# SC-MCSDK 2.0 Getting Started Guide

**Small Cell Multicore Software Development Kit**
**Version 2.x**
**Getting Started Guide**

*Last updated: //*

## Introduction


The Small Cell Multicore Software Development Kit (SC-MCSDK) provides the core foundational building blocks that facilitate application software development on TI's high performance and multicore SOC. The SOC has an ARM CPU and four DSPs. The foundational components include:

- SYS/BIOS which is a light-weight real-time embedded operating system for TI devices
- Linux Software package that has Linux kernel and supporting software to enable application development on Linux.
- Chip support libraries, drivers, and basic platform utilities
- Interprocessor communication for communication across cores and devices
- Basic networking stack and protocols
- Optimized application-specific and application non-specific algorithm libraries
- Debug and instrumentation
- Bootloaders and boot utilities
- Demonstrations and examples

Specifically, this *Getting Started Guide* provides information on installing the Small Cell Multicore Software Development Kit, loading the EVM out-of-box demonstration application via JTAG, and running the out-of-box demonstration application.

By the end of this *Getting Started Guide* the user should have:

- Installed CCS
- Installed Simulator
- Installed the SC-MCSDK Software
- Connected to the EVM via CCS/JTAG or to the Simulator
- Loaded the out-of-box demonstration application onto the device via JTAG
- Executed the out-of-box demonstration application


**Useful Tip**

After completing the material in this *Getting Started Guide*, it is recommended the user continue on to the SC-MCSDK User's Guide for additional information on the SC-MCSDK software elements and to get started with development using the SC-MCSDK.

## Acronyms and Definitions

The following acronyms are used throughout this document.

| Acronym | Meaning |
|---|---|
| AMC | Advanced Mezzanine Card |
| CCS | Texas Instruments Code Composer Studio |
| CSL | Texas Instruments Chip Support Library |
| DDR | Double Data Rate |
| DHCP | Dynamic Host Configuration Protocol |
| DSP | Digital Signal Processor |
| DVT | Texas Instruments Data Analysis and Visualization Technology |
| EDMA | Enhanced Direct Memory Access |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| EVM | Evaluation Module, hardware platform containing the Texas Instruments DSP |
| HUA | High Performance Digital Signal Processor Utility Application |
| HTTP | HyperText Transfer Protocol |
| IP | Internet Protocol |
| IPC | Texas Instruments Inter-Processor Communication Development Kit |
| JTAG | Joint Test Action Group |
| MCSA | Texas Instruments Multi-Core System Analyzer |
| SC-MCSDK | Texas Instruments Small Cell Multi-Core Software Development Kit |
| NDK | Texas Instruments Network Development Kit (IP Stack) |
| NIMU | Texas Instruments Network Interface Management Unit |
| PDK | Texas Instruments Platform Development Kit |
| RAM | Random Access Memory |
| RTSC | Eclipse Real-Time Software Components |
| SRIO | Serial Rapid IO |
| TCP | Transmission Control Protocol |
| TI | Texas Instruments |
| UART | Universal Asynchronous Receiver/Transmitter |
| UDP | User Datagram Protocol |

| | |
|---|---|
| UIA | Texas Instruments Unified Instrumentation Architecture |
| USB | Universal Serial Bus |

**Note:** We use the abbreviation TMS when referring to a specific TI device (processor) and the abbreviation TMD when referring to a specific platform that the processor is on. For example, TMS320TCI6614 refers to the TCI6614 SOC and TMDXEVM6614 refers to the actual hardware EVM that the processor is on.

### Supported Devices/Platforms

This release supports the following Texas Instrument devices/platforms:

- **SOC - TMS320TCI6614**
- **EVM - TMDXEVM6614**
- **Simulator - TCI6614**

# Getting Started

This section will walk you through installing and using the Small Cell Multicore Software Development Kit, including how to flash and run the out-of-box demonstration application. It is expected that you have gone through the EVM *Quick Start Guide*[TBD] for your EVM and have booted the demonstration application[TBD] flashed on the device.

