TMS320DM644x DVEVM Windows CE v5.0 BSP Bootloader

User's Guide

Literature Number: SPRUEW1
March 2007

Texas Instruments

Contents

List of Figures

List of Tables

Read This First

About This Manual

This document accompanies the release of Windows® CE 5.0 BSP for DaVinci™ evaluation module (EVM) board.

Purpose and Scope

This document provides information about the Windows CE 5.0 bootloader for DaVinci EVM. The document illustrates various features incorporated in this release of bootloader, the build and flash procedure, and lists limitations of this release.

The user should have access to Microsoft Platform Builder 5.0 and should be familiar with its usage. Also, the user should be familiar with building the OS image using the Platform-Builder 5.0 IDE and have a valid OS image based on DaVinci BSP.

Notational Conventions

This document uses the following conventions.

- Backward slashes are used as pathname delimiters for filenames.
- Catalog->Third Party refers to the Catalog Window Tree Items in the Platform Builder IDE.
- All the shell commands are in courier new font.
- Menu commands are depicted using the following notation ***menu name > menu command***.

Terms, Acronyms and Descriptions

Table 1. Terms, Acronyms and Descriptions

Term	Description
BLCOMMON	Bootloader Common Architecture
CF	Compact Flash
DHCP	Dynamic Host Configuration Protocol
DVEVM	Digital Video Evaluation Module
IOCTL	Input Output Control
LCD	Liquid Crystal Display
MMU	Memory Management Unit
OAL	OEM Adaptation Layer
OEM	Original Equipment Manufacturer
PQ	Production Quality
UBL	User bootloader (Tiny bootloader that can be picked by Internal ROM bootloader)
USB	Universal Serial Bus

Related Documentation from Texas Instruments

The following documents describe the TMS320DM644x DVEVM Windows CE v5.0 BSP.

SPRUEV9 — *TMS320DM644x DVEVM Windows CE v5.0 BSP Codec Engine Users Guide.*

Provides information about the release contents of Windows CE 5.0 BSP for DaVinci-based DVEVM. The document illustrates various components that are part of this release, the procedure to install this release on to the host system, and the limitations of the release.

SPRUEW1 — ***TMS320DM644x DVEVM Windows CE v5.0 BSP Bootloader Users Guide.***

Provides information about the Windows CE 5.0 bootloader for DaVinci EVM. The document illustrates various features and the build and flash procedures.

SPRUEW0 — ***TMS320DM644x DVEVM Windows CE v5.0 BSP DSP/BIOS Link Users Guide.***

Describes the usage of the DSP/BIOS Link binaries provided along with the Windows CE 5.00 BSP for the Davinci EVM platform and the integration procedures in a given Windows CE image.

SPRUEV8 — ***TMS320DM644x DVEVM Windows CE v5.0 Codec Engine Binary Users Guide***

Provides information on the build procedure for the codec engine samples on Windows CE 5.0 platform.

SPRS283 — ***TMS320DM6446 Digital Media System-on-Chip Data Manual*** (SPRS283)

The TMS320DM6446 (also referenced as DM6446) leverages TI's DaVinci™ technology to meet the networked media encode and decode application processing needs of next-generation embedded devices.

Trademarks

DaVinci, XDS510, Code Composer Studio IDE are trademarks of Texas Instruments.

ARM is a registered trademark of ARM Limited.

Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries.

TMS320DM644x DVEVM Windows CE v5.0 BSP Bootloader

1 Overview

The bootloader is a utility that is used to place the OS image into memory and then jump to the OS startup routine. This release of bootloader obtains the OS image through the Ethernet and loads the OS image in RAM or stores it in flash memory.

1.1 Features

Following are the key features of this release of bootloader:

- Based on the BLCOMMON framework of Microsoft
- Supports persistent storage of following boot parameters
  - IP Address
  - Subnet mask
  - Gateway address
  - TFTP server IP address
  - Device name
  - Boot mode
- Supports download using Platform Builder 5.0 IDE or any general-purpose TFTP server program
- Supports static configuration of the IP address as well as using the DHCP for IP address configuration
- Supports the serial console with commands for:
  - Changing the boot parameters and store them in flash
  - Storing the OS image into the flash
  - Memory dump and copy

1.2 Bootloader Files

Source files for the bootloader are bundled along with the BSP files under the path:

$(_WINCEROOT)\PLATFORM\DAVINCI\SRC\BOOTLOADER\EBOOT

While building the bootloader, there are some source modules being shared between the bootloader and the OAL. All such source modules are placed in the path:

$(_WINCEROOT)\PLATFORM\DAVINCI\SRC\DM644x and
$(_WINCEROOT)\PLATFORM\DAVINCI\SRC\Common

2 Bootloader Usage

2.1 Building Bootloader

This section lists the steps required to build any type of bootloader:

Step 1. Open the command window console on the host system.

Step 2. Navigate to the platform builder installation root directory.

Step 3. Set the environment variable _WINCEROOT to the installation root directory.

Execute the following command if the Platform builder is installed in e:\Wince500 :

```
set _WINCEROOT=E:\wince500
```

Step 4. Navigate to the directory:
$(WINCEROOT)\Public\common\oak\misc

Step 5. Execute the following command:

```
wince ARMV4I CEBASE DAVINCI
```

Step 6. Navigate to:
$(_WINCEROOT)\PLATFORM\DAVINCI

Step 7. Execute the command:

```
sysgen
```

Step 8. Navigate to the directory:
$(_WINCEROOT)\PLATFORM\COMMON
and build the directory with the *build -c* command.

Step 9. Navigate to the directory:
$(_WINCEROOT)\PLATFORM\DAVINCI\SRC\Bootloader

Step 10. Execute the following batch file on the command prompt:
buildboot.bat

2.2 Verification of the Bootloader

To verify the bootloader has been built, execute the following steps:

Step 1. On the command console, navigate to the directory:
$(_WINCEROOT)\PLATFORM\DAVINCI\SRC\BOOTLOADER\Eboot\RELEASE"

Step 2. Execute the command *viewbin -r -t eboot.bin*.

Step 3. Verify the record start address as described in the following subsections. Please note that the actual address may vary depending on the PB updates installed.

Example 1. Viewbin for Boot Flash

```
D:\WINCE500\PLATFORM\DAVINCI\Src\Bootloader\EBOOT\Release>viewbin -r -t eboot.bin
ViewBin... eboot.bin
Image Start = 0x02000000, length = 0x00019884
               Start address = 0x02001000
Checking record #4 for potential TOC (ROMOFFSET = 0x7A020000)
Found pTOC  = 0x87ff8e2c
ROMOFFSET = 0x7A020000

ROMHDR ----------------------------------------
    DLL First           : 0x02000000
    DLL Last            : 0x02000000
    Physical First      : 0x87FE0000
    Physical Last       : 0x87FF9884
    RAM Start           : 0x87FDA000
    RAM Free            : 0x87FDF000
    RAM End             : 0x87FE0000
    Kernel flags        : 0x00000000
    Prof Symbol Offset  : 0x00000000
    Num Copy Entries    :          1
    Copy Entries Offset : 0x87FF8EA0
    Num Modules         :          1
    Num Files           :          0
    MiscFlags           : 0x00000002
    CPU                 :     0x01c2 (Thumb)
    Extensions          : 0x00000000

COPY Sections ---------------------------------
    Src: 0x87FF95B0   Dest: 0x87FDA000   CLen: 0x2D3      DLen: 0x4DA0

MODULES ---------------------------------------
```

Example 1. Viewbin for Boot Flash (continued)

```
     2/20/2007  09:38:31      104448  nk.exe

FILES ---------------------------------------
Done.
```

2.3 Setup for Flashing Bootloader

The following subsections describe the procedure to set up the hardware and JTAG environment for flashing the bootloader into the NOR Flash.

2.3.1 Hardware Setup

The DaVinci EVM uses a high-density 20-pin CTI connector as a JTAG header rather than traditional TI 14-pin header. Therefore, the 14-pin to 20-pin adapter is required to connect the emulator to the JTAG port.

For setting up the DVEVM board for flashing the bootloader, the following must be verified.

Step 1. Ensure that Jumper J4 is set to select the NOR Flash on CS2. That is, pins 1 and 2 are shorted in Jumper J4.

Step 2. Ensure the DIP Switch S3 settings are as shown in Table 2.
Verify that these settings are as follows:

a. Set COUT0 = ON and COUT1 = OFF, to select the NOR boot.

b. Set COUT2 = ON, to set 16 bit data bus width.

Table 2. DIP Switch S3 Settings

1	2	3	4	5	6	7	8	9	10
ON	OFF	ON	OFF	ON	ON	ON	ON	ON	OFF

Step 3. Ensure that the serial cable is connected to the board and the other end connected to the host PC.

Step 4. Ensure that the Ethernet cable is connected to the board and the other end connected to the LAN HUB or switch.

Step 5. Connect the JTAG emulator header to the DaVinci EVM board via the 20-pin to 14-pin adapter.

Step 6. Connect the JTAG emulator to the host PC and power up the emulator.

For customers using the BlackHawk USB560 Emulator, see Section 2.4. If the emulator being used in XDS510™ USB software, then see Section 2.5.

