

Sitara Linux Software Developer's Guide

v5.07.00.00

NOTE - This PDF is hyperlinked. Clicking on a link (typically bolded and underlined) will take you to that chapter or open the appropriate website.



Contents

Articles

Sitara Linux Software Developer's Guide	1
How to Build a Ubuntu Linux host under VMware	15
Sitara SDK Installer	32
Sitara Linux SDK Setup Script	35
AMSDK START HERE Script	38
Matrix Users Guide	41
Power Management Users Guide	50
ARM Multimedia Users Guide	55
Camera Users Guide	79
Cryptography Users Guide	86
Oprofile User's Guide	94
Open Source Wireless Connectivity WLAN Bluetooth User Guide	98
OMAP Wireless Connectivity OpenSource Compat Wireless Build	99
OMAP Wireless Connectivity mac80211 compat wireless implementation Architecture	100
OMAP Wireless Connectivity Battleship Game demo	103
AMSDK u-boot User's Guide	108
AMSDK Linux User's Guide	118
Code Composer Studio v5 Users Guide	122
Linux Debug in CCSv5	153
How to setup Remote System Explorer plug-in	165
How to Run GDB on CCSv5	181
Pin Mux Utility for ARM MPU Processors	189
Pin Setup Tool for AM18xx ARM Microprocessors	211
AM335x Flashing Tools Guide	219
Flash v1.6 User Guide	222
AM18x Flash Tool User's Guide	236
AMSDK File System Optimization/Customization	238
Sitara Linux Training	240
How to use a Mouse instead of the Touchscreen with Matrix	242
How to Recalibrate the Touchscreen	243
AM335x U-Boot User's Guide	244
Sitara Linux SDK Top-Level Makefile	259
Sitara Linux SDK GCC Toolchain	262
Sitara Linux SDK create SD card script	266

How to add a JVM	271
AM335x U-Boot User's Guide	273

References

Article Sources and Contributors	288
Image Sources, Licenses and Contributors	289

Article Licenses

License	293
---------	-----

Sitara Linux Software Developer's Guide



For the SDG specific to your SDK release, please refer to Archived SDGs under Reference Documentation

Welcome to the Sitara Linux Software Developer's Guide

Thank you for choosing to evaluate one of our Sitara ARM microprocessors ^[1]. Please *bookmark* this page and refer back to it as needed. It is designed to quickly provide the information you need most while evaluating the AMx microprocessor. We are always striving to improve this product. Please let us know if you have **ideas or suggestions**.

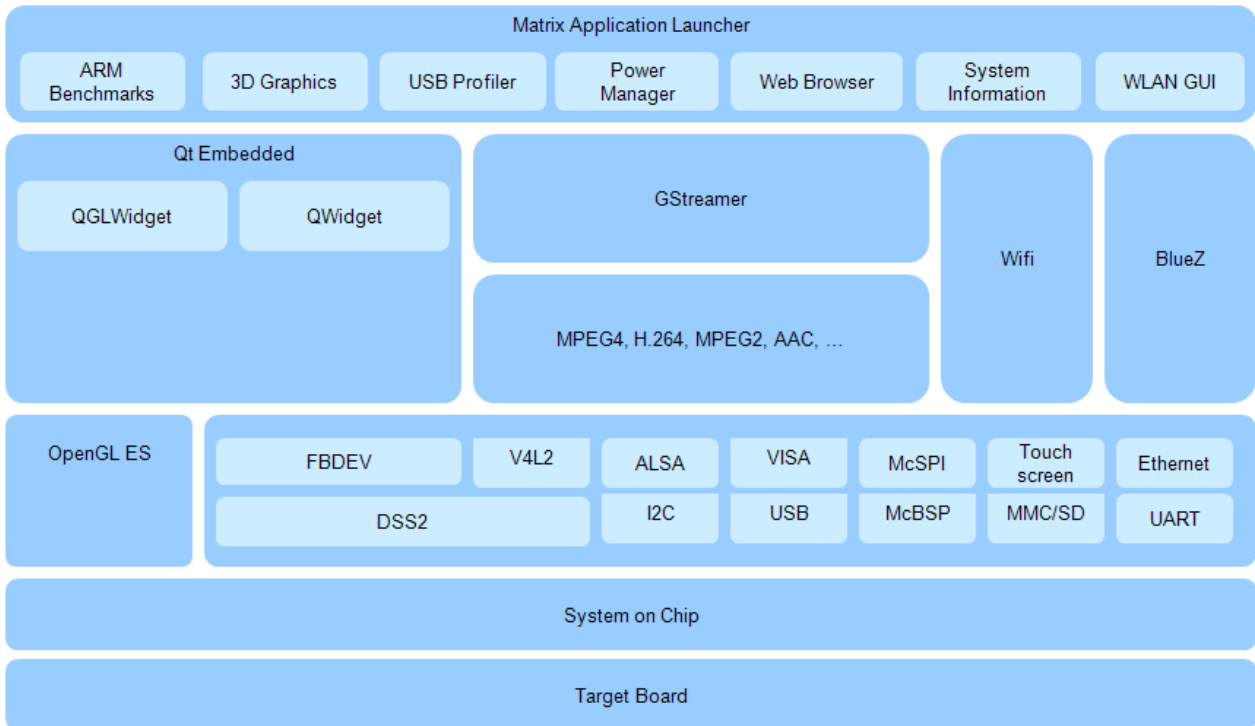
Supported Platforms & Version Information

The following Sitara ARM microprocessors are supported with this SDK version.

Platforms	SDK	PSP	Kernel	U-Boot	Toolchain	Release Date
BeagleBone	5.07	04.06.00.10	3.2	2013.01	gcc4.5.3	April 2013
AM335xEVM	5.07	04.06.00.19	3.2	2013.01	gcc4.5.3	April 2013
AM335x StarterKit (SK)	5.07	04.06.00.10	3.2	2013.01	gcc4.5.3	April 2013
Beagleboard-xM	5.07	-	3.3.7	2012.04.01	gcc4.5.3	April 2013
AM37xEVM	5.07	04.02.00.07	2.6.37	2012.04.01	gcc4.5.3	April 2013
AM35xEVM	5.07	04.02.00.07	2.6.37	2011.09	gcc4.5.3	April 2013
AM180xEVM	5.07	03.21.00.04	2.6.37	2010.12	gcc4.5.3	April 2013

Linux Software Stack

The following software stack illustrates at a high level the various components provided with the Sitara Linux SDK. **NOTE - Availability of certain applications are platform dependent and clarified in the associated User Guides below.**



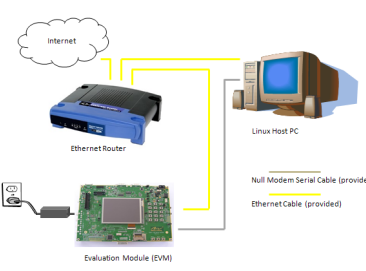
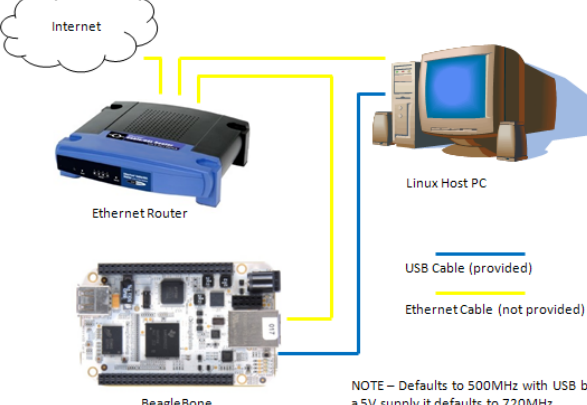
EVM Hardware Overview

Details for various hardware platforms supported by this Sitara Linux SDK are provided before.