The overall steps are:

1. Make sure your EVM hardware is set up
2. Install Code Composer Studio 5.4
3. Install the SC-MCSDK 2.2
4. Install the Simulator
5. Use JTAG to load the application
6. Run the application

---

### Hardware Setup

Follow instructions in the link below to setup EVM

- TMDXEVM6614 EVM Hardware Setup [1]

---

### Installing Code Composer Studio

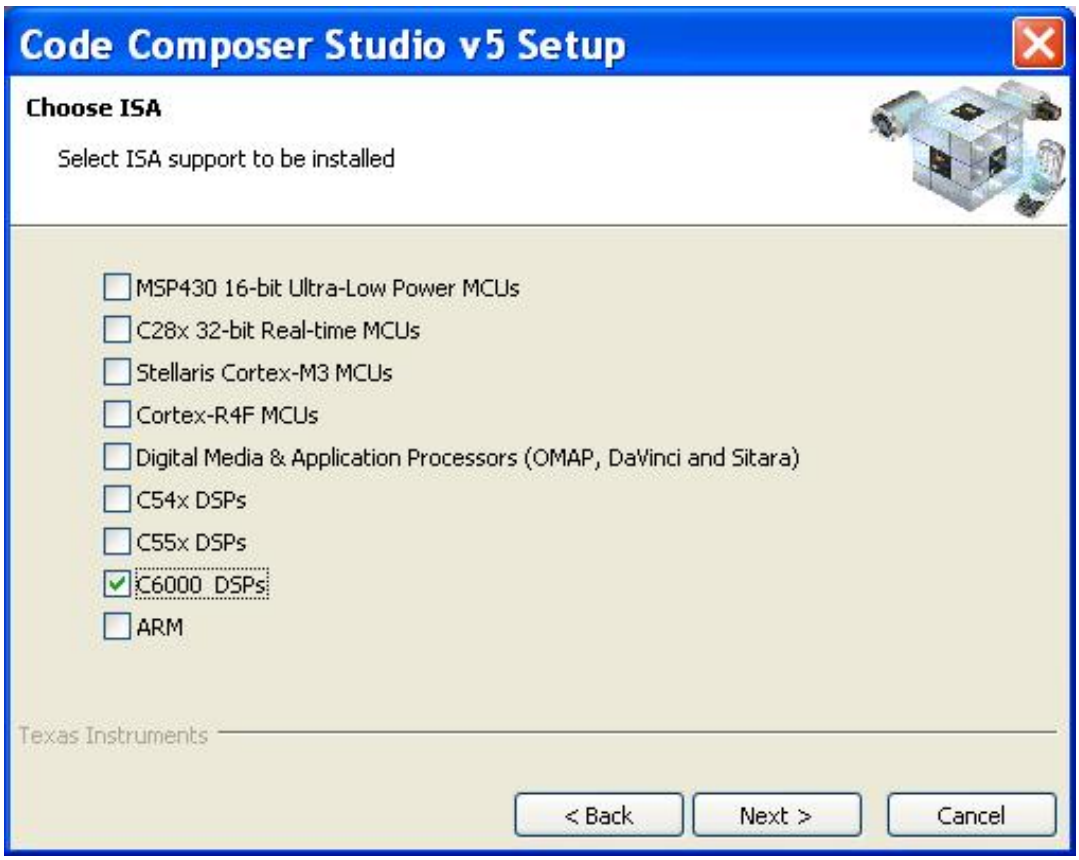
The SC-MCSDK 2.2 uses CCS 5.4. To install CCS please refer to the instructions provided in the CCSv5 Getting Started Guide [2].

**Note:** When installing Code Composer Studio you can choose to control what is installed. For example, you may not want support for particular devices and ISA's. If you choose to do a custom installation, the following components must be installed to support the SC-MCSDK:

- SYS/BIOS 6
- IPC
- XDC
- All C6* DSP

**Note:** In Windows 7, install and run CCS/SC-MCSDK in administrator mode, this will rule out issues with CCS start-up and RTSC component recognition
Sample Custom Installation snap shot:


Select C6000 DSPs during Choose ISA option



## Installing the Small Cell Multicore Software Development Kit

After installing CCS, the next step is to install the SC-MCSDK. The release contents are delivered as installers

On Windows

Run the sc-mcsdk-<build-id>-DSP_setupwin32.exe executable and choose the directory that CCSv5 is installed in for installation of the SC-MCSDK

On Linux

Run the sc-mcsdk-<build-id>-DSP_setuplinux.bin executable and choose the directory that CCSv5 is installed in for installation of the SC-MCSDK

Some of the components installed by CCS may be updated by the SC-MCSDK installer; see the Release Notes for full details of components included in the installer. The development environment can be either Windows or Linux.