2.4 Code Composer Studio Configuration Setup With BlackHawk USB560M Emulator

The following steps describe the procedure to se tup the Code Composer Studio IDE™ for use with the BlackHawk USB560M Emulator.

Step 1. Ensure that the JTAG Emulator header is securely connected to the appropriate port on DVEVM.

Step 2. Ensure USB connectivity between the host PC and the emulator. Also ensure that the emulator is powered ON.

Step 3. Open the Code Composer Studio Setup application.

Step 4. In the middle tab, locate the configuration corresponding to the BlackHawk USB560M for DM6446 Emulator as in Figure 1.

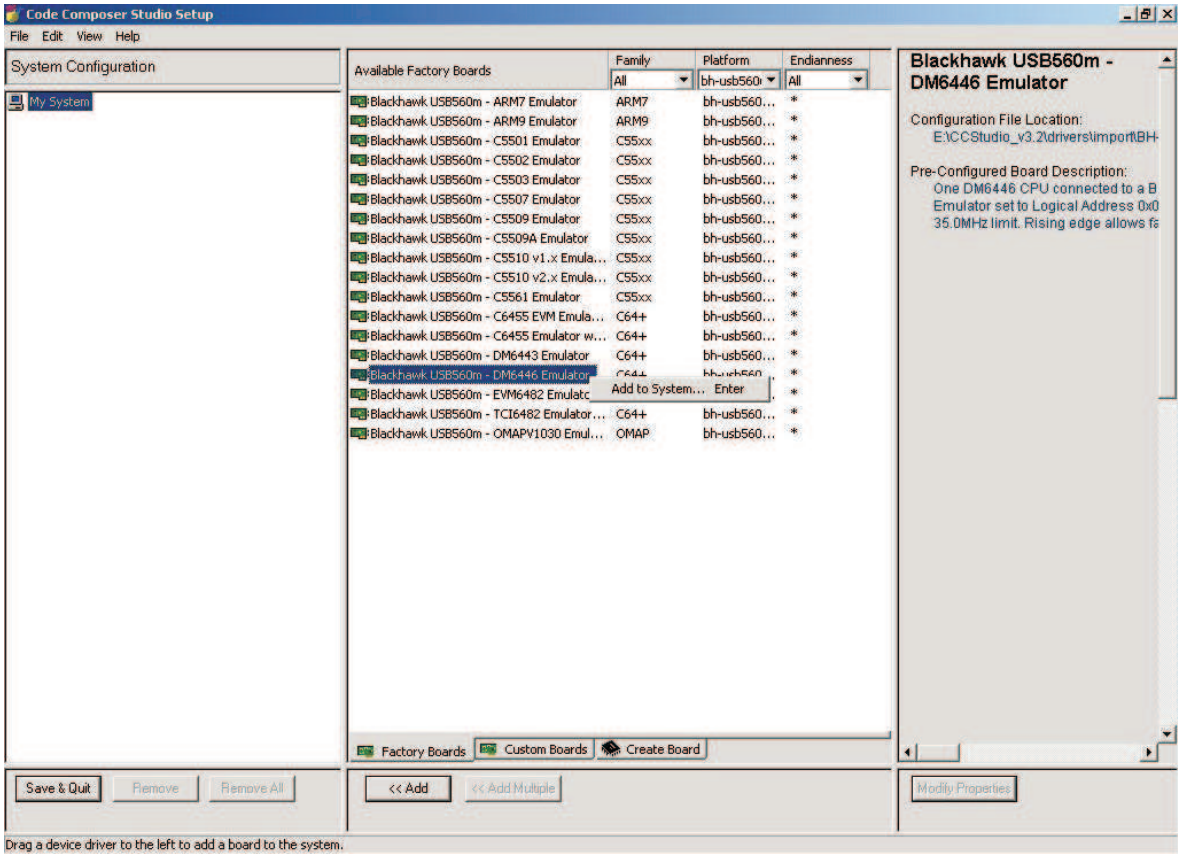

Figure 1. Code Composer Studio Setup Configuration Selection

Step 5. Right-click on this configuration and select *Add to system* as shown in the Figure 1.

On adding this configuration to the system, the Code Composer Studio Setup windows will be seen as shown in Figure 2

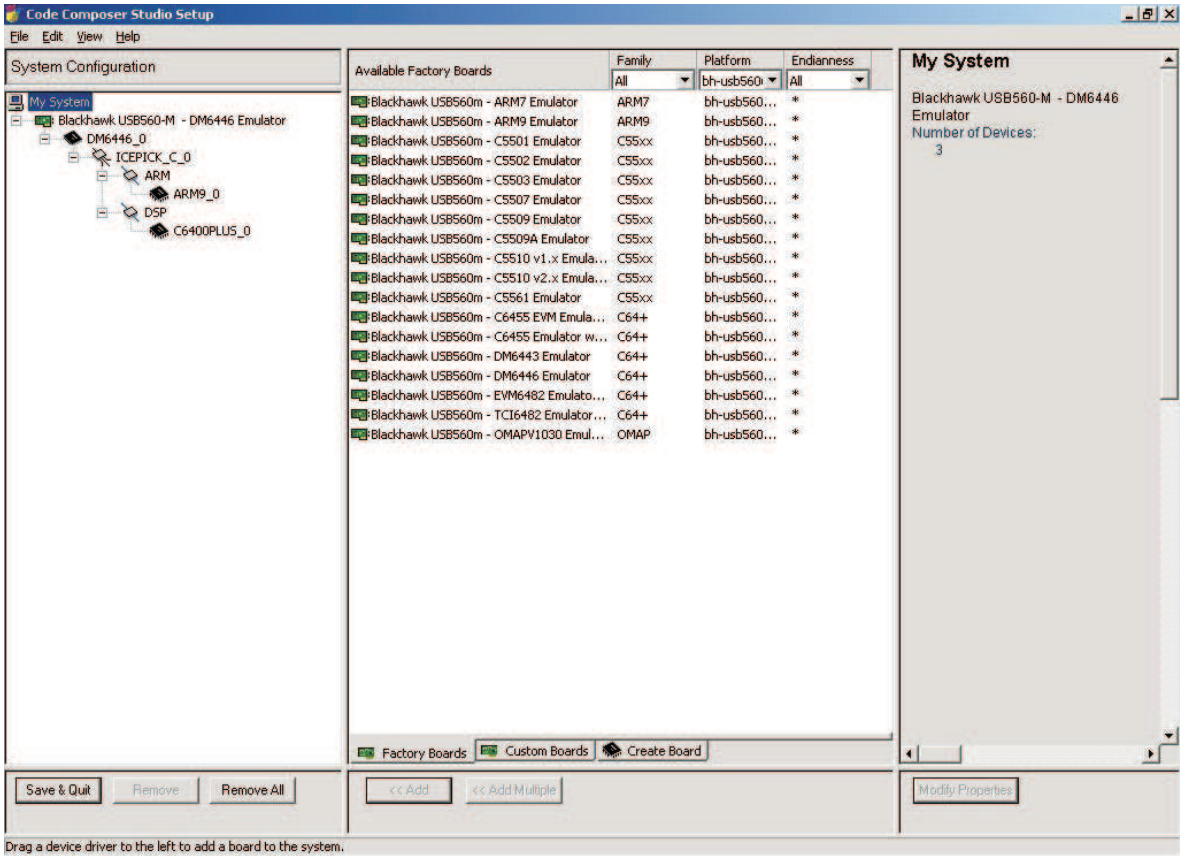

Figure 2. BlackHawk USB560M - DM6446 Emulator Selected

Step 6. Set up the appropriate GEL files for ARM® and DSP by right-clicking on the corresponding item and selecting properties (as shown in Figure 3).