Platform	Document	EVM Provider
AM335xEVM	Hardware User's Guide ^[2]	www.ti.com ^[3]
AM335x StarterKit (SK)	Hardware User's Guide ^[4]	www.ti.com ^[3]
BeagleBone	Hardware User's Guide (Rev A3) ^[5]	Beagleboard.org ^[6]
Beagleboard-xM	Hardware User's Guide ^[7]	Beagleboard.org ^[6]
AM37xEVM	Hardware User's Guide ^[8]	Mistral ^[9]
AM35xEVM	Hardware User's Guide ^[10]	Logic ^[11]
AM180xEVM	Hardware User's Guide ^[12]	Logic ^[11]

Start your Linux Development

Before using the Sitara Linux SDK you must have a Linux development environment. Linux development environments vary, so we recommend the following for ease of use and consistency when working together. The next few steps will assist you in the setup & configuration of your Linux host based on the physical setup shown below.

<p>1. Configure a Linux Host - If you already have a Linux host machine, go to Step 2. If you do not have a Linux host machine, you can configure a Linux host machine on your Windows PC using a virtual machine.</p> <ul style="list-style-type: none"> • Build a Ubuntu 10.04 LTS Linux host with VMware on WinXP (preferred) • Build a Ubuntu 10.04 LTS Linux host with VirtualBox on WinXP 	
<p>For standard EVMs: 2. Configure your development environment</p>  <p>Internet</p> <p>Ethernet Router</p> <p>Linux Host PC</p> <p>Null Modem Serial Cable (provided)</p> <p>Ethernet Cable (provided)</p> <p>Evaluation Module (EVM)</p>	<p>For Beaglebone: 2. Configure your development environment</p>  <p>Internet</p> <p>Ethernet Router</p> <p>Linux Host PC</p> <p>USB Cable (provided)</p> <p>Ethernet Cable (not provided)</p> <p>BeagleBone</p> <p>NOTE – Defaults to 500MHz with USB bus power. With a 5V supply it defaults to 720MHz.</p>
<p>3. Install the SDK - Within your Linux host machine, Install the Sitara SDK</p>	<p>3. Run the START_HERE Script - When Beaglebone boots, a START_HERE directory will display on your Linux host desktop. To start your evaluation and development using the BeagleBone, Run the START_HERE.sh script</p> <p>NOTE: The START_HERE.sh script will run the setup.sh script. For Beaglebone boards before rev A5 the setup.sh script will attempt to install a udev rule to load the ftdi_sio module with the proper parameters to recognize the Beaglebone. If you experience difficulty with the Beaglebone being recognized you can load the module manually by doing:</p> <ul style="list-style-type: none"> • <code>sudo rmmod ftdi_sio</code> • <code>sudo modprobe ftdi_sio vendor=0x0403 product=0xa6d0</code>
<p>4. Run the Setup Script - Once the SDK has been installed, Run the Setup.sh Script to guide you through the remaining development environment configuration.</p>	<p>NOTE: If using a VMWare image you will likely need to import the beaglebone into the VMWare image as a mass storage device using the instructions here</p>

Configuring a Serial Connection

If you chose not to use the setup.sh script (or START_HERE.sh) to configure you connection you can always perform the following steps to configure a serial connection with the board.

- The following steps will help you validate that you have a valid serial connection with your target board

IMPORTANT

These steps are written for boards like the EVM-SK and Beaglebone which use an FTDI USB-to-Serial adapter. For boards with a straight UART connection you can ignore the steps for /dev/ttyUSBx and instead just use the serial port on your Linux host that is connected to the board

- Connect the USB cable between your Linux host and your target board

IMPORTANT

On the AM335x EVM-SK, the USB cable enables a USB-to-serial interface to the Linux host PC. If using a virtual machine, please insure the AM335x EVM-SK is a selected USB device.

NOTE

If you are running your VMWare image with Windows 7 some USB-to-Serial adapters do not properly work with Windows 7. Make sure that your adapter specifically says that it supports Windows 7. You may also need to install an additional driver for your adapter.

- Open a terminal and run the following command to find the USB serial adapters available on the system. The FTDI chip used on the board presents two serial interfaces to the Linux host. The first interface is the JTAG interface and the second one is the serial console interface.

```
ls /dev/ttyUSB*
```

You should see output like:

```
/dev/ttyUSB0
```

```
/dev/ttyUSB1
```

NOTE

If using VMWare you may need to pass the "Future Technology Devices" device into the VMWare image

- As mentioned above since the board's serial interface is the second interface you will want to open a serial console to the second device node. In this case that is **/dev/ttyUSB1**. This can be done by doing:
 - **minicom -w -s**
 - **Select Serial port setup and press ENTER**
 - **Press A to modify the Serial Device and change the device to /dev/ttyUSB1. Press ENTER**
 - **Press ENTER to exit the serial setup**
 - **Select Exit and press ENTER**
- You should now see a minicom window.
- Power on the board, you should see "cccc" being printed in the window, if so the serial connection is working as expected. Go ahead and leave this console RUNNING in the background.

NOTE

For the EVM-SK board the power button is under the board in the upper-right hand corner. You must press and hold this button to power on the board.

NOTE

For the Beaglebone the board will power on as soon as the USB cable is connected. you can reset the board by pressing the reset button which should provide you with the expected output

SDK Directory Structure Overview

The Sitara Linux SDK contains the following top-level directories and files

```
— bin
— board-support
— docs
— example-applications
— filesystem
— Graphics_SDK_setuptools_4_05_00_03.bin
— host-tools
— linux-devkit
— Makefile
— Rules.make
— setup.sh
```

These directories contain the code and tools used to develop for Sitara devices.

- **bin** - Contains the helper scripts for configuring the host system and target device. Most of these scripts are used by the `setup.sh` script.
- **board-support** - Contains the SDK components that need to be modified when porting to a custom platform. This includes the kernel and boot loaders as well as any out of tree drivers.
- **docs** - Contains various SDK documentation such as the software manifest and additional user's guide. This is also the location where you can find the *training* directory with the device training materials.
- **example-applications** - Contains the sources for the TI provided example applications seen during the out-of-box demonstration.
- **filesystem** - Contains the reference file systems. These include the smaller base file system as well as the full-featured SDK file system.
- **host-tools** - Contains the host side tools such as pinmux and flash tool.
- **linux-devkit** - Contains the cross-compile toolchain and libraries to speed development for the target device.
- **Graphics_SDK_setuptools_<version>.bin** - This is the installer for the graphics SDK. The graphics SDK components are used by the Sitara Linux SDK to provide additional demos as well as integrated with the pre-built Qt libraries to accelerate various Qt functions.
- **Makefile** - Provides build targets for many of the SDK components from the top-level of the SDK.
- **Rules.make** - Sets default values used by the top-level Makefile as well as sub-component Makefiles
- **setup.sh** - Configures the users host system as well as the target system for development

Example Applications User's Guides

There are a number of Example Applications provided within the Sitara Linux SDK. Below are the applications available on each platform and the User's Guides associated with each component.