A sample directory structure after a typical SC-MCSDK default installation is shown below

```
+---sc_mcsdk_bios_<version>
+---ccsv5
+---edma3_lld_<version>
+---ipc_<version>
+---sc_mcsdk_linux_<version>
+---ndk_<version>
+---pdk_tci6614_<version>
+---syslib_<version>
\---xdctools_<version>
```

**Note:** If you are installing on a Windows 7 machine, modify the default path to "C:\ti". Otherwise, some of the CCS projects in the release may not build properly.

**Note:** The SC-MCSDK installer might default to *C:\Program Files(x86)\Texas Instruments* for 64 bit machines. There are some projects in the SC-MCSDK which assumes the default SC-MCSDK installation directory to be *C:\Program Files\Texas Instruments*. The workaround would be to install SC-MCSDK in *C:\Program Files\Texas Instruments* folder, then all the projects will re-compile without any change. If it is installed in *C:\Program Files(x86)\Texas Instruments* then minor tweaks might be required for few projects to re-compile.
The Linux kernel and u-boot pre-built binaries provided in the release under the folder sc_mcsdk_linux_<version>.

The following is a more detailed listing of the contents are under this folder:

```
release_folder
   ti
     sc_mcsdk_linux_<version>
        bin
        board-support
           linux-src
           uboot-src
        boot
        docs
        example-applications
        host-tools
           loadlin
           u-boot
        images
```

```
        linux-devkit
        post
```

- bin

  contains pre-built binaries for mkfs.ubifs and ubinize

- host-tools/loadlin

  contains dss scripts and collaterals for loading and running Linux kernel on the Simulator

  - loadlin.xsl
  - tci6614-sim.ccxml - simualtor configuration file for ccs
  - tci6614-evm.ccxml- EVM configuration file for ccs
  - arago-min-root-sim.cpio - canned min-root file system for Simulator - used by initrd during kernel boot up.
  - arago-min-root-evm.cpio - canned min-root file system for EVM - used by initrd during kernel boot up
  - loadlin-sim.js - dss script that loads and runs Linux kernel on simulator.
  - loadlin-evm.js - dss script that loads and runs Linux kernel on the EVM.
  - tracelog.xml - trace file

- images

  - zImage-tci6614-evm.bin

    precompiled Linux kernel image suitable for loading on to the EVM through CCS.

  - tci6614-evm.dtb

    device Tree blob for tci6614 EVM

  - uImage-tci6614-evm.bin

    A precompiled Linux kernel image suitable for loading through u-boot.

  - u-boot-tci6614-evm.bin

    A precompiled U-Boot image in blob binary format suitable for loading on EVM through CCS

  - arago-sc-mcsdk-base-image-tci6614-evm.tar.gz- Tar file. Use this if using nfs rootfs
  - arago-sc-mcsdk-base-image-glibc-ipk-2011.06-tci6614-evm.ubifs.img - ubifs image for rootfs volume
  - arago-sc-mcsdk-recovery-image-glibc-ipk-2011.06-tci6614-evm.ubifs.img - ubifs image for rootfs-recovery

    volume

  - tci6614-boot.ubifs.img - ubifs image for boot volume that has the uImage and tci6614-evm.dtb files
  - tci6614-evm-ubifs.ubi - ubi image for tci6614 EVM. This has three volumes:- boot volume, rootfs volume

    (primary rootfs) and rootfs-recovery volume (recovery rootfs)

  - arago-base-image-tci6614-sim.tar.gz - Tar file. Use this to create arago-min-root-sim.cpio if you want to use the latest file system supplied in the release.(steps are included in the User Guide)

- boot

  - mkfs.ubifs - mkfs host utility to create the ubifs image
  - ubinize - ubinize host utility to creat the ubi image

- host-tools

  - evmc6614lxe_<version>.gel - Gel file used for booting Linux through CCS.

Other folders doesn't have any contents as of now, but may be filled in future releases.