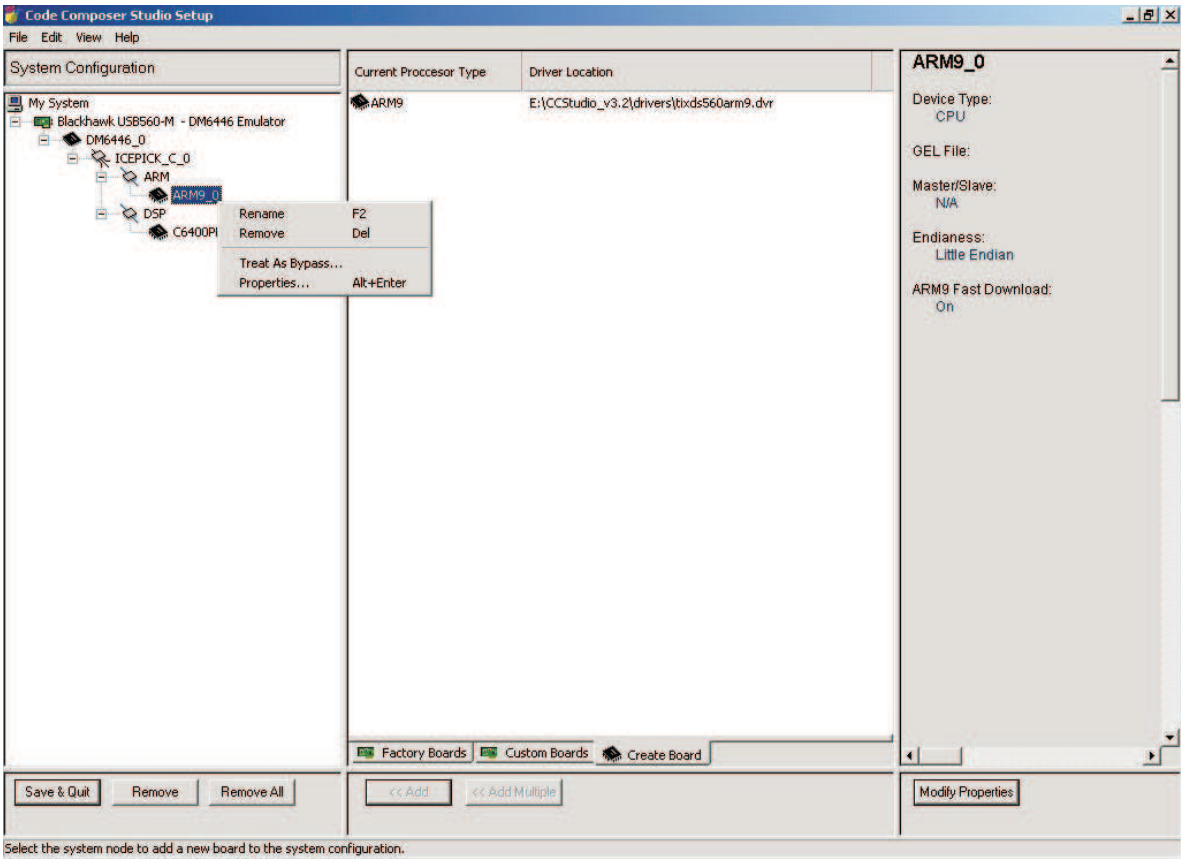

Figure 3. Code Composer Studio Configuration Gel File Setup

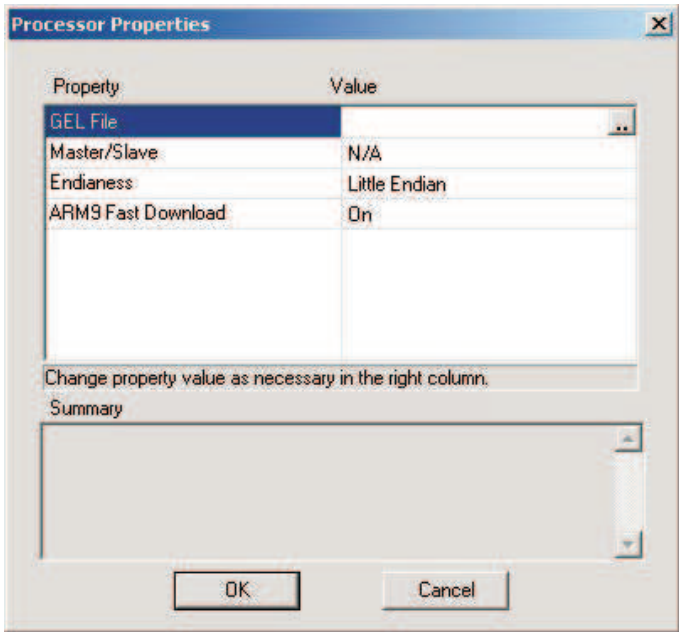

Figure 4. Gel File Setup

Step 7. The dialog box for setting up the Gel file appears in Figure 4. Navigate to the CCS_3.2.2/Boards/Davincievm/gel for the correct GEL file. Use the DaVinciEVM_arm.gel for ARM and DaVinciEVM_dsp.gel for DSP components.

Step 8. Save and exit the Code Composer Studio setup application. This would prompt you with a dialog box to start the Code Composer Studio on exit. Select *Yes* to start the Code Composer Studio.

Step 9. Select Ignore in this dialog box if the error dialog box appears as shown in Figure 5. This will launch the CCStudio:Parallel Debug Manager as shown in Figure 6.

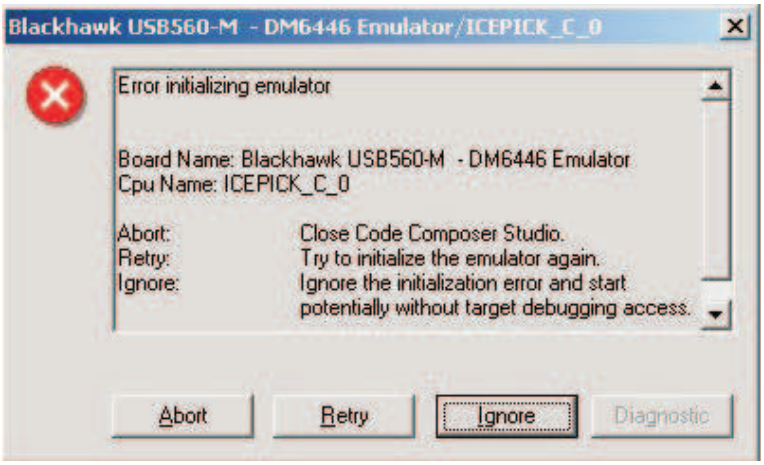

Figure 5. Code Composer Studio Launch Error Message

Note: This is a known limitation with the BlackHawk emulator and the Code Composer Studio startup.

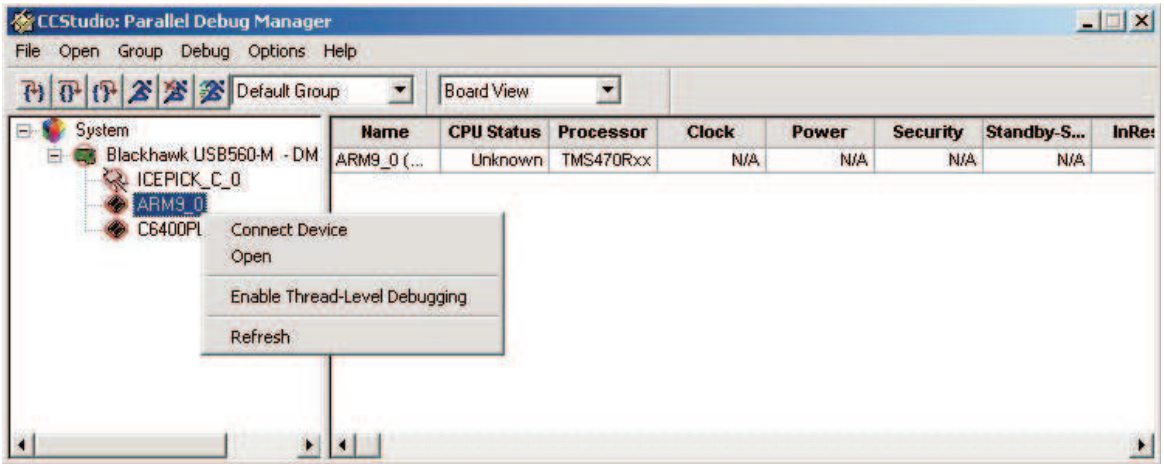

Figure 6. Parallel Debug Manager Session

Step 10. Right-click on the ARM9_0 to get the pop-up menu as shown in Figure 6. Select the *Open* menu item. This launches Code Composer Studio IDE for ARM.

2.5 Code Composer Studio Configuration Setup With XDS510 USB Emulator

The following steps describe the procedure to set up Code Composer Studio, with XDS510 USB:

Step 1. Open the Code Composer Studio Setup to select the appropriate emulator setup.

Step 2. Add the *Davinci EVM via XDS510USB Emulator* to *My System Configuration* in the left pane, if you have XDS510USB emulator.