NOTE: The example applications below assume that you are using the default pinmux/profile configuration that the board ships with, unless otherwise noted in the individual application's User's Guide

Applications available by development platform

Applications	AM335x	BeagleBone	AM335x SK	AM180x	AM35x	AM37x	Beagleboard-xM	Users Guide	Description
Matrix GUI	X	X	X	X	X	X	X	<u>Matrix User's Guide</u>	Provides an overview and details of the graphical user interface (GUI) implementation of the application launcher provided in the Sitara Linux SDK
Power & Clocks	X	X	X			X	X	<u>Power Management User's Guide</u>	Provides details of power applications integrated into the Matrix GUI, along with helpful command line equivalents.
Multimedia	X		X		X	X	X	<u>Multimedia User's Guide</u>	Provides details on implementing ARM/Neon based multimedia using GStreamer pipelines and FFmpeg open source codecs.
Camera						X		<u>Camera User's Guide</u>	Provides details on how to support smart sensor camera sensor using the Media Controller Framework
USB Profiler	X	X	X	X	X	X	X	NA	
ARM Benchmarks	X	X	X	X	X	X	X	NA	
Oprofile	X	X	X	X	X	X	X	<u>Oprofile User's Guide</u>	Provides applications which show how Oprofile can be used for debugging and understanding system performance.
Display	X		X		X	X		NA	
Cryptography	X	X	X	X	X	X	X	<u>Cryptography User's Guide</u>	Provide details on how to implement cryptography through use of OpenSSL and various example applications.

WLAN and Bluetooth	X		X	X		X		Open Source Wireless Connectivity WLAN/Bluetooth User's Guide	Provides details on how to enable the WL1271 daughtercard which is connected to the EVM
QT Demos	X	X	X	X	X	X	X	NA	
Web Browser	X	X	X	X	X	X	X	NA	
System Settings	X	X	X	X	X	X	X	NA	

Bootloaders - How to Build & Install

The following provides instructions on how to build and install uboot and MLO. Additional reference documentation is also provided.

- [Building & Installing Uboot & MLO](#)

Kernel - How to Configure, Build & Install

The following provides instructions on how to configure, build and install the Linux kernel. Additional reference documentation is also provided.

- [Build, Configure and Install the Linux Kernel](#)

Host Side Development Tools

There are number of host side development tools provided in the Sitara Linux SDK to ease your development. See the appropriate User's Guide below for more information.

Host Tools for Target Development

These are tools that will assist in cross-compiling code for the target device or making images to run on the target device

Top-Level Makefile

The Sitara Linux SDK contains a top-level Makefile in the root of the SDK tree that can be used to rebuild many of the example applications and board support packages. See the [Sitara Linux SDK Top-Level Makefile](#) page for more information on using this Makefile.

GCC Cross-Compile Toolchain

Starting with version 05.xx of the Sitara Linux SDK there is now a GCC cross-compile toolchain available within the SDK in the linux-devkit directory. For details on using this toolchain please see the [Sitara Linux SDK GCC Toolchain](#) page.

create-sdcard.sh script

Starting with version 05.04 of the Sitara Linux SDK there is now a shell script available called **create-sdcard.sh** which can be used to create an SD card image for booting the various Sitara EVMs. For details on using this script please see the [Sitara Linux SDK create SD card script](#) page.

Code Composer Studio v5 User's Guide

Code Composer Studio v5 - a full featured, Eclipse based IDE that includes the Remote System Explorer plugin. CCSv5 can be used for both application development & debug using gdbserver and kernel debug using an XDSv100 JTAG emulator.

- [Code Composer Studio v5 \(CCSv5\)](#)

GEL files for Sitara Processors can be found at the following link

- [Sitara CCS Support](#)

For detailed instructions based on the Beagbone on how to do Linux debugging with CCSv5 see:

- [Linux Debug in CCSv5](#)

Host tools for Target Configuration

These tools are intended to be run on the host and assist in configuring the target device

Pin Mux Utility User's Guides

Pin Mux Utility - a graphical utility designed to ease the pin mux configuration effort by highlighting pinmux conflicts, configuring the IO and outputting the required source code.

- [AM35x/AM37x/AM335x Pin Mux Utility User's Guide](#)
- [AM180x Pin Setup Tool User's Guide](#)

Flash Tool User's Guides

- [AM335x Flash Tool User's Guide](#) USB RNDIS and Ethernet flash programming from Linux host.
- [AM335x U-Boot User's Guide](#) Detailed description of all AM335x U-Boot/SPL functionality.
- [AM35x/AM37x Flash Tool User's Guide](#) Flash v1.6.0.0 GUI Based Flash Tool for AM35x/AM37x for USB and UART flash programming
- [AM180x Flash Tool User's Guide](#) Serial Flash Tool for AM180x

Linux Training

In Sitara we recognize the need for solid, real world embedded Linux training. As we continue to enhance the features of our Sitara Linux SDK ^[13] we have also focused on developing training material based on the same SDK.

That way, anything provided in the training can be easily reproduced on your side once you have installed the Sitara Linux SDK.

We have a number of modules that are complete and others we are actively working on. All of our Linux training is provided on this wiki.

If you have comment on the training or a request for Linux training that is not be captured here, please let us know using the sdk_feedback@list.ti.com ^[14] mailing list

NOTE

If you are unable to access Google Drive documents you can also find this training material at [www.ti.com/sitarabootcamp www.ti.com/sitarabootcamp]

