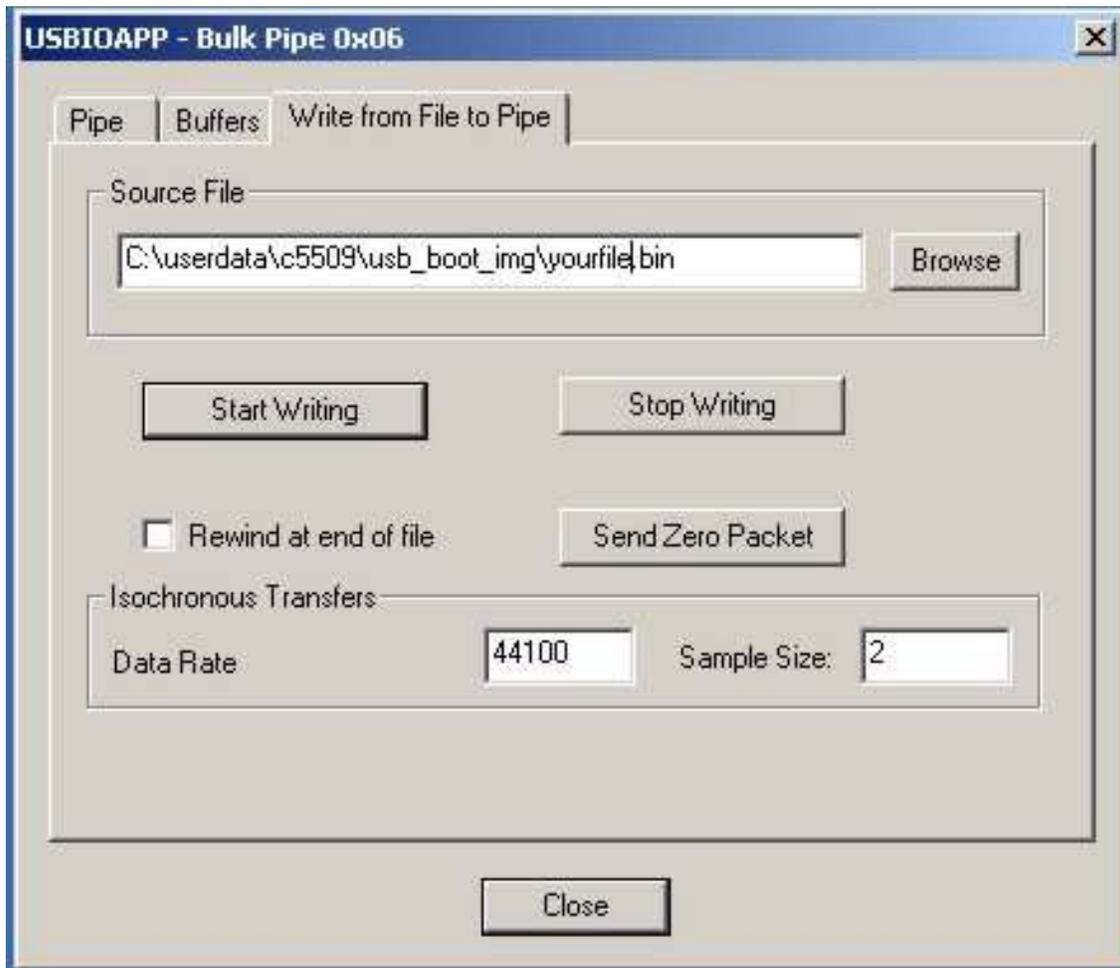
Interface 0 Information:
InterfaceNumber     :    00
AlternateSetting    :    00
Class               :   255
SubClass            :   255
Protocol            :   255
NumberOfPipes       :    01

NbOfPipes           :    01

Pipe 0 Information:
MaximumPacketSize   :    64
EndpointAddress     : 0x06
Interval            :    00
MaximumTransferSize :   4096
PipeType            : 0x02
InterfaceNumber     : 0x00
    