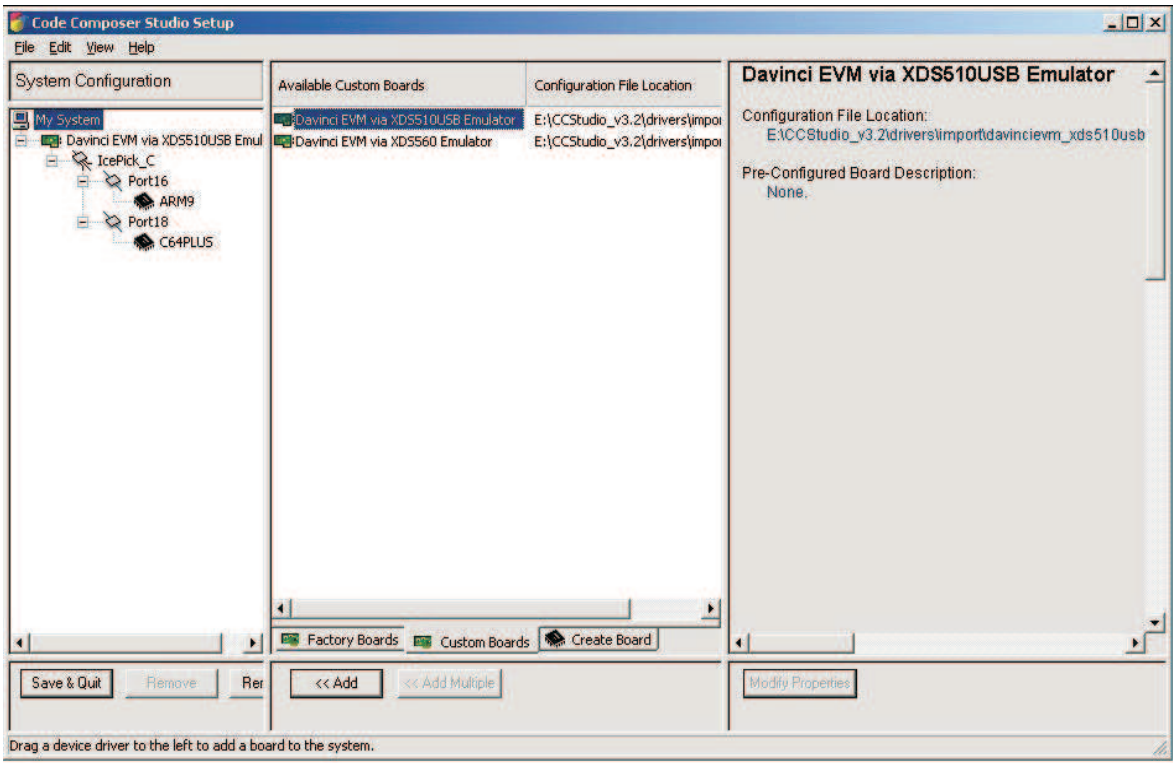

Figure 7. Code Composer Studio Setup Emulator Selection

Step 3. Click the *Save & Quit* button, present in the bottom of the left pane. This will query you the start Code Composer Studio on exit as shown in Figure 8.

Step 4. Select *yes* to start the Code Composer Studio window.

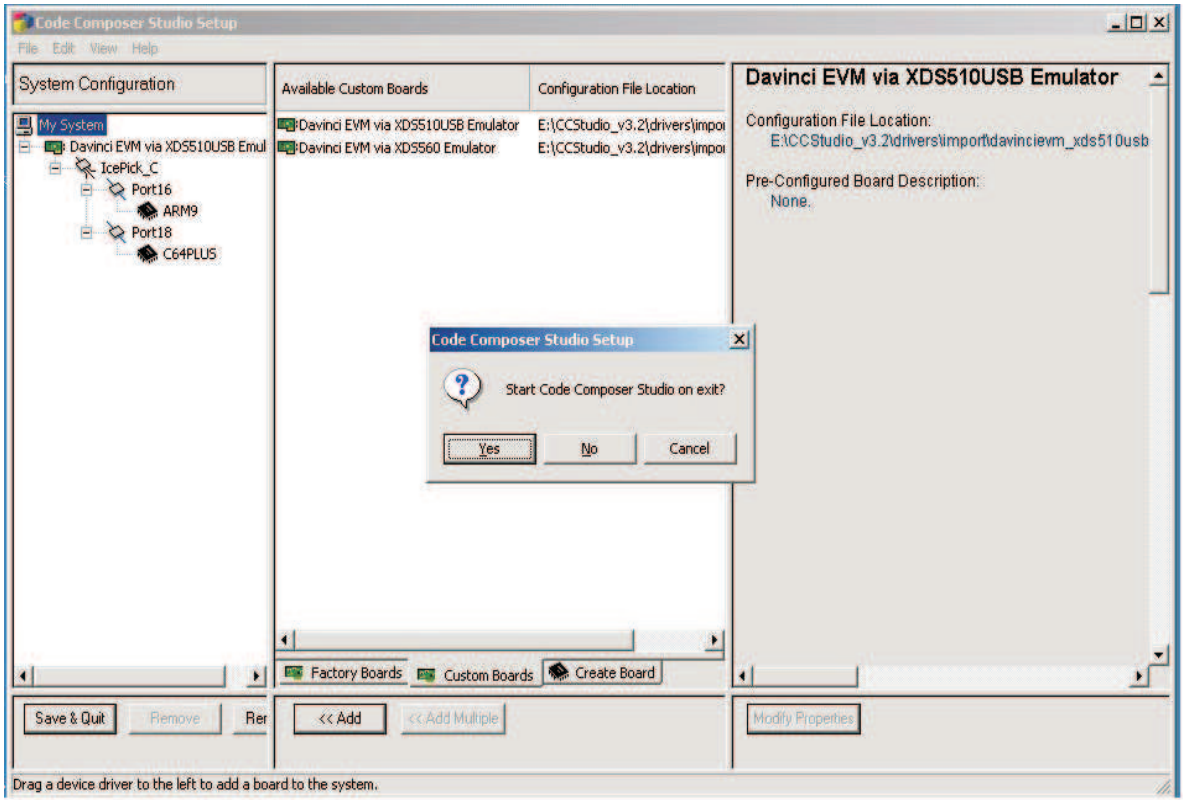

Figure 8. Code Composer Studio Setup Exit

2.6 Flashing the Bootloader into NOR Flash

The Code Composer Studio project, for flashing the bootloader into the NOR Flash, is shipped with the BSP files. This project can be located in the path:
$(_WINCEROOT)\PLATFORM\DAVINCI\TOOLS\davinci_ubl\boot\tools\flashwriter_arm\build

The following steps illustrate the procedure to flash the bootloader into the NOR Flash of the DaVinci EVM board.

Step 1. Open the Code Composer Studio project from the Code Composer Studio 3.2 IDE for ARM only and compile the same. Load the output file (*.out) file on to the target board and execute it.

Step 2. Enter the path of the eboot.nb0 file to be programmed into the prompt dialog window (as shown in Figure 9). This file can be found in the path:

$(_WINCEROOT)\ PLATFORM\DAVINCI\SRC\BOOTLOADER\Eboot\RELEASE

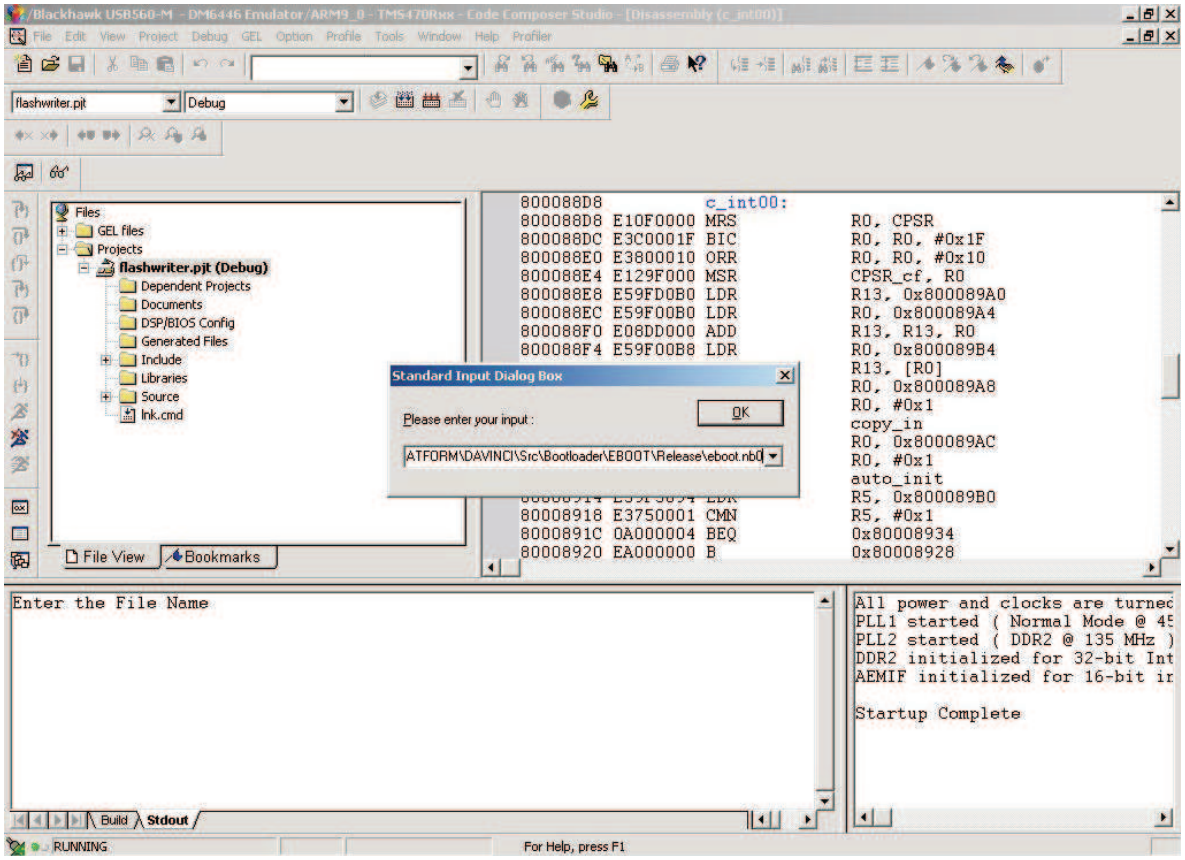

Figure 9. Code Composer Studio Program "File Path" Option

Step 3. Enter the value *0x0000* in the offset field when prompted (as shown in Figure 10).