Lecture	Lab	Description
Linux Host Configuration	Sitara Linux Training: Linux Host Configuration	This page details how the Linux Host is configured for Sitara Linux Training. These are the same methods to prepare laptops used in live TI training.
Introduction to Linux	-	Introduces the community-based Linux ecosystem on TI platforms. What will be covered are the components that make up the ecosystem such as the boot loader, Linux kernel, device drivers, user application layer and the relationship between them.
Linux Boot Process ^[15]	-	Looks at all aspects of the boot process from power up to running user a application beginning with ROM boot loader progressing through secondary program loader, u-boot, kernel and finishing with user-level initialization.
Sitara Linux Training: Hands-on with the SDK ^[16]	Sitara Linux Training: Hands on with the SDK	Learn about the various components that make up the ARM MPU Linux software development kit including the out-of-box application launcher, the CCS IDE, example applications. In addition, host tools such as the pin-mux utility and the flash tool will be introduced. All these components are packaged into a single easy to use installer from TI.com
Code Composer Studio v5	Code Composer Studio v5	Covers what the Eclipse-based Code Composer Studio is, how to use it for embedded Linux application development, debugging and additional plug-ins that are provided
Sitara Linux Training: Power Management ^[17]	Sitara Linux Training: Power Management	Discusses how to improve product power performance by minimizing power consumption and guaranteeing system performance. In addition, power management techniques enabled via the Linux SDK will be discussed
Sitara Linux Training: Cryptography ^[18]	Sitara Linux Training: Cryptography	Covers cryptography basics and explore cryptographic functions enabled via open source projects. In addition, cryptographic hardware acceleration access and Linux SDK example applications will be discussed.
Sitara Linux Training: Linux Board Port ^[19]	Sitara Linux Training: Linux Board Port	Discusses the fundamentals necessary to port a TI Linux-based EVM platform to a custom target platform. Introduces the necessary steps needed to port the following components: Linux kernel.
Sitara Linux Training: U-Boot Board Port ^[19]	Sitara Linux Training: U-Boot Board Port	Discusses the fundamentals necessary to port a TI Linux-based EVM platform to a custom target platform. Introduces the necessary steps needed to port the following components: secondary program loader, u-boot.
Sitara Linux Training: U-Boot/Kernel Debug with CCSv5 ^[20]	Sitara Linux Training: U-Boot Linux Debug with CCSv5	Learn about how U-Boot and Kernel Debug can be done using CCSv5 using JTAG. This presentation and accompanying lab will discuss what debug information is necessary to be built into U-Boot and the Kernel to allow source code level debug with a JTAG interface.
ARM Multimedia	ARM Multimedia	Introduces open-source based multimedia codecs for the ARM Cortex-A8. In addition, look at the capability of the NEON coprocessor to accelerate multimedia. Plus, introduces GStreamer, an open-source pipeline-based framework that enables access for multimedia through FFMPEG/libav support on the ARM. GStreamer will be illustrated with Linux SDK examples.

Sitara Linux Training: Hands on with QT ^[21]	Sitara Linux Training: Hands on with QT	Learn how to develop a GUI quickly with the Linux SDK. Learn background information on QT. Learn how to use the SDK to get started developing a GUI. Learn about QT Creator and all the QT toolset.
Oprofile	Oprofile	Introduces the Opensource tool Oprofile. When is it useful during the development cycle. Introduce some of the more popular features. Cover both modes of operation, internal HW counters or timer interrupts. Cover internal operation details. Also point out use cases where Oprofile may not be useful.
Init Scripts ^[22]	Sitara Linux Training: Init Scripts	Learn how the Linux init scripts work with the sysvinit system as well as how the profile scripts can be used to affect the login process.
Optimizing Linux Boot Time ^[23]	Sitara Linux Training: Optimizing Linux Boot Time	Learn how to identify the portions of the Linux boot taking the most time and remove or defer those operations until later. The goal of this lab is to have a system booting to a display on the LCD and reading a touchscreen event in less than 3 seconds.

Other How Tos

This section provides how-to articles on additional setups and configurations you may find useful in your development.

Host How Tos

- [Creating a HelloWorld CCS Project](#)
- [How to Develop with 3D Graphics](#)
- [How to Connect to an EVM via Telnet](#)
- [How to Setup a Samba Server](#)
- [Understanding the Boot Sequence](#)
- [How to Move Files From Host to Target](#)
- [AM37x NAND flashing from U-boot](#)
- [Preventing BeagleBone board reset on JTAG Connect](#) ^[24]

Target How Tos

- [How to utilize Error Correction \(ECC\)](#)
- [How to Recalibrate the Touchscreen](#)
- [AMSDK File System Optimization/Customization](#)
- [How to add Oracle Hotspot JVM to your target Linux Filesystem](#)
- [How to use a Mouse instead of the Touchscreen with Matrix](#)
- [How to enable DVI display](#)
- [Update U-Boot Environment Variables stored in SPI Flash from Linux](#)

Reference Documentation

Release Notes

- [Sitara SDK Release Notes](#)
- [WLAN/BT Release Notes](#)
- [Graphics SDK Release Notes](#)
- [Linux PSP Release Notes](#)

Graphics Documentation

- [Graphics SDK Getting Started Guide](#)
 - [Graphics SDK Quick installation and user guide](#)
- [Graphics SDK Release Notes](#)
- [SGX Debugging Tips and FAQ](#)
- [Qt Tips](#)

Linux PSP Documentation

PSP Release Notes

AM335x PSP

- [AM335x PSP 04.06.00.10 Release Notes](#)
- [AM335x PSP 04.06.00.09 Release Notes](#)
- [AM335x PSP 04.06.00.08 Release Notes](#)
- [AM335x PSP 04.06.00.07 Release Notes](#)

AM35x/AM37x PSP

- [AM35x-AM37-PSP 04.02.00.07 Release Notes](#)

AM180X PSP

- [AM180x-PSP 03.21.00.04 Release Notes](#)

PSP Features and Performance Guides

AM335x PSP

- [AM335x PSP 04.06.00.10 Features and Performance Guide](#)
- [AM335x PSP 04.06.00.08 Features and Performance Guide](#)
- [AM335x PSP 04.06.00.08 Features and Performance Guide](#)
- [AM335x PSP 04.06.00.07 Features and Performance Guide](#)

AM35x/AM37x PSP

- [AM35x-AM37-PSP 04.02.00.07 Feature Performance Guide](#)

AM180X PSP

- [AM180x-PSP 03.21.00.04 Feature Performance Guide](#)
-

PSP User's Guides

AM335x PSP

- [AM335x PSP User's Guide](#)

AM35x/AM37x PSP

- [AM35x-AM37-PSP 04.02.00.07 User's Guide](#)

AM180X PSP

- [AM180x-PSP 03.21.00.04 User's Guide](#)

Miscellaneous Documentation

QT Documentation

- [QT Reference Documentation](#) ^[25]

Archived - Software Developer's Guide

- [Sitara SDK 5.06.00.00 - Software Developer's Guide \(archived\)](#) ^[26]
- [Sitara SDK 5.05.00.00 - Software Developer's Guide \(archived\)](#) ^[27]
- [Sitara SDK 5.04.01.00 - Software Developer's Guide \(archived\)](#) ^[28]
- [Sitara SDK 5.04.00.00 - Software Developer's Guide \(archived\)](#) ^[29]
- [Sitara SDK 5.03.03.00 - Software Developer's Guide \(archived\)](#) ^[30]
- [Sitara SDK 5.03.02.00 - Software Developer's Guide \(archived\)](#) ^[30]
- [Sitara SDK 5.03.01.00 - Software Developer's Guide \(archived\)](#) ^[31]
- [Sitara SDK 5.03.00.00 - Software Developer's Guide \(archived\)](#) ^[32]
- [Sitara SDK 5.02.00.00 - Software Developer's Guide \(archived\)](#) ^[33]
- [Sitara SDK 4.01 - Software Developer's Guide \(archived\)](#) ^[34]
- [Sitara SDK 4.00 - Software Developer's Guide \(archived\)](#) ^[35]

GPLv3 Disclaimer

There are GPLv3 licensed software components contained within the Sitara Linux SDK on both the target and the host. The software manifest (software_manifest.htm) for the Sitara Linux SDK is located in the docs/ directory of the installed SDK. All GPLv3 components for both target and host are contained in the SDK directory.