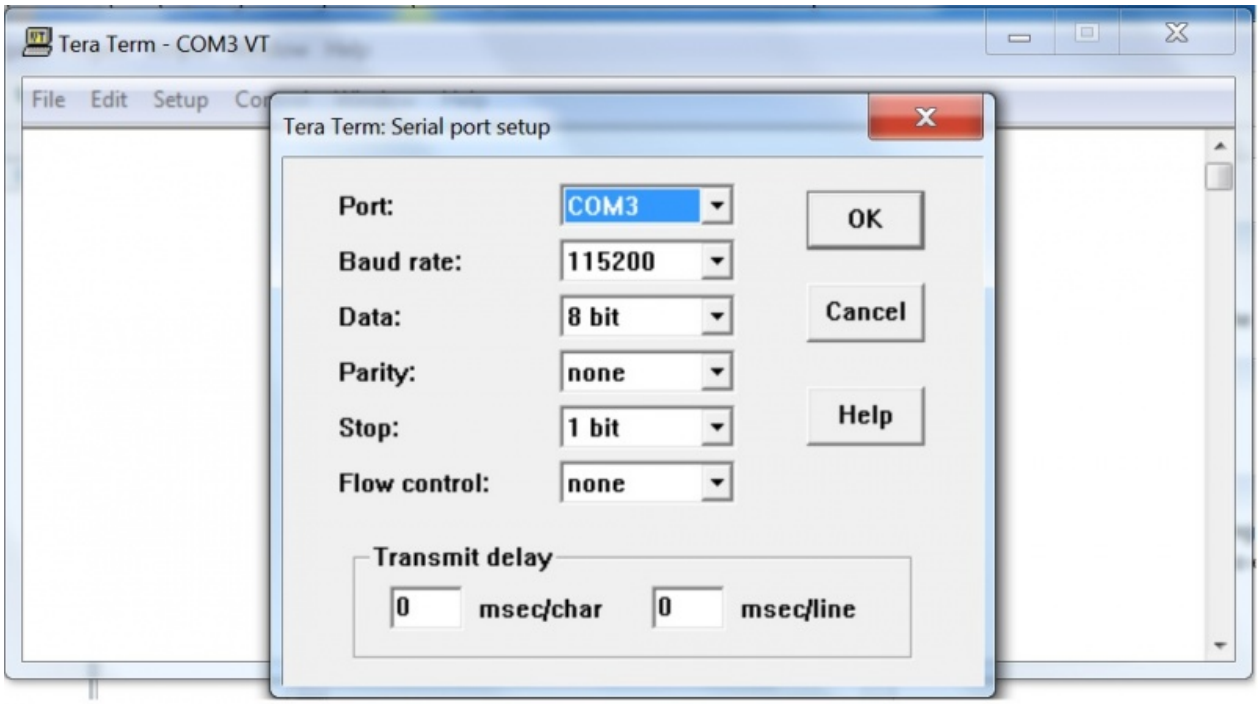
# Loading and Running Application on the target

The TCI6614 includes a Cortex A8 ARM and several DSPs. It is possible to use CCS to load and run an application on the DSPs but also on the ARM.