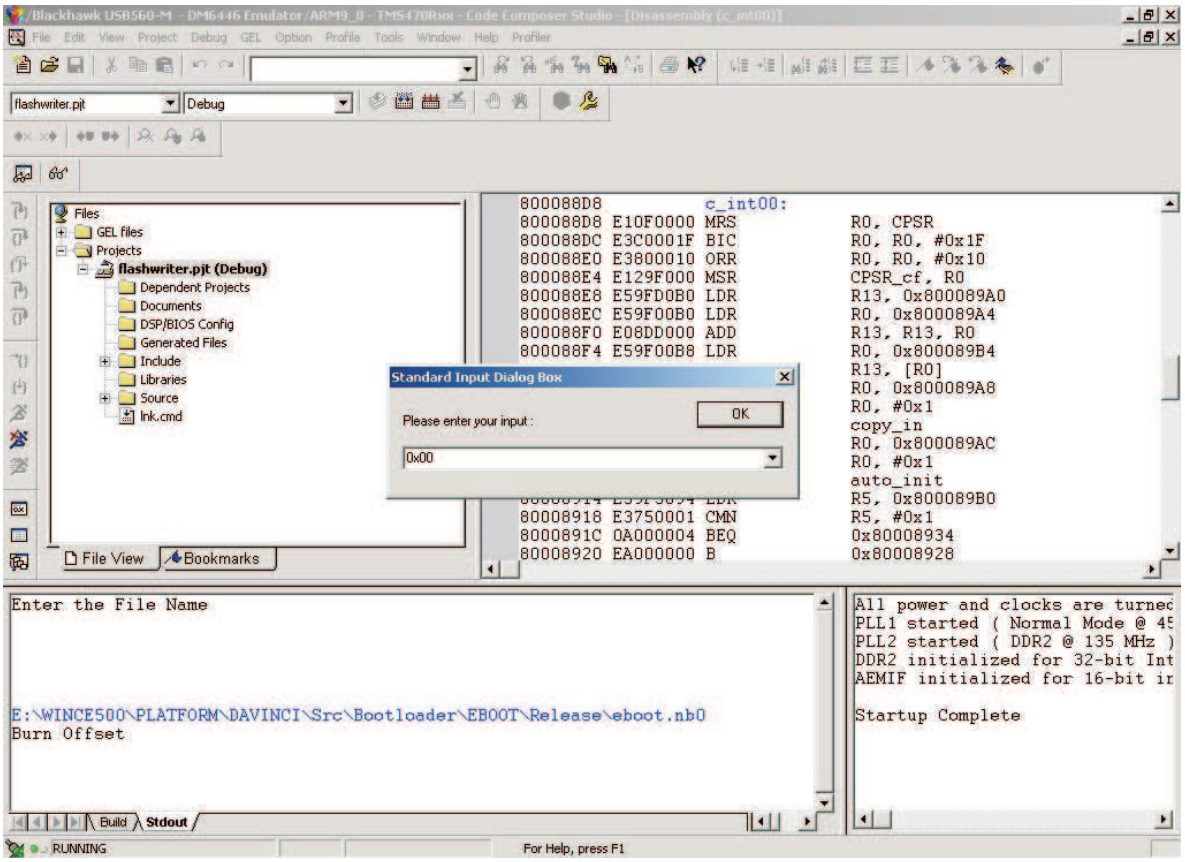

Figure 10. Code Composer Studio Program "Offset" Option

Step 4. Enter *Y* for complete flash erase when prompted (as shown in Figure 11).

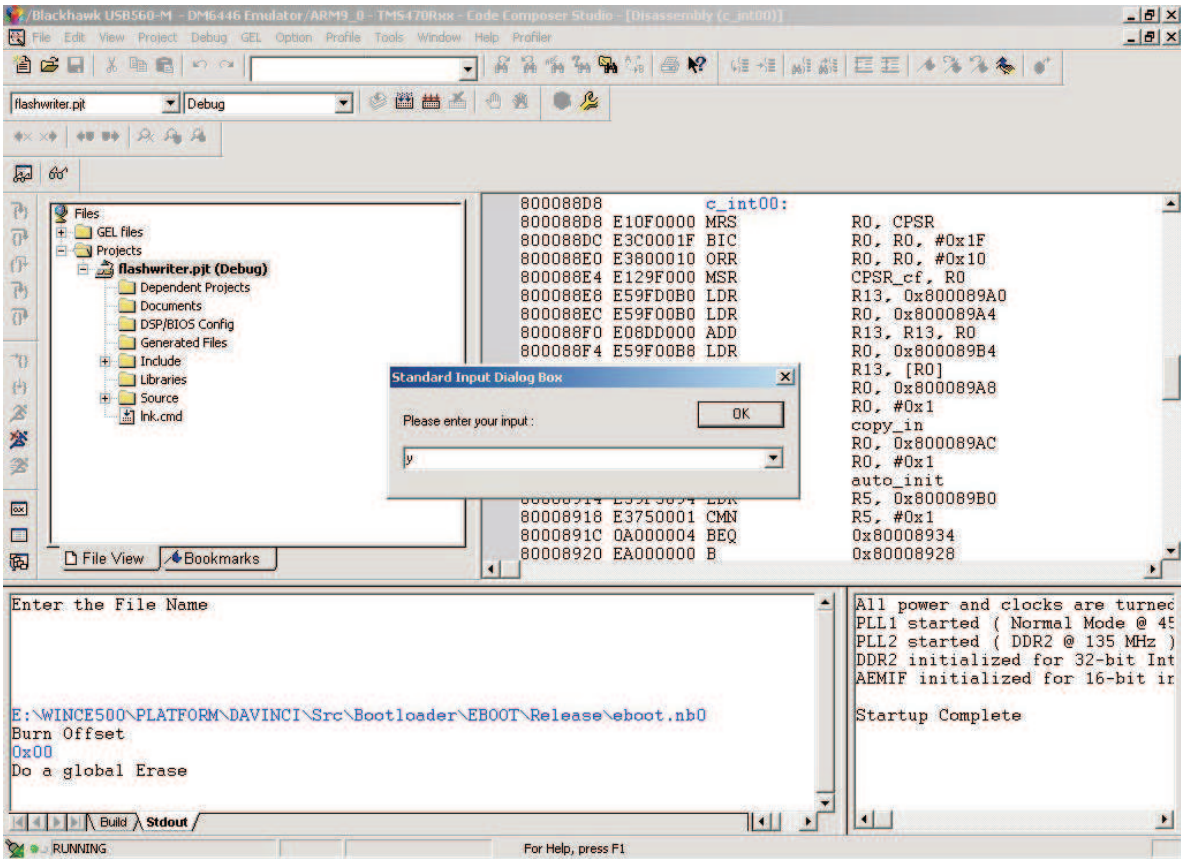

Figure 11. Code Composer Studio Program "Flash Erase" Option

The eboot.nb0 will be programmed into the flash and this may take few minutes. Figure 12 shows the messages that are output on the Output window of Code Composer Studio IDE. If Global erase input is given *Y*, then the Output window is different from the one below.

