

BIOS MCSDK 2.0

SRIO Boot Example

Applies to patch release based on 02.00.00.10
Publication Date: June 23, 2011
Version 1.2



Texas Instruments, Incorporated
20450 Century Boulevard
Germantown, MD 20874 USA

Document License

This work is licensed under the Creative Commons Attribution-NoDerivs 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nd/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

Contributors to this document

Copyright (C) 2011 Texas Instruments Incorporated - <http://www.ti.com>

© Copyright 2011 Texas Instruments, Inc.
All Rights Reserved

Contents

1	Overview	1
2	Revision History	1
3	References.....	1
4	DDR Init Boot Image	1
4.1	Procedure to build ddrinit.....	1
5	HelloWorld Boot Image.....	2
5.1	Procedure to build HelloWorld	2
6	Host Boot Example Application	3
6.1	Procedure to build Host Boot Example	3
7	Test Setup.....	3

SRIO Boot Example

1 Overview

The SRIO boot example is created to help customer quickly boot DSP through SRIO. The boot example includes three CCSv5 projects to build the DDR initialization image, HelloWorld boot image and the Host boot example application.

2 Revision History

Revision	Details
1.0	Initial Version
1.1	Added support for multi-core boot on DDR
1.2	Added support for C6670L EVM

3 References

[1] Bootloader for the C66x DSP User's Guide (Rev 1.0, <http://www.ti.com/litv/pdf/sprugy5>)

4 DDR Init Boot Image

The DDR Init project uses the BIOS MCSDK Platform Library to initialize the DDR.

4.1 Procedure to build ddrinit

- Import the project from tools\boot_loader\examples\srio\srioboot_ddrinit\evmc66xxl in CCSv5
- Clean and re-build the project
- The srioboot_ddrinit_evm66xxl.map and srioboot_ddrinit_evm66xxl.out will be generated under tools\boot_loader\examples\srio\srioboot_ddrinit\evmc66xxl\bin

- Run `srioboot_ddrinit_elf2HBin.bat` under `tools\boot_loader\examples\srio\srioboot_ddrinit\evmc66xxl\bin`, the batch file does the following file conversion:
 - Uses Code Gen utility `hex6x.exe` utility to convert the ELF format `.out` file to a ASCII hex format boot table file
 - Uses `Bttbl2Hfile.exe` to convert the boot table file to a header text file.
 - Uses `hfile2array.exe` to convert the header text file to a header file with array of the image data
 - Copies the converted header file to `tools\boot_loader\examples\srio\srioboot_example\src` so that the boot image can be linked into the Host boot example application

5 HelloWorld Boot Image

The HelloWorld project uses the BIOS MCSDK Platform Library to initialize the UART, it will print the “Hello World” and booting information for all the DSP cores to the UART once it runs.

5.1 Procedure to build HelloWorld

- Import the project from `tools\boot_loader\examples\srio\srioboot_helloworld\evmc66xxl` in CCSv5
- Clean and re-build the project
- The `srioboot_helloworld_evm66xxl.map` and `srioboot_helloworld_evm66xxl.out` will be generated under `tools\boot_loader\examples\srio\srioboot_helloworld\evmc66xxl\bin`
- Run `helloworld_elf2HBin.bat` under `tools\boot_loader\examples\srio\srioboot_helloworld\evmc66xxl\bin`, the batch file does the following file conversion:
 - Uses Code Gen utility `hex6x.exe` utility to convert the ELF format `.out` file to a ASCII hex format boot table file
 - Uses `Bttbl2Hfile.exe` to convert the boot table file to a header text file.
 - Uses `hfile2array.exe` to convert the header text file to a header file with array of the image data
 - Copies the converted header file to `tools\boot_loader\examples\srio\srioboot_example\src` so that the boot image can be linked into the Host boot example application

6 Host Boot Example Application

The Host Boot Example project uses the BIOS MCSDK Platform Library to initialize the DDR, it first pushes the DDR init boot image data to L2 memory of Core 0 on the remote booting EVM via SRIO, and then writes the entry address of the DDR init boot image to the SRIO boot magic address (refer to [1] to SRIO boot details) on Core 0 via SRIO. The RBL running on the DSP Core 0 polls the entry address and jumps to that address and starts to boot (initialize the DDR). The DDR init code will keep on polling the SRIO boot magic address after DDR is properly initialized.

The Host Boot example then pushes the Hello World boot image data to DDR memory of the remote booting EVM via SRIO, and then writes the entry address of the Hello World boot image to the SRIO boot magic address on Core 0 to boot core 0. Core 0 starts to boot and print the “Hello World” booting information, and then boot all the other cores by writing the entry address of write_boot_magic_number() function to the SRIO boot magic address on other cores and sending an IPC interrupt to other cores. The RBL running on other cores will jump to write_boot_magic_number() and start to boot, each core will write 0xBABEFACE to its SRIO boot magic address.

Note that Host Boot application needs to wait for some time after pushing the DDR init boot image and before pushing the Hello World boot image to the remote EVM, this will ensure DDR is properly initialized on the remote EVM.