These GPLv3 components are provided for development purposes only and may be removed for production solutions.

How to Identify the GPLv3 components

To identify the GPLv3 components installed on the target file system, **run the gplv3-notice script** located on the target file system located here: */etc/init.d/gplv3-notice*

The gplv3-notice script will list all Sitara Linux SDK built shipped installed packages. If you installed additional GPLv3 components this script may not identify them until the next target reboot.

How to Remove Target side GPLv3 Components

The gplv3-notice script also outputs how to remove the packages. To remove individual packages from the target development file system, use the `opkg remove <package>`

Software Updates

We are continually improving the quality and content of the software we provide in the EVM. Updates to the SDK may be obtained at [Software Updates](#)^[36] as they become available.

Technical Support

- [E2E Support Forums](#)^[37] - an active community of TIers and other customer like you already using the AM37x EVM. You may find your question has already been answer with a quick Search of the Forums. If not, a quick post will likely provide you the answers you need. Support@ti.com - a support email list you may submit your question to.
- support@ti.com^[38]

Want to Contribute?

We are always striving to improve this Sitara Linux SDK. Please let us know if you have ideas or suggestions. The sections below will give you ideas on how to best contribute to the SDK and PSP Linux kernel.

SDK Contributions

All Sitara Linux SDK contributions can be sent to the sdk_feedback@list.ti.com^[14] mailing list. This covers submitting bug fixes, new features, or any other feedback to components provided with the Sitara Linux SDK including, but not limited to:

- u-boot
- target file system
- Host Tools
- Scripts
- Example Applications
- Documentation

Some general guidelines to help us with your feedback are:

- **Documentation**
 - Provide the URL of the document being discussed
 - Describe the section in question
 - If you have suggestions on the change requested please include them
- **Source Code/Scripts/Makefiles**
 - Provide the location of the source code being discussed within the SDK
 - If possible provide a patch as this is easier to determine the changes being proposed and helps us to make sure you get credit for your work
- **Tools**
 - Provide the version of the tool
 - If possible give screen captures or step-by-step instructions for reproducing any issues

Following the above guidelines will help to streamline the communication, but we would like to get your feedback no matter what.

PSP Linux Kernel Contributions

For information on how to contribute to the PSP Linux kernel please see the [Contributing to the Linux PSP](#) page.

References

- [1] <http://www.ti.com/lstds/ti/dsp/arm.page>
- [2] http://processors.wiki.ti.com/index.php/AM335x_General_Purpose_EVM_HW_User_Guide
- [3] <http://www.ti.com>
- [4] <http://processors.wiki.ti.com/index.php/AM335xStarterKitHardwareUsersGuide>
- [5] http://beagleboard.org/static/beaglebone/a3/Docs/Hardware/BONE_SRM.pdf
- [6] <http://www.beagleboard.org>
- [7] http://beagleboard.org/static/BBxMSRM_latest.pdf
- [8] http://www.mistralsolutions.com/assets/downloads/AM37x_EVM.php
- [9] <http://www.mistralsolutions.com/>
- [10] <http://www.logicpd.com/products/development-kits/zoom-am3517-evm-development-kit#tabs-som-4>
- [11] <http://www.logicpd.com/>
- [12] <http://www.logicpd.com/products/development-kits/zoom-am1808-evm-development-kit#tabs-som-4>
- [13] <http://www.ti.com/tool/linuxezsdk-sitara>
- [14] mailto:sdk_feedback@list.ti.com
- [15] <https://docs.google.com/open?id=0BzESOSf028mLaGFuaVFFSzVkVVU>
- [16] <https://docs.google.com/open?id=0BzESOSf028mLbFBucThORVRwZnM>
- [17] <https://docs.google.com/open?id=0BzESOSf028mLZkxBQng2d0QxN00>
- [18] <https://docs.google.com/open?id=0BzESOSf028mLbjgwbHpfWEpoSDg>
- [19] <https://docs.google.com/open?id=0BzESOSf028mLTjIPYmVJTUI TODA>
- [20] <https://docs.google.com/open?id=0BzESOSf028mLbXVCZINNNkVIU3M>
- [21] <https://docs.google.com/open?id=0BzESOSf028mLTkdaSm9RTE52T0E>
- [22] <https://docs.google.com/open?id=0BzESOSf028mLY0dXSFhjZE53OFU>
- [23] <https://docs.google.com/open?id=0BzESOSf028mLSTdoWV9DQUR5bHM>
- [24] http://circuitco.com/support/index.php?title=BeagleBone#Board_Reset_on_JTAG_Connect.28A3.2CA4.2CA5.29
- [25] <http://doc.qt.nokia.com/4.6/index.html>
- [26] http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer%E2%80%99s_Guide&oldid=130385
- [27] http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer%E2%80%99s_Guide&oldid=127948
- [28] http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer%E2%80%99s_Guide&oldid=106866
- [29] http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer%E2%80%99s_Guide&oldid=102587
- [30] http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer%E2%80%99s_Guide&oldid=94632
- [31] http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer%E2%80%99s_Guide&oldid=88367
- [32] http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer%E2%80%99s_Guide&oldid=85314
- [33] http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer%E2%80%99s_Guide&oldid=78069
- [34] http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer%E2%80%99s_Guide&oldid=62247
- [35] http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer%E2%80%99s_Guide&oldid=43627
- [36] <http://www.ti.com/sitara>
- [37] http://e2e.ti.com/support/dsp/sitara_arm174_microprocessors/default.aspx
- [38] <mailto:support@ti.com>

How to Build a Ubuntu Linux host under VMware



Return to the [Sitara Linux Software Developer's Guide](#)

Note - If behind a company firewall, be sure to set the necessary proxy settings within Ubuntu 10.04 LTS

Introduction

This guide demonstrates how to get a virtual Ubuntu Linux machine running with VMware under Windows XP. Please use only the 32-bit Ubuntu 10.04 release as this is what is called an LTS (Long Term Support). There are SDK scripts that will be checking for this release identity.