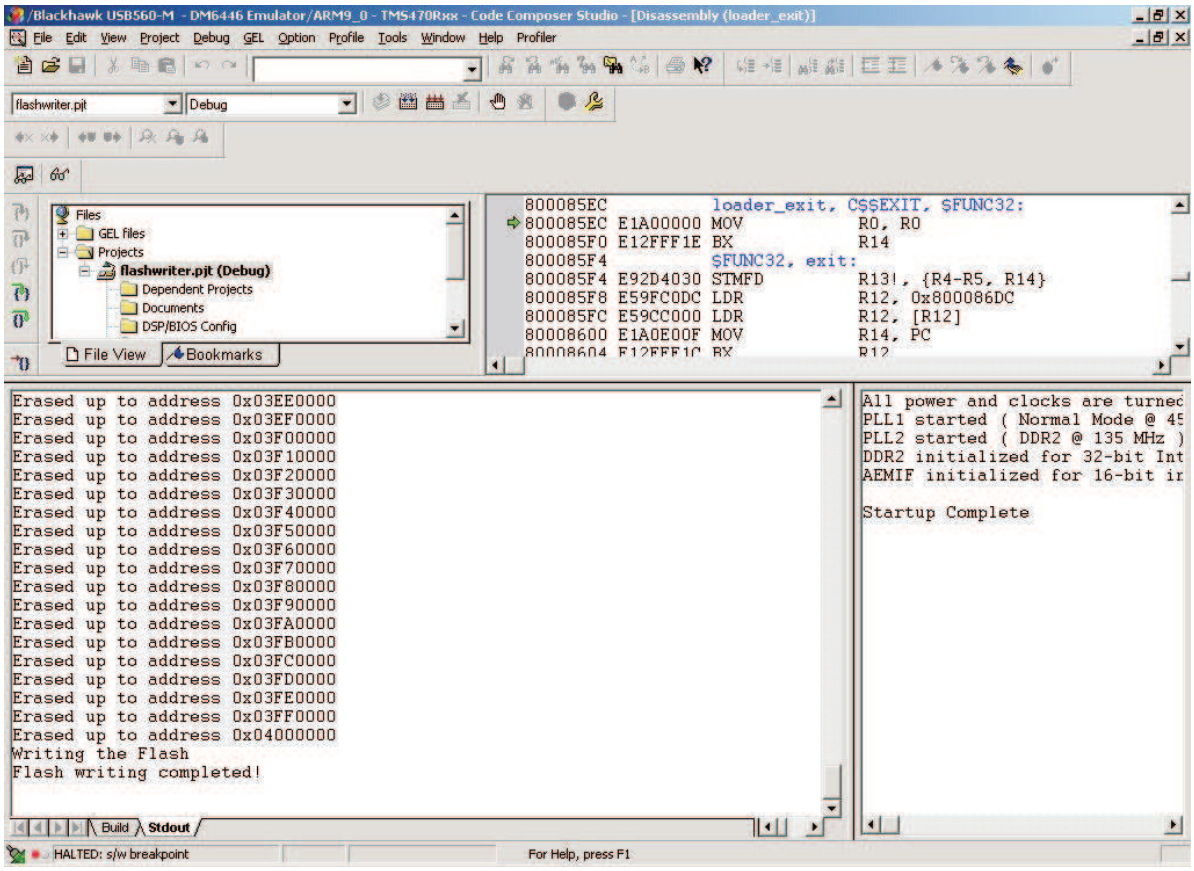

Figure 12. Code Composer Studio IDE Output Window Messages

Step 5. After programming the flash, power OFF the board and disconnect the JTAG emulator. Power ON the board and the output showing the bootloader messages may be seen on the HyperTerminal (as shown below). The HyperTerminal configuration settings for the serial console of the Target are 115200-8-N.

```
INFO:OALLogSetZones: g_oalLogMask: 0xb

Microsoft Windows CE Ethernet Bootloader Common Library Version 1.1 Built Aug  2
 2006 14:01:32

Windows CE 5.0 Ethernet Bootloader 0.2 for DaVinci EVM(Aug  2 2006 18:43:59)
EnvLibBase is 0x02030000 EnvLibEnd is 0x02040000
Env Library with version ENVER1.11 exists in flash
It is being populated to ram copy
I2C will be enabled.
I2C OAR : 0x0000 I2C IMR : 0x0000 I2C STR : 0x0410
I2C CLKL : 0x00dc I2C CLKH : 0x00dc I2C MODE : 0x0020
I2C ExMR : 0x0001 I2C PSC : 0x0002 I2C IVR : 0x0000
I2C PID1 : 0x0105 I2C PID2 : 0x0005

Environment Variables  Respective Values
ENV_SIG                         ENVER1.11
DEVICE_ID                     Davinci1455
IP_GATEWAY                        0.0.0.0
IP_ADDR                           0.0.0.0
SUBNET_MASK                 255.255.255.0
IP_SVR                            0.0.0.0
SVR_SUBNET_MASK             255.255.255.0
BOOT_ADDR                     0X00040000
TFTP_CONFIG                            1 ->TFTP_SVR
```

```
BOOT_MODE                              1 ->BOOT_DOWNLOAD
Press key to abort auto-boot/download sequence (5 seconds)
```

2.7 Downloading an OS Image to RAM Using Bootloader

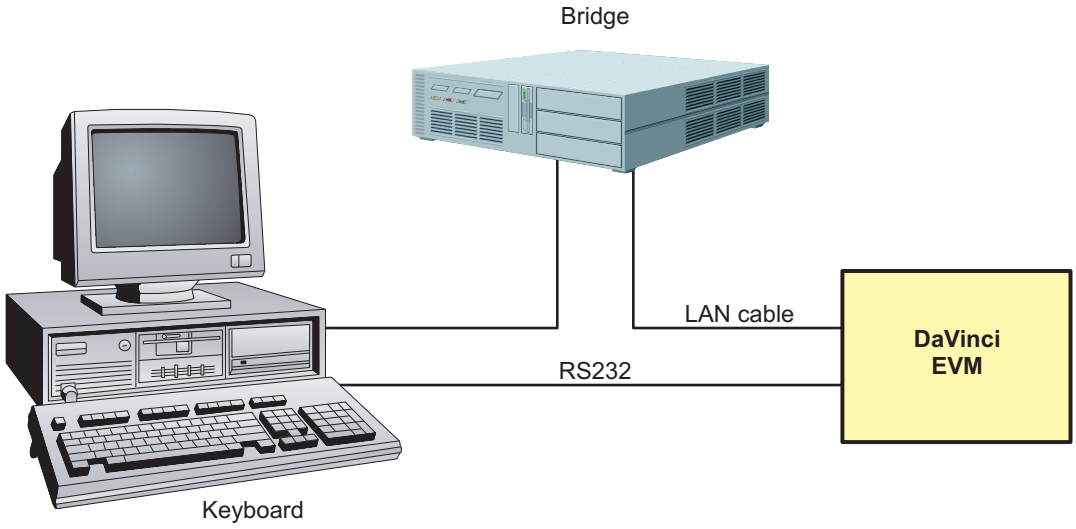

Figure 13. Setup for Image Download

Figure 13 illustrates the setup required for downloading the OS image into the target board using the platform builder IDE. Use the following steps to download the OS image to RAM using the platform builder IDE.

Step 1. Open the project from the platform builder IDE.

Step 2. Ensure that the serial port of the target is connected to COM port of host and the HyperTerminal is started for corresponding COM port. The HyperTerminal setting is:

a. Baud rate of 115200
b. 8-bit data
c. 1 stop bit
d. No parity
e. No flow-control

Step 3. Ensure that both the host and the target are securely connected to an Ethernet HUB/switch.

Step 4. Power up the target board and type the 'Enter' key twice on the serial console prompt (within 5 seconds) to abort auto-boot sequence and enter the menu command mode. If no character is typed, the auto-boot sequence starts the download of OS image over Ethernet.

Step 5. Type *exit* to come out of the menu command mode and start the Ethernet download of image.

The host responds to the BOOTME packets from the target and the download starts.

For more information about setting the Platform Builder for Ethernet download, see the Platform Builder On-Line Help documentation.

Note: The target uses DHCP for IP configuration to boot. In absence of a DHCP server, assign static IP address to the board using the command *env modify* on the serial console (before executing step 5) in the menu command mode.

2.8 Downloading and Flashing OS Image

Figure 13 describes the setup required for downloading and flashing the OS image into the target board. Follow steps 1 to 4 as described in Section 2.7 to prepare the setup. The following steps may be followed subsequently for flashing the OS image:

1. Abort the auto-boot sequence by pressing any key on the serial console.
2. Type the following command on bootloader prompt in the HyperTerminal:

   ```
   flash kernel
   ```

3. Type the following command on the HyperTerminal:

   ```
   exit
   ```

4. After downloading the image, the bootloader will program the image into Flash. After programming to flash is complete, the OS image will launch.

Note: Depending on the OS image size, the flash programming may take any time between 5 minutes to 45 minutes. Do not power-off the board while flash programming is in progress.

2.9 Booting From Flash Resident OS Image

Section 2.8 describes the procedure to flash the OS image into the target board. After having flashed the OS image, the boot parameters need to be changed to automatically boot from the OS image resident in the flash. The following steps describe the procedure:

Step 1. Ensure that the serial port of the target is connected to COM port of host and the HyperTerminal is started for corresponding COM port. The HyperTerminal settings should be:

a. Baud rate of 115200
b. 8-bit data
c. 1 stop bit
d. No parity
e. No flow-control

Step 2. Power ON the board and abort the auto boot sequence by pressing any key on the HyperTerminal.

Step 3. Type the following command to set the boot-parameter BOOT_MODE (for booting from Flash)

```
DAVINCI> env modify BOOT_MODE 0
```

Step 4. Power off the board and power on again. At this power on, the OS image comes up automatically (without any user intervention).

Step 5. After modification of the Boot Mode, switch OFF the board and power it ON again. This time the bootloader should load the Image directly from the NOR Flash.

2.10 Bootloader Command Line Interface

This section describes the various commands supported in this release of the bootloader and their usage. This set of commands operates on the boot parameters that are stored in flash (persistent storage). Section 2.10.2 provides the list of the boot parameters and their significance.

2.10.1 Bootloader Commands

Table 3 lists various commands supported in this release of bootloader. The arguments and its usage are illustrated in subsequent subsections.

Table 3. List of Commands in Bootloader

Number	Term	Description
1	?	Display the list of commands supported
2	exit	Leave the bootloader command line interface and start the download of the OS image.
3	quit	Leave the bootloader command line interface and start the download of the OS image.
4	env display	Display the list of boot parameters and their corresponding values.

Table 3. List of Commands in Bootloader (continued)

Number	Term	Description
5	env modify <arg1> <arg2>	Set the boot-parameter (as specified by <arg1>) with the value (as specified by <arg2>).
6	flash kernel	Command to program the OS image into the NOR-Flash upon the subsequent download.
7	flash info	Displays the information about the Flash memory type.
8	boot show	Displays the Windows CE specific BSP arguments.
9	mem	Command to write and display the contents of SDRAM.
10	oalzone set	Command to set the verbosity of bootloader execution.
11	oalzone get	Command to get the verbosity of bootloader execution.
12	net test	Utility command to send the sample packets over the Ethernet interface. This is made available for diagnostic purposes so as to be able to sniff the Ethernet packets from other systems.