6.1 Procedure to build Host Boot Example

- Import the project from tools\boot_loader\examples\srio\srioboot_example\evmc66xxl in CCSv5
- Clean and re-build the project
- The srioboot_example_evm66xxl.map and srioboot_example_evm66xxl.out will be generated under tools\boot_loader\examples\srio\srioboot_example\evmc66xxl\bin

7 Test Setup

Two breakout boards and two EVM’s are required to do the test:

- Connect the SRIO Lane 0-3 from one breakout board to the other, be sure lane n is connected to lane n on each breakout board.
- Insert the two EVM’s to the AMC slot on the two breakout boards
- Set the following boot dip switch settings on both EVM’s:
- C6678L EVM:

SW3(pin1, 2, 3, 4)	SW4(pin1, 2, 3, 4)	SW5(pin1, 2, 3, 4)	SW6(pin1, 2, 3, 4)
--------------------	--------------------	--------------------	--------------------

(off, off, on, on)	(on, on, on, off)	(on, off, on, off)	(off, on, on, on)
--------------------	-------------------	--------------------	-------------------

- C6670L EVM:

SW3(pin1, 2, 3, 4)	SW4(pin1, 2, 3, 4)	SW5(pin1, 2, 3, 4)	SW6(pin1, 2, 3, 4)
(off, off, on, on)	(on, on, on, off)	(on, off, on, off)	(off, off, on, on)

This will set the board to boot from SRIIO boot mode, with reference clock at 312.5 MHz, data rate at 3.125GBs, and lane setup 4-1x ports and DSP System PLL at 100 MHz.

Please refer to

http://processors.wiki.ti.com/index.php/TMDXEVM6678L_EVM_Hardware_Setup#Boot_Mode_Dip_Switch_Settings

and

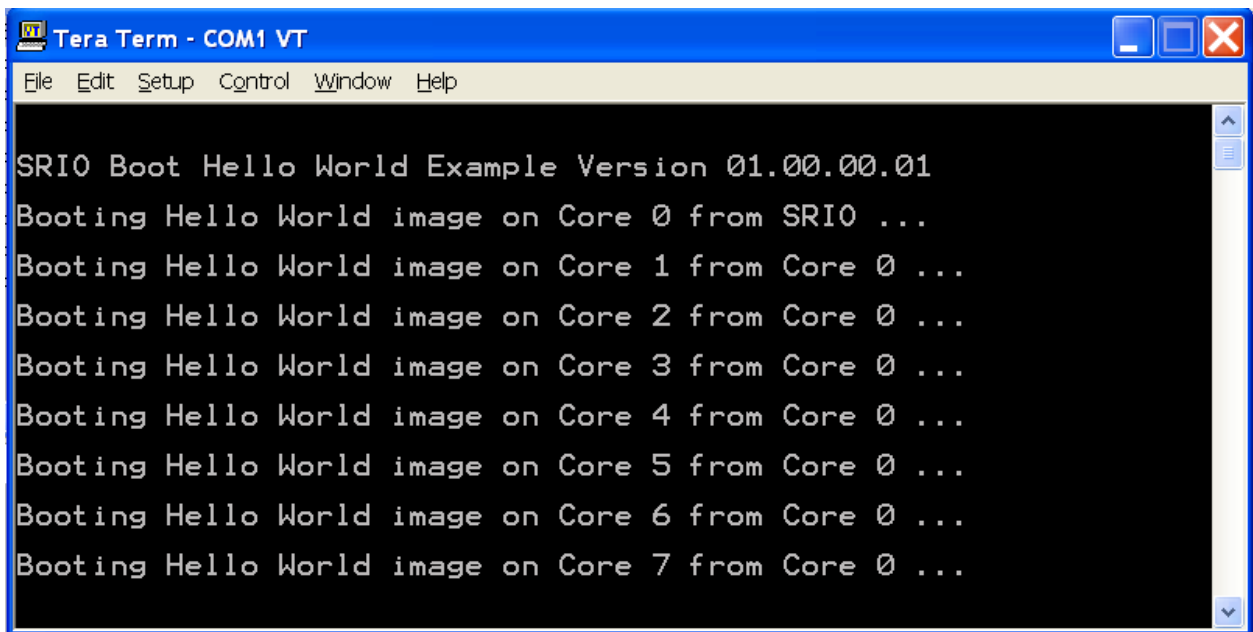
http://processors.wiki.ti.com/index.php/TMDXEVM6670L_EVM_Hardware_Setup#Boot_Mode_Dip_Switch_Settings

for the boot dip switch settings for C6678L and C6670L EVM's.

- Connect the booting EVM to the PC's serial port using a RS-232 cable, connect the JTAG emulator on the Host EVM.
- Power on both EVM's
- Open a Hyper Terminal or Tera Terminal connection, set the baud rate to 115200 bps, data 8-bit, parity none, stop 1-bit, and flow control none
- Connect the Host EVM using CCSv5, load and run srioboot_example_evm66xxl.out
- The CCS console will display the following message from the Host EVM:

```
[C66xx_0] SRIIO Boot Host Example Version 01.00.00.01
[C66xx_0]
[C66xx_0] Transfer DDR init code via SRIIO successfully
[C66xx_0] Transfer boot code via SRIIO successfully
```

- The Hyper Terminal will display the following message from the booting EVM:



The image shows a screenshot of a Tera Term window titled "Tera Term - COM1 VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The main display area is black with white text. The text shows the boot process for "SRI0 Boot Hello World Example Version 01.00.00.01". It starts with "SRI0 Boot Hello World Example Version 01.00.00.01", followed by "Booting Hello World image on Core 0 from SRI0 ...". Then, it lists booting the image on Core 1 through Core 7, all from Core 0. The text is as follows:

```
SRI0 Boot Hello World Example Version 01.00.00.01
Booting Hello World image on Core 0 from SRI0 ...
Booting Hello World image on Core 1 from Core 0 ...
Booting Hello World image on Core 2 from Core 0 ...
Booting Hello World image on Core 3 from Core 0 ...
Booting Hello World image on Core 4 from Core 0 ...
Booting Hello World image on Core 5 from Core 0 ...
Booting Hello World image on Core 6 from Core 0 ...
Booting Hello World image on Core 7 from Core 0 ...
```