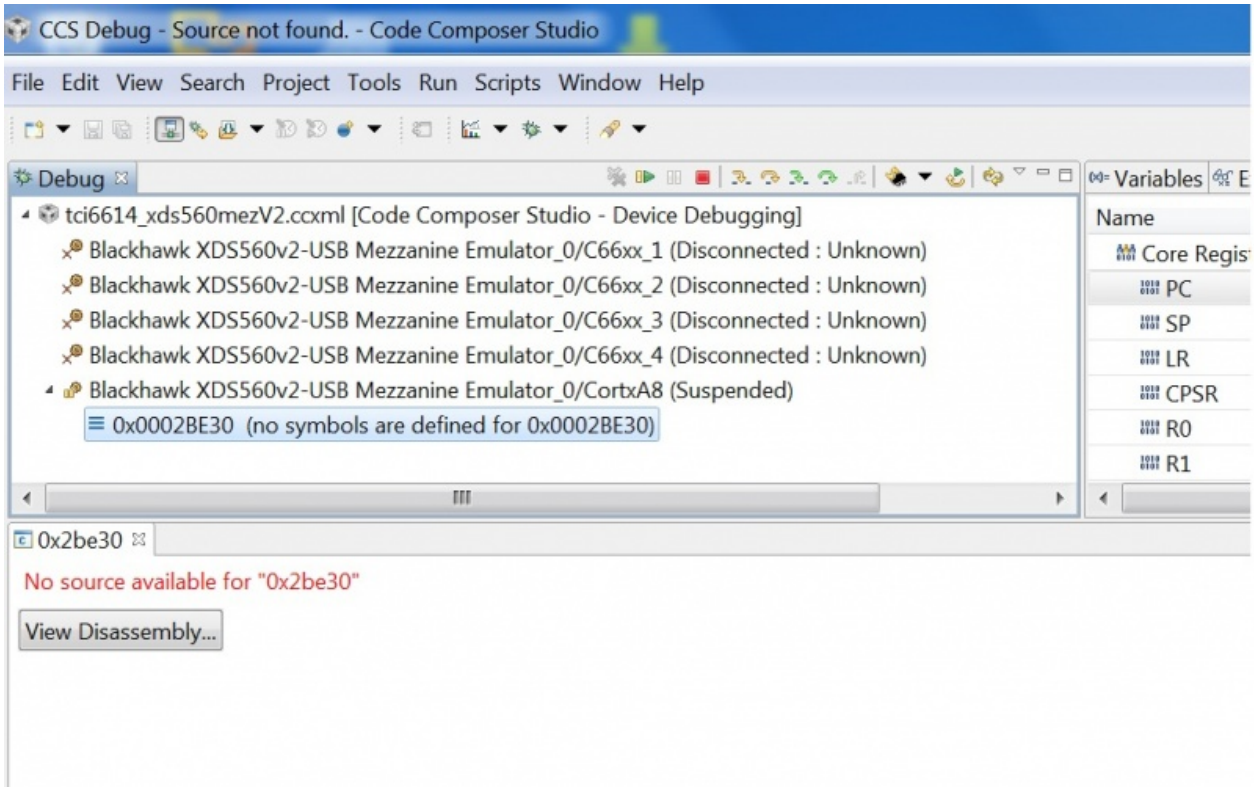
## Running applications on ARM

### Loading and running U-Boot on EVM using CCS

- Open Tera-Term or Hyper Terminal, create a new connection with baud rate 115200bps, 8 data bit, no parity, 1 stop bit and no flow control, be sure the serial cable is connected between the PC and the EVM with the right COM port.

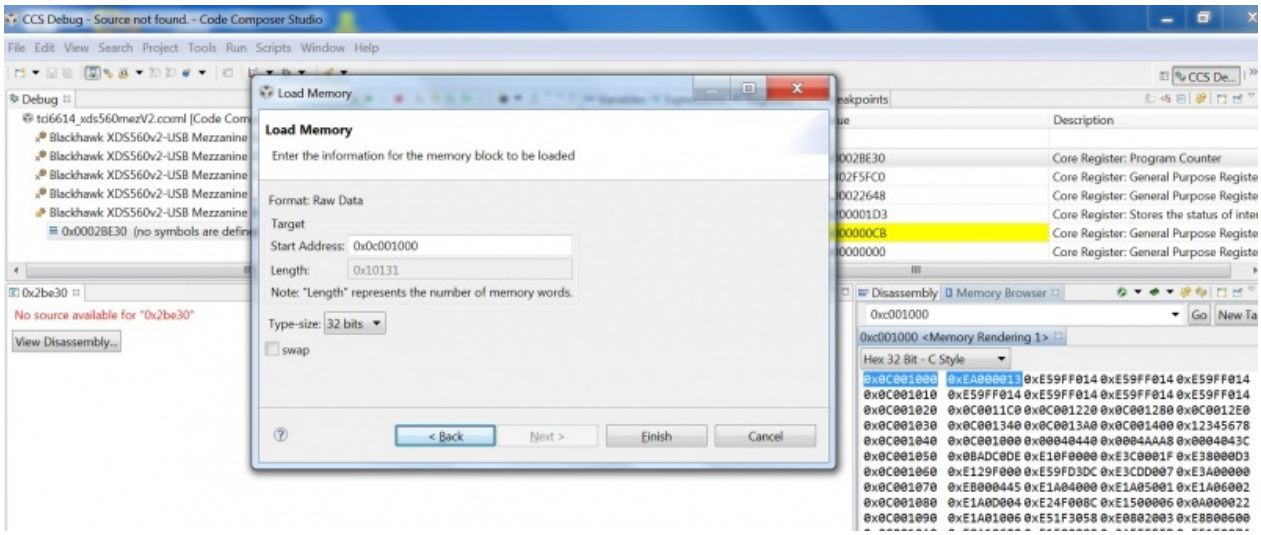



- Be sure the boot dip switch is set to ARM master no boot mode on the TCI6614 EVM. See the HW Setup Guide for details.
- In CCS, launch the tci6614_xds560v2 target
- In CCS Debug perspective view, connect the CortexA8 ARM core in the Debug Window:

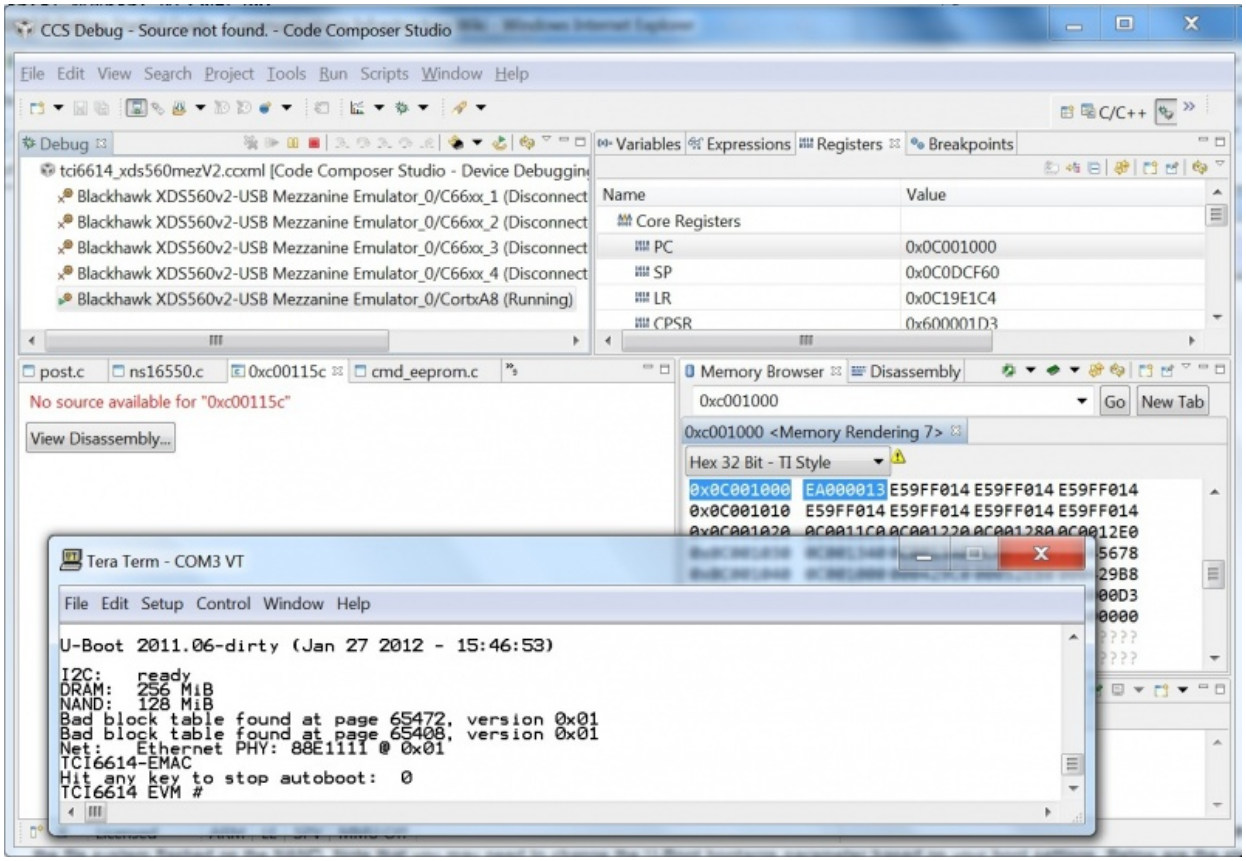



- In Memory Browser window, load u-boot-tci6614-evm.bin

to start address 0x0c001000 (MSMC memory) with 32-bits type-size:




- In Registers window, change the PC register value to 0x0c001000 and run the image and U-Boot will be booting up and print out the booting messages to the UART console on Tera/Hyper Terminal




# Setting Up DSP to Run Aplications

## Use JTAG to Connect to the EVM

Before starting this section, make sure that the dip switches on the EVM are configured for DSP (or ARM) master, no boot, LE mode.

The following are the settings for the DSP master, no boot LE mode

- SW3 (p4,p3,p2,p1) = on,on,on,off
- SW4 (p4,p3,p2,p1) = on,on,on,on
- SW5 (p4,p3,p2,p1) = on,on,on,on
- SW6 (p4,p3,p2,p1) = on,on,on,on
- SW2 (p4,p3,p2,p1) = on,on,on,off

The following steps will guide you through connecting the JTAG to the EVM. These instructions are for the XDS560v2-USB Mezzanine Emulator.