2.10.1.1 Bootloader Command Usage

This section describes the usage of all the bootloader commands.

```
Davinci>?
Commands

exit | quit - Exits the Bootloader Menu Interface

env -    Env Library Related Commands
         display - Displays the env library
         modify - Modifies the environment variables
         NOTE:TFTP_CONFIG = 1- TFTP Server and 0 -TFTP Client
         NOTE:BOOT_MODE   = 0 -BootFlash, 1 -BootDownload and 2 -BootMenu

flash -  Flash Memory Related Commands
         info - Print Flash Memory Information
         Kernel - Flashes Windows CE Kernel

boot -   Boot Parameters Related Commands
         flashimage - Boots from flashed image
         show - Display Boot Parameters

mem -    Memory Related Commands[requires sub-strings]
         dump - Dump specified Memory Locations
         test - Perform Simple Memory Test
         copy - Copy One Memory Block to Another Block
         write - Writes to a particular location
         fill - Fills a Memory Block

oalzone- OAL Zones Commands
         get - Get the current OALZONES value
         set - Set the OALZONES value

net -    Network Parameters Related Commands
         test - Network Packet Test
Davinci>
```

Exit

```
Davinci>exit
Leaving BootLoader Menu
 Using Ethernet download.
IP Address: 0.0.0.0
Subnet Mask:255.255.255.0
Invoking EbootInitEtherTransport with Platform Name Davinci1455
InitDHCP():: Calling ProcessDHCP()
```

```
ProcessDHCP()::DHCP_INIT
Got Response from DHCP server, IP address: 192.168.13.11

ProcessDHCP()::DHCP IP Address Resolved as 192.168.13.11, netmask: 255.255.255.0
Lease time: 14400 seconds
Got Response from DHCP server, IP address: 192.168.13.11
No ARP response in 2 seconds, assuming ownership of 192.168.13.11
+EbootSendBootmeAndWaitForTftp
Sent BOOTME to 255.255.255.255
Sent BOOTME to 255.255.255.255
Sent BOOTME to 255.255.255.255
```

Quit

```
Davinci>quit
Leaving BootLoader Menu
 Using Ethernet download.
IP Address: 0.0.0.0
Subnet Mask:255.255.255.0
Invoking EbootInitEtherTransport with Platform Name Davinci1455
InitDHCP():: Calling ProcessDHCP()
ProcessDHCP()::DHCP_INIT
Got Response from DHCP server, IP address: 192.168.13.11

ProcessDHCP()::DHCP IP Address Resolved as 192.168.13.11, netmask: 255.255.255.0
Lease time: 14400 seconds
Got Response from DHCP server, IP address: 192.168.13.11
No ARP response in 2 seconds, assuming ownership of 192.168.13.11
+EbootSendBootmeAndWaitForTftp
Sent BOOTME to 255.255.255.255
Sent BOOTME to 255.255.255.255
```

Env display

```
Davinci>env display

Environment Variables  Respective Values
ENV_SIG                        ENVER1.11
DEVICE_ID                    Davinci1455
IP_GATEWAY                       0.0.0.0
IP_ADDR                          0.0.0.0
SUBNET_MASK                255.255.255.0
IP_SVR                           0.0.0.0
SVR_SUBNET_MASK            255.255.255.0
BOOT_ADDR                     0x00040000
TFTP_CONFIG                             1 ->TFTP_SVR
BOOT_MODE                               1 ->BOOT_DOWNLOAD
Davinci>
```

Env modify <arg1> <arg2>

```
Davinci>env display

Environment Variables  Respective Values
ENV_SIG                        ENVER1.11
DEVICE_ID                    Davinci1455
IP_GATEWAY                       0.0.0.0
IP_ADDR                          0.0.0.0
SUBNET_MASK                255.255.255.0
IP_SVR                           0.0.0.0
SVR_SUBNET_MASK            255.255.255.0
BOOT_ADDR                     0x00040000
TFTP_CONFIG                             1 ->TFTP_SVR
BOOT_MODE                               1 ->BOOT_DOWNLOAD
Davinci>env modify BOOT_MODE 0
Done ..This change applies to next boot
Davinci>env display

Environment Variables  Respective Values
```

```
ENV_SIG                          ENVER1.11
DEVICE_ID                      Davinci1455
IP_GATEWAY                         0.0.0.0
IP_ADDR                            0.0.0.0
SUBNET_MASK                  255.255.255.0
IP_SVR                             0.0.0.0
SVR_SUBNET_MASK              255.255.255.0
BOOT_ADDR                     0x00040000
TFTP_CONFIG                                1 ->TFTP_SVR
BOOT_MODE                                  0 ->BOOT_FLASH
Davinci>
```

Flash kernel

```
Davinci>flash kernel

Now Download WindowsCE Kernel you want to flash!
Davinci>
```

Flash info

```
Davinci>flash info

FlashType    = NOR
FlashStart   = 0x02000000
FlashLength  = 0x02000000
SectorSize   = 0x00010000
SectorCount  = 0x200
Davinci>
```

Boot show

```
Davinci>boot show

O E M   B O O T   P A R A M E T E R S
=======================================

OEM Platform Name               : Davinci1455
Boot Signature                  : 53475241
Boot Version                    : 1
OAL Version                     : 1
Reset Mode                      : Cold Boot
Boot Mode                       : 0
OEM Platform MAC Address        : 00:0e:99:02:55:14
Static IP address               : 0.0.0.0
Static Subnet Mask              : 255.255.255.0
IP  route                       : 0.0.0.0
Davinci>
```

mem

```
mem dump <start_addr> <length> <word_size>
mem write <start_addr> <data> <word_size>
mem copy <src_addr> <dst_addr> <byte_count>
mem fill <start_addr> <count> <access_width> <data>.
mem test <start_addr> <count>.
Davinci>mem test 0x80020000 0x10000

EBOOT: Starting memory test for area 0x80020000 to 0x80030000 at 70 secs.

EBOOT: Memory test complete at 70 secs (4 ms elapsed).
Davinci>
Davinci>mem dump 0x80020000 0x20 4

0x80020000: 80020000 80020004 80020008 8002000C  *................*
0x80020010: 80020010 80020014 80020018 8002001C  *................*
0x80020020: 80020020 80020024 80020028 8002002C  * ...$...(...,...*
0x80020030: 80020030 80020034 80020038 8002003C  *0...4...8...<...*
0x80020040: 80020040 80020044 80020048 8002004C  *@...D...H...L...*
0x80020050: 80020050 80020054 80020058 8002005C  *P...T...X...\...*
```

```
0x80020060: 80020060 80020064 80020068 8002006C  *`...d...h...l...*
0x80020070: 80020070 80020074 80020078 8002007C
Davinci>
Davinci>mem write 0x80020000 0x0f 4
 Address 0x80020000 to 0x80020004 Filled with Data F.
Davinci>
Davinci>mem dump 0x80020000 0x20 4

0x80020000: 0000000F 80020004 80020008 8002000C  *................*
0x80020010: 80020010 80020014 80020018 8002001C  *................*
0x80020020: 80020020 80020024 80020028 8002002C  * ...$...(...,...*
0x80020030: 80020030 80020034 80020038 8002003C  *0...4...8...<...*
0x80020040: 80020040 80020044 80020048 8002004C  *@...D...H...L...*
0x80020050: 80020050 80020054 80020058 8002005C  *P...T...X...\...*
0x80020060: 80020060 80020064 80020068 8002006C  *`...d...h...l...*
0x80020070: 80020070 80020074 80020078 8002007C
Davinci>mem copy 0x80020000 0x80040000 0x10
Davinci>
Davinci>mem fill 0x80020000 0x10 4 0x0
 Address 0x80020000 to 0x80020040 Filled with Data 0.
Davinci>
```

Oalzone set

```
Davinci>oalzone set 0xffffffff
INFO:OALLogSetZones: g_oalLogMask: 0xffffffff
-HandleZoneMenu
Davinci>+MenuGetLine
-MenuGetLine

Davinci>+MenuGetLine
```

Oalzone get

```
Davinci>oalzone get
Current Val of g_oalLogMask is 0x0000000B
```

Net test

```
Davinci>net test
Ethernet: Starting Sample UDP Packet Test...
Send 1 UDP Frame: Start Time 35610 MSecs End Time 35610 MSecs
Send 2 UDP Frame: Start Time 35616 MSecs End Time 35616 MSecs
Send 3 UDP Frame: Start Time 35622 MSecs End Time 35622 MSecs
Send 4 UDP Frame: Start Time 35628 MSecs End Time 35628 MSecs
Send 5 UDP Frame: Start Time 35634 MSecs End Time 35634 MSecs
Took 30 ms for Sending 5 UDP Frame(s).
Davinci>
```

2.10.2 Boot Parameters

Table 4 lists various boot parameters that are stored in the persistent storage (flash). The significance of each of the parameters is described in the following subsections.