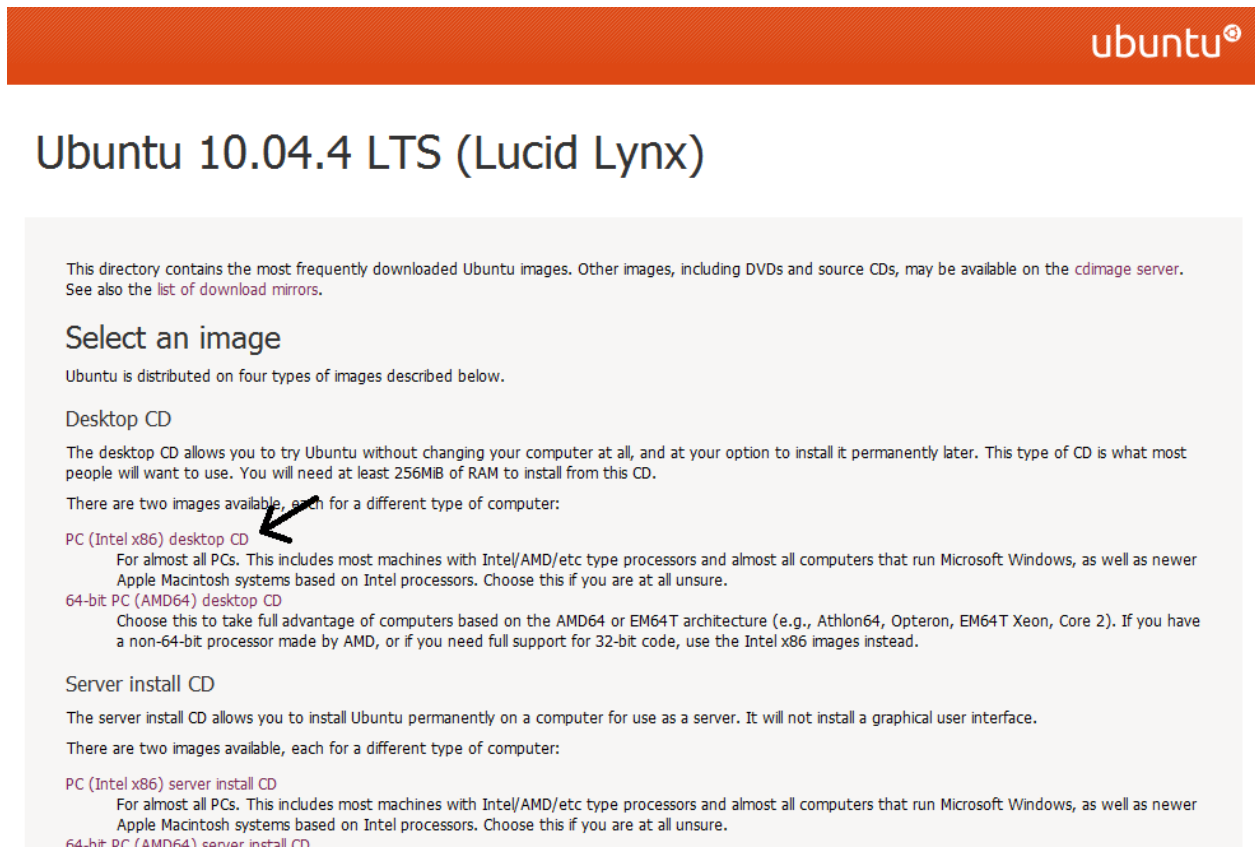
Requirements:

- Windows XP host with internet connection, at least 1G of RAM and 40G of free hard drive space.

The instructions here are for setting up a 40G virtual machine. The entire 40G is not taken at once, but as the machine is used and software is installed, the machine can grow and take up as much as 40G.

Download the Ubuntu 10.04 LTS ISO image

Get the Ubuntu 10.04 LTS CD ISO image from: <http://releases.ubuntu.com/lucid/> ^[1]. Select PC (Intel x86) desktop CD under the Desktop CD section.



This directory contains the most frequently downloaded Ubuntu images. Other images, including DVDs and source CDs, may be available on the [cdimage server](#). See also the [list of download mirrors](#).

Select an image

Ubuntu is distributed on four types of images described below.

Desktop CD

The desktop CD allows you to try Ubuntu without changing your computer at all, and at your option to install it permanently later. This type of CD is what most people will want to use. You will need at least 256MiB of RAM to install from this CD.

There are two images available, each for a different type of computer:

- [PC \(Intel x86\) desktop CD](#)
For almost all PCs. This includes most machines with Intel/AMD/etc type processors and almost all computers that run Microsoft Windows, as well as newer Apple Macintosh systems based on Intel processors. Choose this if you are at all unsure.
- [64-bit PC \(AMD64\) desktop CD](#)
Choose this to take full advantage of computers based on the AMD64 or EM64T architecture (e.g., Athlon64, Opteron, EM64T Xeon, Core 2). If you have a non-64-bit processor made by AMD, or if you need full support for 32-bit code, use the Intel x86 images instead.

Server install CD

The server install CD allows you to install Ubuntu permanently on a computer for use as a server. It will not install a graphical user interface.

There are two images available, each for a different type of computer:

- [PC \(Intel x86\) server install CD](#)
For almost all PCs. This includes most machines with Intel/AMD/etc type processors and almost all computers that run Microsoft Windows, as well as newer Apple Macintosh systems based on Intel processors. Choose this if you are at all unsure.
- [64-bit PC \(AMD64\) server install CD](#)

Click download and the follow instructions to download and save the ISO image (CD image).

Download VMware and install

Get VMware from: <http://www.vmware.com> ^[2]

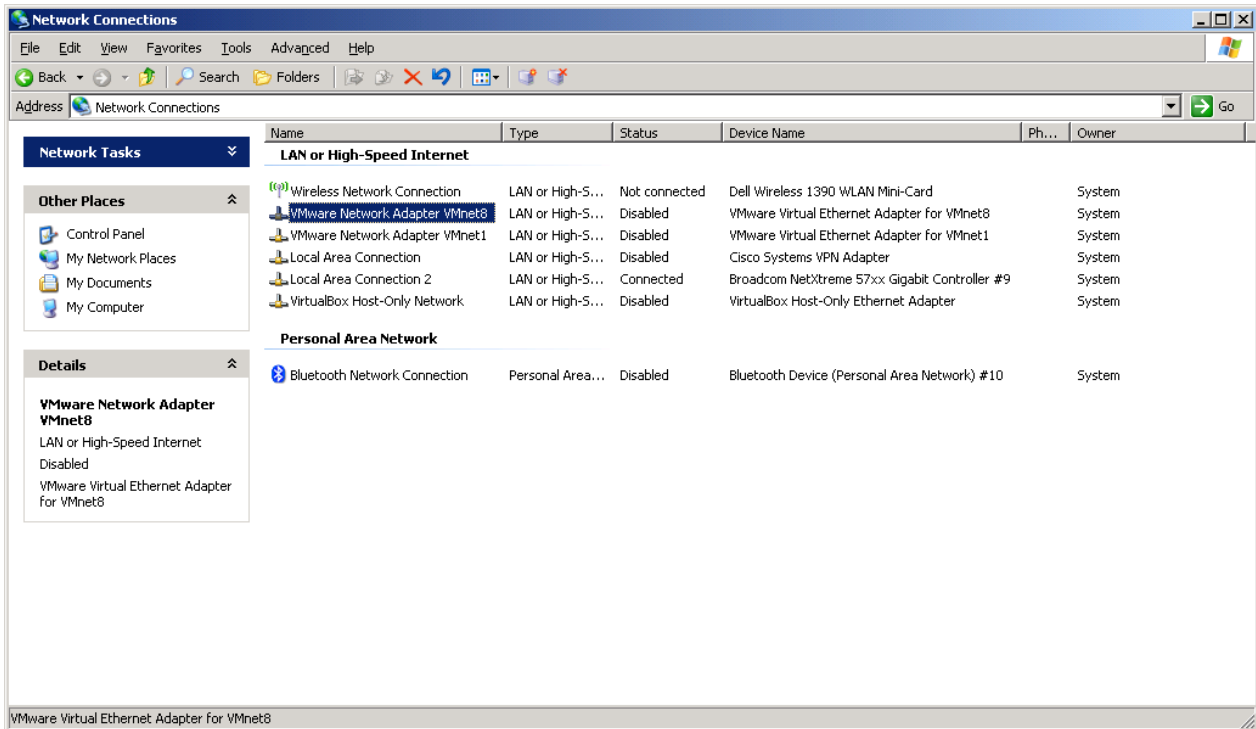
Vmware Player is a free download from the website and enables the user to create an entire virtual machine from scratch using just the ISO image downloaded from Ubuntu. It is necessary to sign up for an account at VMware in order to get to the download areas. The general steps to getting VMware are as follows:

- Login to the vmware website
- Select VMware Player from the products menu
- Follow the steps to download VMware Player

NOTE - We have tested with v3.0.1 and v3.1.2 with no known issues. As of Oct 26, 2010, v3.1.2 is the latest version.

- Run the executable to install VMware
- Accept license and all default settings.

After VMware is installed the Windows host will have two new (virtual) network adapters. These can be seen in the Windows host by looking under Control Panel --> Network Connections. If the virtual machine will use a bridged connection to an existing network, these virtual adapters should be disabled.



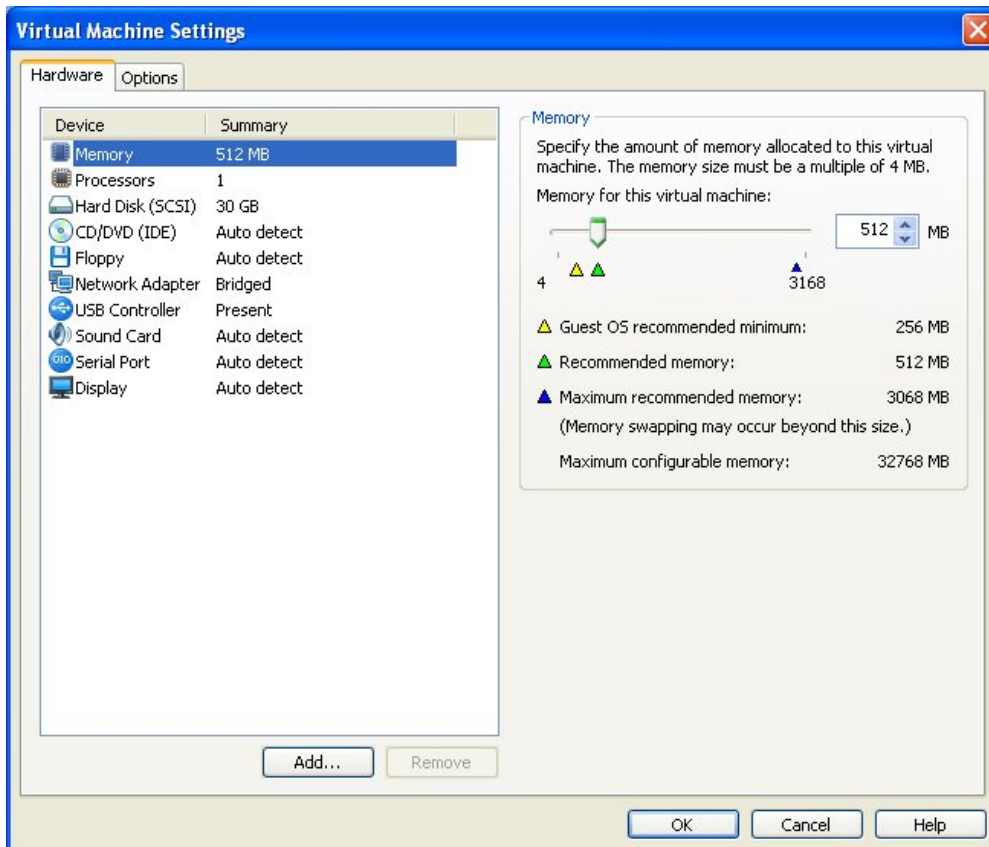
Create a New Virtual Machine with VMware

Before starting a new installation it is assumed that the Windows host has a proper internet connection to a DHCP server and that the Windows host has enough hard drive space for the new virtual machine.

The following steps are performed with VMware 3.0.1. The exact steps with other versions may vary slightly

- Start VMware.
- From the File menu select "Create a New Virtual Machine..."
- Choose to install the operating system later. Click "Next".
- Select Linux as the "Guest Operating System" and then choose Ubuntu as the "Version". Click "next".
- Provide a "Virtual machine name" and "Location" where the machine will be stored on the Windows host. The defaults are fine here. Click "Next".
- For "Maximum disk size (GB)" it is good to start with 40G if possible. This means that it will take up 40G on the Windows host. Make sure that the Windows host has at least this much before proceeding. It is also a good practice to tell VMware to split the virtual disk into 2G files. This will makes the image easier to copy and transport if necessary. Click "Next".
- Click "Finish" to complete the creation of the virtual machine.

The machine name will now be listed under the home page of VMware. It is necessary to modify some machine settings before playing the machine for the first time. Select the machine in the home page. Under the "VM" menu select "Settings..."



Click on CD/DVD and change the connection to "Use ISO image file". Click on "Browse..." and select the Ubuntu ISO image file that was previously downloaded. Click on Network Adapter and change the Network connection to "Bridged" and then check the box to "Replicate physical network connection state".

Adding a serial port to the virtual machine

If you plan to use a serial terminal application, a serial port must be added to the virtual machine. This port must be a physical serial port which exists on the host PC. Click on "Add..." and select "Serial Port". Click "Next". Choose "Use physical serial port on host". Click "Next". Click Finish. Click "Ok".