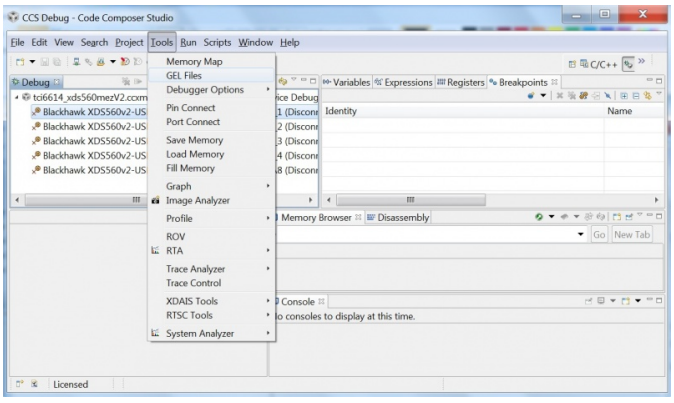
1. First time CCSv5 users needs to follow the steps described in CCSv5_Running_for_the_first_time [3] page.
2. The above procedure will open up the CCSv5 GUI.
3. Create a New Target Configuration, by selecting the tab *File->New* and clicking *Target Configuration File*.
4. Enter a configuration file name (say "evmtci6614_xds560v2.ccxml") in the file name field and hit finish button (the procedure assumes you are using XDS560v2-USB Mezzanine emulation).
5. Select the connection type as "Blackhawk XDS560v2-USB Mezzanine Emulator" in the drop down list and enter device number in the *Device* search filed. Select the EVM device as specified in the Target Config figure and save the file by clicking *Save* button.
6. Click View->Target configurations to list the available target configurations. The configuration file (evmtci6614_xds560v2.ccxml) created above will be under "User Defined" section

7. Right click on the configuration file and select *Launch Selected Configuration*. This will launch the configuration and open the debug view.
8. Right click on core0 and select *Connect Target* to connect to the target CPU. This step will require the board is powered up and the PC is connected to the board using USB.
9. Most of the development and debugging will require CCSv5. For more information on CCSv5 see the CCSv5 Getting Started Guide [2] page.

**Loading and Setting up the EVM with the Gel file**

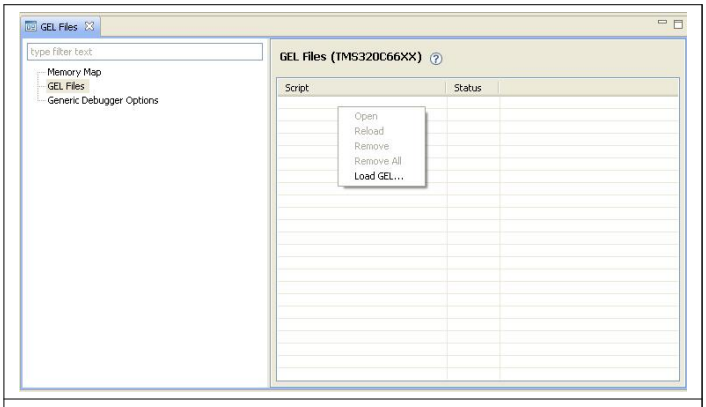
After the target configuration is launched, the GEL file can be set for each core. Before executing any application it is recommended to execute the GEL script *Global_Default_Setup* for the evmtci6614. The example below shows how to load and execute the GEL script from the GEL file for TMDXEVM6614 EVM.

1. Click on the core on which the GEL file needs to be loaded and select Tools->GEL Files from the Tools Menu. This is captured in the GEL Menu under CCSv5
2. Right-click on the first row of the empty list to get a Load GEL command
3. Execute the Load GEL command.
4. Browse & Open the GEL file. The evmtci6614.gel GEL file is located under pdk_tci6614_1_xx_xx_xx\packages\ti\platform\evmtci6614\gel directory.
5. Run Global_Default_Setup from the Scripts Menu.



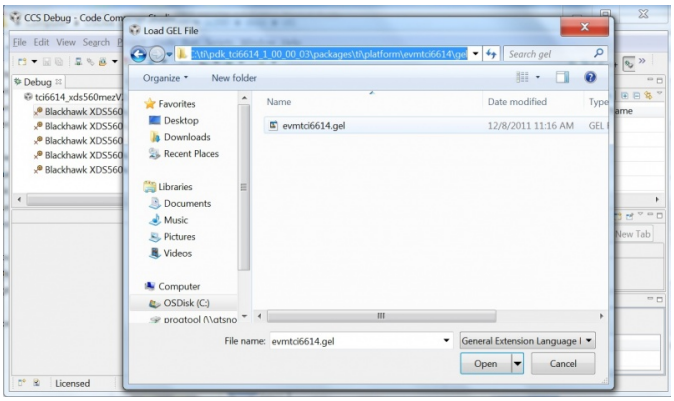

GEL Menu under CCSv5

.