Table 4. Boot Parameters List

Number	Parameter Name	Description	User Modifiable
1	ENV_SIG	This is the signature for the boot parameters stored in the persistent storage media (NOR Flash). This parameter is used by the bootloader code to determine the version of the boot parameters.	No
2	DEVICE_ID	The variable contains the name of the device as seen from the Platform Builder IDE tools. This is automatically generated at the first boot using the MAC address of the board.	No

Table 4. Boot Parameters List (continued)

Number	Parameter Name	Description	User Modifiable
3	IP_GATEWAY [1]*	This variable contains the IP Address (in dotted decimal format) of the IP Gateway. This variable is used while using the board as the TFTP Client, for downloading OS image over Ethernet using any general TFTP Server program (not Platform Builder IDE). This variable needs to be non-zero, if the TFTP server and the board belong to different IP Sub-net.	Yes
4	IP_ADDR	This variable contains the IP Address (in dotted decimal format) of the board. If this variable contains non-zero entry, then the board invokes the DHCP for getting the IP Address for the board. In the absence of the DHCP server, it is recommended to have the static IP address by giving non-zero value to this parameter.	Yes
5	SUBNET_MASK	This variable contains the IP Sub-net mask (in dotted decimal format) of the board.	Yes
6	IP_SVR [1]	This variable contains the IP Address (in dotted decimal format) of the TFTP Server. This variable is used while using the board as the TFTP Client, for downloading OS image over Ethernet using any general TFTP Server program (not Platform Builder IDE).	Yes
7	SVR_SUBNET_MASK [1]	This variable contains the IP Sub-net mask (in dotted decimal format) of the TFTP Server.	Yes
8	BOOT_ADDR	This variable contains the offset (with respect to the base of NOR-Flash and not absolute address) of the OS image in the NOR-Flash. This offset is used while flashing the OS image to the NOR-Flash and while booting from the flash resident images. While setting the value for this variable, sufficient care needs to be exercised so as not to erase any existing data in NOR-Flash. This value must be 64KBytes aligned and must be greater than 0x4000. Any value less than 0x4000 could result in corruption of the bootloader.	Yes
9	TFTP_CONFIG [1]	This variable contains the role assumed by the target during the Ethernet download. The target board may assume the TFTP Server role (for use with Platform Builder IDE) or may assume the TFTP Client role (for use with TFTP Server programs on host). The values it can have are: • 1 (TFTP Sever role for use with PB IDE) • 0 (TFTP Client role for use with general TFTP Server programs) **Note:** Changing this environment variable will not affect the target's role in downloading.	Yes
10	BOOT_MODE	This variable contains the information about the boot-mode. It can have following values: • 0 (Boot from Flash automatically) • 1 (Start Ethernet download automatically after timeout) • 2 (always halt at bootloader menu without automatically proceeding)	Yes

(1) These options are used while using the board as the TFTP Client with a general TFTP server program. This feature is no longer supported. Thus, modifying these values would have no effect.

2.10.2.1 Boot Parameters Usage

This section gives examples of how to change the boot parameters.

IP_GATEWAY

```
Davinci>env modify IP_GATEWAY 192.168.13.1
Davinci>env display

Environment Variables  Respective Values
ENV_SIG                       ENVER1.11
DEVICE_ID                   Davinci1455
```

```
IP_GATEWAY                  192.168.13.1
IP_ADDR                           0.0.0.0
SUBNET_MASK                 255.255.255.0
IP_SVR                            0.0.0.0
SVR_SUBNET_MASK             255.255.255.0
BOOT_ADDR                     0x00040000
TFTP_CONFIG                            1 ->TFTP_SVR
BOOT_MODE                              1 ->BOOT_DOWNLOAD
Davinci>
```

IP_ADDR

```
Davinci>env modify IP_ADDR 192.168.13.11
Davinci>env display

Environment Variables  Respective Values
ENV_SIG                         ENVER1.11
DEVICE_ID                     Davinci1455
IP_GATEWAY                   192.168.13.1
IP_ADDR                     192.168.13.11
SUBNET_MASK                 255.255.255.0
IP_SVR                            0.0.0.0
SVR_SUBNET_MASK             255.255.255.0
BOOT_ADDR                     0x00040000
TFTP_CONFIG                            1 ->TFTP_SVR
BOOT_MODE                              1 ->BOOT_DOWNLOAD
Davinci>
```

SUBNET_MASK

```
Davinci>env modify SUBNET_MASK 255.255.0.0
Davinci>env display

Environment Variables  Respective Values
ENV_SIG                         ENVER1.11
DEVICE_ID                     Davinci1455
IP_GATEWAY                   192.168.13.1
IP_ADDR                     192.168.13.11
SUBNET_MASK                   255.255.0.0
IP_SVR                            0.0.0.0
SVR_SUBNET_MASK             255.255.255.0
BOOT_ADDR                     0x00040000
TFTP_CONFIG                            1 ->TFTP_SVR
BOOT_MODE                              1 ->BOOT_DOWNLOAD
Davinci>
```

IP_SVR

```
Davinci>env modify IP_SVR 192.168.13.209
Davinci>env display

Environment Variables  Respective Values
ENV_SIG                         ENVER1.11
DEVICE_ID                     Davinci1455
IP_GATEWAY                   192.168.13.1
IP_ADDR                     192.168.13.11
SUBNET_MASK                   255.255.0.0
IP_SVR                      192.168.13.209
SVR_SUBNET_MASK             255.255.255.0
BOOT_ADDR                     0x00040000
TFTP_CONFIG                            1 ->TFTP_SVR
BOOT_MODE                              1 ->BOOT_DOWNLOAD
Davinci>
```

SVR_SUBNET_MASK

```
Davinci>env modify SVR_SUBNET_MASK 255.255.0.0
Davinci>env display

Environment Variables  Respective Values
```

```
ENV_SIG                          ENVER1.11
DEVICE_ID                      Davinci1455
IP_GATEWAY                    192.168.13.1
IP_ADDR                      192.168.13.11
SUBNET_MASK                    255.255.0.0
IP_SVR                     192.168.13.209
SVR_SUBNET_MASK                255.255.0.0
BOOT_ADDR                       0x00040000
TFTP_CONFIG                              1 ->TFTP_SVR
BOOT_MODE                                1 ->BOOT_DOWNLOAD
Davinci>
```

BOOT_ADDR

```
Davinci>env modify BOOT_ADDR 0x00080000
Davinci>env display

Environment Variables  Respective Values
ENV_SIG                          ENVER1.11
DEVICE_ID                      Davinci1455
IP_GATEWAY                    192.168.13.1
IP_ADDR                      192.168.13.11
SUBNET_MASK                    255.255.0.0
IP_SVR                     192.168.13.209
SVR_SUBNET_MASK                255.255.0.0
BOOT_ADDR                       0X00080000
TFTP_CONFIG                              1 ->TFTP_SVR
BOOT_MODE                                1 ->BOOT_DOWNLOAD
Davinci>
```

TFTP_CONFIG

```
Davinci>env modify TFTP_CONFIG 0
Davinci>env display

Environment Variables  Respective Values
ENV_SIG                          ENVER1.11
DEVICE_ID                      Davinci1455
IP_GATEWAY                    192.168.13.1
IP_ADDR                      192.168.13.11
SUBNET_MASK                    255.255.0.0
IP_SVR                     192.168.13.209
SVR_SUBNET_MASK                255.255.0.0
BOOT_ADDR                       0X00080000
TFTP_CONFIG                              0 ->TFTP_CLIENT
BOOT_MODE                                1 ->BOOT_DOWNLOAD
Davinci>
```

BOOT_MODE

```
Davinci>env modify BOOT_MODE 0
Done ..This change applies to next boot
Davinci>env display

Environment Variables  Respective Values
ENV_SIG                          ENVER1.11
DEVICE_ID                      Davinci1455
IP_GATEWAY                    192.168.13.1
IP_ADDR                      192.168.13.11
SUBNET_MASK                    255.255.0.0
IP_SVR                     192.168.13.209
SVR_SUBNET_MASK                255.255.0.0
BOOT_ADDR                       0X00080000
TFTP_CONFIG                              0 ->TFTP_CLIENT
BOOT_MODE                                0 ->BOOT_FLASH
Davinci>
```

3 Bootloader Options

3.1 Bootloader Build Options

This section describes the steps required to build the bootloader binary image.

3.1.1 Building EBOOT

Please see the Section 2 for information on building EBOOT with default settings.

3.2 Bootloader Flashing Options

This section describes the procedure to flash the bootloader.

3.2.1 Without UBL Using Code Composer Studio

Please see Section 2.3 for flashing the bootloader into the NOR-Flash.

3.2.2 Using Platform Builder

1. Build the bootloader, as described in Section 2.1.
2. Open the *Eboot.bin* image from the platform builder.
3. Download the eboot.bin, as described in Section 3.3.1.
4. After the download is complete, the bootloader updates itself.

3.3 Download Options

This section describes the procedure to download an image from the host to the Platform Builder, determined by the environment variable TFTP_CONFIG in the Environment library.

3.3.1 Ethernet Download using Platform Builder

To download an image from Platform Builder, the TFTP_CONFIG environment variable should be set to 1. This can be done using the *env modify* command (as described in Section 2.10.2). Section 2.7 illustrates the procedure to download the OS image using the platform builder.

4 References

- Microsoft Platform Builder 5.0 On-Line Help

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information 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.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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