```

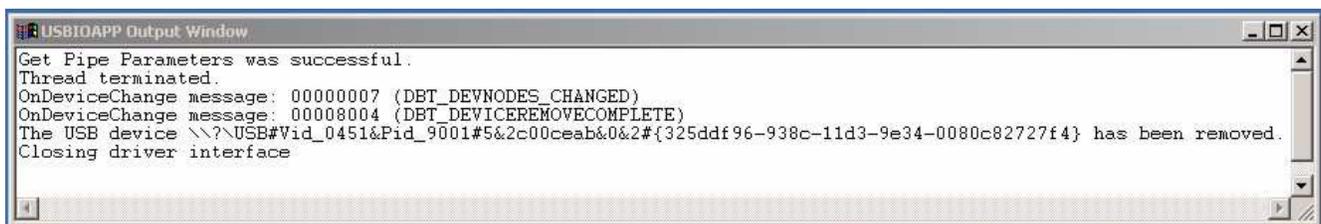
**Figure 7. Output Window Following Step 9**

11. Select the Write from File to Pipe tab in the dialog box shown in [Figure 8](#). Select the source file to be loaded, and make sure the Rewind at end of file option is not checked. Then select the Start Writing button.



**Figure 8. USB Demo Application in Step 11**

The file is sent to the DSP and, when the transfer is complete, the DSP automatically disconnects itself from the host and starts executing the loaded code. An output window similar to the one shown in [Figure 9](#) should appear.



**Figure 9. Output Window for Step 11**

At this point, the USB bootloader is completed, and the DSP begins execution of the downloaded application. If a message similar to [Figure 9](#) does not appear, then check the connections and repeat the entire process.

## 7 Example Boot Table: `boot_img.bin`

A simple demo project is available for download with this document from the following url: <http://www-s.ti.com/sc/techlit/spra840.zip>. This demo employs a simple code segment that toggles the XF pin on the C5506/C5507/C5509/C5509A. All of the files necessary to build the project and generate the boot table are included.

## 8 Summary of USB Boot-Load Steps

The steps involved in using the USB bootloader on the C5506/C5507/C5509/C5509A are summarized below:

1. Build the COFF output file for the desired application using Code Composer Studio.
2. Convert the COFF (.out) file to a binary boot table file by one of the methods described in [Section 5](#).
3. Set the BOOTM[3:0] on the DSP to select the USB-boot mode.
4. Power up and reset the DSP, and connect the USB cable to the host.
5. Establish a communication pipe with the DSP (see [Section 7](#)) using the Thesycon (or custom-written) USB driver and host application.
6. Download the binary boot table and the DSP will automatically run the application when the load is complete.

## 9 Customizing the USB Bootload

In certain applications it may be necessary to customize the USB bootload for a particular application requirement. Modifications to the bootloader are beyond the scope of the novice USB user, and require a deeper understanding of the USB standard. This section provides some information that may be useful in the customization process.

### 9.1 Vendor Identification (VID) and Product Identification (PID)

The USB standard requires each USB slave device to have a vendor identification (VID) and product identification (PID) that are registered with the USB Implementers Forum (USBIF) group. Since the bootloader target software resides in the on-chip ROM and cannot be changed, each C5506/C5507/C5509/C5509A device uses the same VID and PID for the standard bootloader. The bootloader uses the VID and PID to the values shown in [Table 1](#).

After the USB bootload is complete, the bootloader program disconnects the USB peripheral from the USB network and transfers program control to the newly loaded user-application. At this point, the user-application can reconfigure the USB peripheral (including the VID and PID values) for application-specific use, and then reconnect to the USB network using the new configuration. Note that the evaluation copy of the Thesycon USB host driver, mentioned in this document, is restricted to function with the VID and PID values of [Table 1](#). A fully licensed version of the driver is required for use with different VID and PID values.

The source code for the USB-boot mode can be provided as a reference. However, Texas Instruments cannot provide extensive support on any modifications to this program. The USB-boot mode of the standard C5506/C5507/C5509/C5509A bootloader is implemented using the USB Chip Support Library from Texas Instruments. This library provides a set of functions for using the USB peripheral of the C5506/C5507/C5509/C5509A DSP. For more information on this library, see the *TMS320C55x CSL USB Programmer's Reference Guide* ([SPRU511](#)).

## 10 References

- *Using the TMS320VC5503/C5506/C5507/C5509/C5509A Bootloader* ([SPRA375](#))
- USB Specification, <http://www.usb.org>
- *TMS320C55x DSP Peripherals Reference Guide* ([SPRU317](#))
- *TMS320C55x Assembly Language Tools User's Guide* ([SPRU280](#))
- *TMS320C55x CSL USB Programmer's Reference Guide* ([SPRU511](#))
- Thesycon USB Driver Manual - Universal USB Device Driver, [www.thesycon.com](http://www.thesycon.com)

## 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](http://www.ti.com/legal/termsofsale.html)) or other applicable terms available either on [ti.com](http://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  
Copyright © 2019, Texas Instruments Incorporated