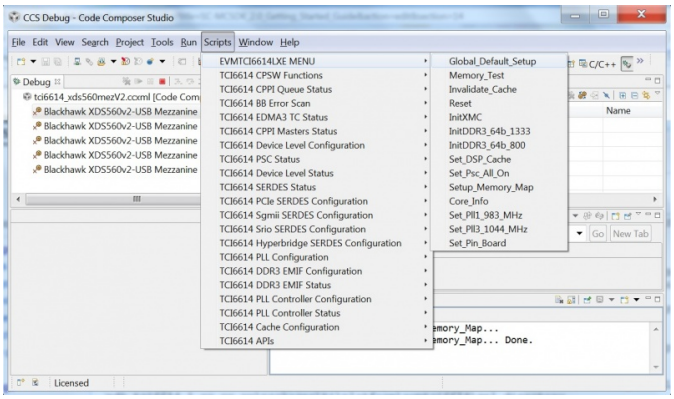

Load GEL Command

.




Browsing the GEL File

.




Scripts->Global_Default_Setup

.

**Note**: Ubuntu CCS5.1.1 does not detect XDC tools neither from CCS directory nor from release directory. This is a problem of linux CCS511 only, windows versions works fine. This happens when CCS is installed on Linux in “sudo” mode, but then launched as regular user. Following are work arounds:-

1. Always install and launch CCS as same user – either install and launch as “sudo”, or install and launch as regular user. When installing as regular user, an additional script needs to be run to install emulation drivers - <installdir>/ccsv5/install_scripts/install_drivers.sh (run it as sudo).
2. Or – install as “sudo”, then recursively change the owner of the ccsv5/eclipse/ directory to the regular user. This then allows that user to launch CCS without running into this issue.

**Note**: If you set the EVM as boot from NAND, then the EVM will boot the U-boot and the gel file need not to be run.

---

# References

[1] http://wiki.ci.dal.design.ti.com/index.php/TMDXEVM6614LXE_EVM_Hardware_Setup
[2] http://processors.wiki.ti.com/index.php/CCSv5_Getting_Started_Guide
[3] http://processors.wiki.ti.com/index.php/GSG:CCSv5_Running_for_the_first_time

# Article Sources and Contributors

**SC-MCSDK 2.0 Getting Started Guide** *Source*: http://wiki.ci.dal.design.ti.com/index.php?oldid=5470 *Contributors*: A0221004, A0868405, AravindBatni, Elvitalobo, Hao, Icesar, M-karicheri2, RajSivarajan, SajeshSaran, Tmannan

# Image Sources, Licenses and Contributors

**Image:C66x-multicore-dsp-arm.jpg** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:C66x-multicore-dsp-arm.jpg *License*: unknown *Contributors*: RajSivarajan

**File:Helpful_tips_image.jpg** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:Helpful_tips_image.jpg *License*: unknown *Contributors*: RajSivarajan

**File:Light_bulb_icon.png** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:Light_bulb_icon.png *License*: unknown *Contributors*: RajSivarajan

**Image:CCS-setup-01.JPG** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:CCS-setup-01.JPG *License*: unknown *Contributors*: Hao

**Image:CCS-setup-02.JPG** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:CCS-setup-02.JPG *License*: unknown *Contributors*: Hao

**Image:Tci6614 uart setup.jpg** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:Tci6614_uart_setup.jpg *License*: unknown *Contributors*: Hao

**Image:Tci6614 connect arm.jpg** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:Tci6614_connect_arm.jpg *License*: unknown *Contributors*: Hao

**Image:Tci6614 load uboot.jpg** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:Tci6614_load_uboot.jpg *License*: unknown *Contributors*: Hao

**Image:Tci6614 run uboot.jpg** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:Tci6614_run_uboot.jpg *License*: unknown *Contributors*: Hao

**Image:tci6614_CCSV5GelMenuSelect.JPG** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:Tci6614_CCSV5GelMenuSelect.JPG *License*: unknown *Contributors*: Hao

**Image:emptygelwindow.jpg** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:Emptygelwindow.jpg *License*: unknown *Contributors*: Hao

**Image:loadgeltci6614.jpg** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:Loadgeltci6614.jpg *License*: unknown *Contributors*: Hao

**Image:loadgeltci6614_global_setup.jpg** *Source*: http://wiki.ci.dal.design.ti.com/index.php?title=File:Loadgeltci6614_global_setup.jpg *License*: unknown *Contributors*: Hao