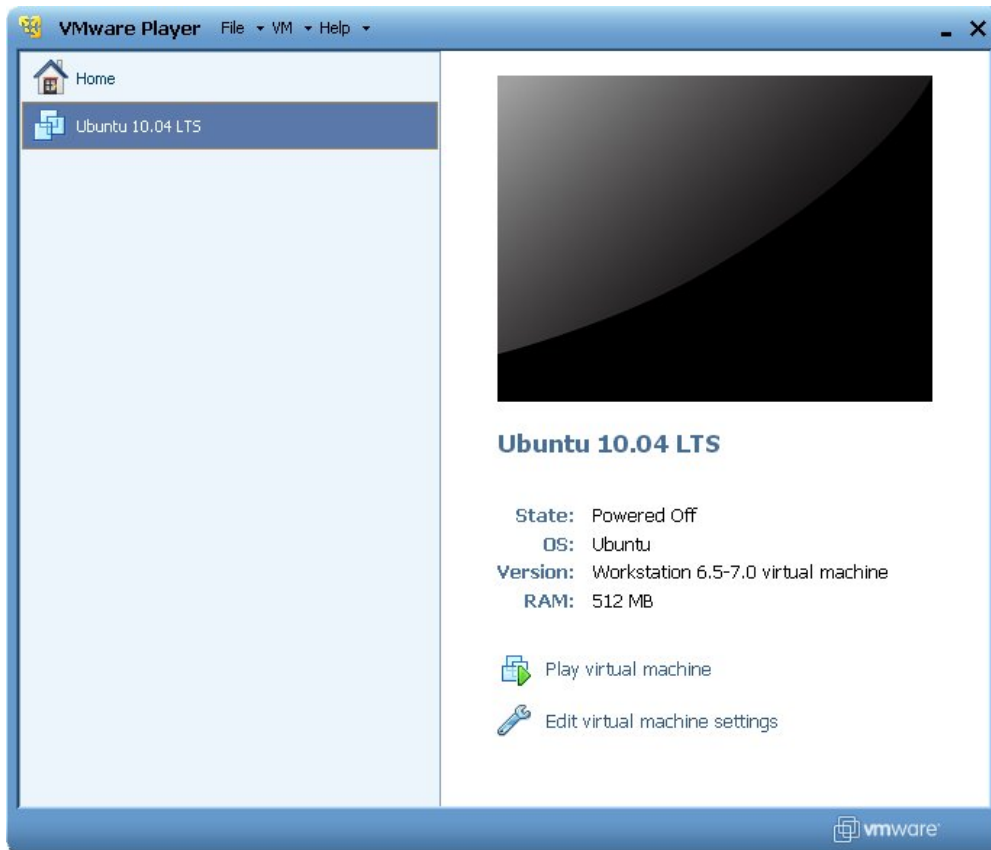
Since this is a physical port on the host PC, it cannot be used by the host PC and the virtual machine at the same time. When the virtual machine is started, the serial port will be unavailable for use by the host PC. If the serial port is being used at the time that the virtual machine is started, the virtual machine will not be able to access the serial port after it is booted up. So if you want the virtual machine to gain control of the physical serial port of the host PC, there can not be any application like hyperterminal or teraterm running on the host PC at the time that the virtual machine is started.

Further instructions for using the serial port with minicom inside of Ubuntu are here ^[3].

Minicom is the preferred application for use with the Sitara SDK. And the installation and setup of minicom is done automatically by the Sitara SDK installer.

Now click on "Play virtual machine". Since this is the first time starting the machine and the Ubuntu ISO image is in the virtual CD drive, the Ubuntu OS will install itself in the virtual machine.

Click through the Ubuntu installation, making the appropriate choices as you go. When prompted for a login name make the login name **user**. This will help with SDK installation scripts.



The full installation will take 20-30 minutes. When it completes the machine will reboot. The machine will now prompt for the login (**user**) and password.

After the machine reboots into Ubuntu it is helpful to take the Ubuntu ISO out of the virtual CD drive. Click on the VM menu and select "Settings...". Click on CD/DVD and change the connection from "Use ISO image file" to "Use physical drive". The actual drive letter can be selected from the drop down list. If there is no physical drive on the host machine, the CD/DVD device can be simply removed from the machine.

Install VMware Tools

VMware tools is a very useful addition to VMware. It allows you to resize the VMware screen and also allows cut-and-paste of text from the Ubuntu machine to and from the Windows host.

Later versions of VMware, such as VMware Workstation 7.x, include VMware Tools by default.

Click on the VM menu. Select "Install VMware Tools". The VMware tools are contained in an ISO image that VMware will automatically mount. This drive will show up on the Ubuntu desktop as if it were a disk in a DVD drive. There will be a single tarball on the drive. Copy this tarball to a location in the user filesystem. Extract the tarball and execute the Perl script from the tarball to install VMware Tools. The Perl script must be executed as a super-user. This is done in Ubuntu by pre-pending the command with "sudo". When prompted for a password, enter the password for the user account. In Ubuntu, there is no "root" account. However, the first user account created when Ubuntu is installed can become a super-user with the "sudo" command.

An example is shown below. This assumes that the tarball has already been copied to a directory **/home/user/VMwareTools** and extracted in place. Select all of defaults during installation of VMware Tools.

```
user@Ubuntu1004:~$ pwd
/home/user
user@Ubuntu1004:~$ cd VMwareTools/
user@Ubuntu1004:~/VMwareTools$ ls -l
```

```
total 104400
-rw-r--r-- 1 user user 106900818 2010-11-11 12:27 VMwareTools-8.4.5-324285.tar.gz
drwxr-xr-x 7 user user 4096 2010-11-11 12:26 vmware-tools-distrib
user@Ubuntu1004:~/VMwareTools$ cd vmware-tools-distrib/
user@Ubuntu1004:~/VMwareTools/vmware-tools-distrib$ ls -l
total 560
drwxr-xr-x 2 user user 4096 2010-11-11 12:26 bin
drwxr-xr-x 2 user user 4096 2010-11-11 12:26 doc
drwxr-xr-x 3 user user 4096 2010-11-11 12:26 etc
-r--r--r-- 1 user user 552155 2010-11-11 12:26 FILES
lrwxrwxrwx 1 user user 13 2011-02-14 14:27 INSTALL -> ./doc/INSTALL
drwxr-xr-x 2 user user 4096 2010-11-11 12:26 installer
drwxr-xr-x 17 user user 4096 2010-11-11 12:26 lib
lrwxrwxrwx 1 user user 31 2011-02-14 14:27 vmware-install.pl -> ./bin/vmware-uninstall-tools.pl
user@Ubuntu1004:~/VMwareTools/vmware-tools-distrib$ sudo ./vmware-install.pl
```

Confirming a Valid Network Connection

After logging into the machine for the first time, bring up a terminal window. This can be found under the Applications menu in Ubuntu. Applications --> Accessories --> Terminal. Type **pwd** in this terminal. This should return **/home/user**. Now type **ifconfig**. This should return information about the network connection. Under **eth0** the IP address should be similar (but not the same) as the IP address owned by the Windows host.

```
user@Ubuntu1004:~$ pwd
/home/user
user@Ubuntu1004:~$ ifconfig
eth0 Link encap:Ethernet HWaddr 00:0c:29:da:a8:6e
inet addr:128.247.107.65 Bcast:128.247.107.255 Mask:255.255.254.0
inet6 addr: fe80::20c:29ff:feda:a86e/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:759 errors:0 dropped:0 overruns:0 frame:0
TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:62873 (62.8 KB) TX bytes:4937 (4.9 KB)
Interrupt:19 Base address:0x2024

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:12 errors:0 dropped:0 overruns:0 frame:0
TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:720 (720.0 B) TX bytes:720 (720.0 B)
user@Ubuntu1004:~$
```

How to Read a USB SD Card Reader in VMware

When a USB card reader with an SD card is inserted into the USB slot of the host machine, the virtual machine will automatically detect the drive and mount partitions from the SD card. If this does not happen automatically it can be done manually by clicking the VM menu and selecting Removable Devices and then selecting the card reader from the sub-menu under Removable Devices. From this sub-menu it is possible to connect or disconnect the USB card reader.

Using the Target as an SD Card Reader

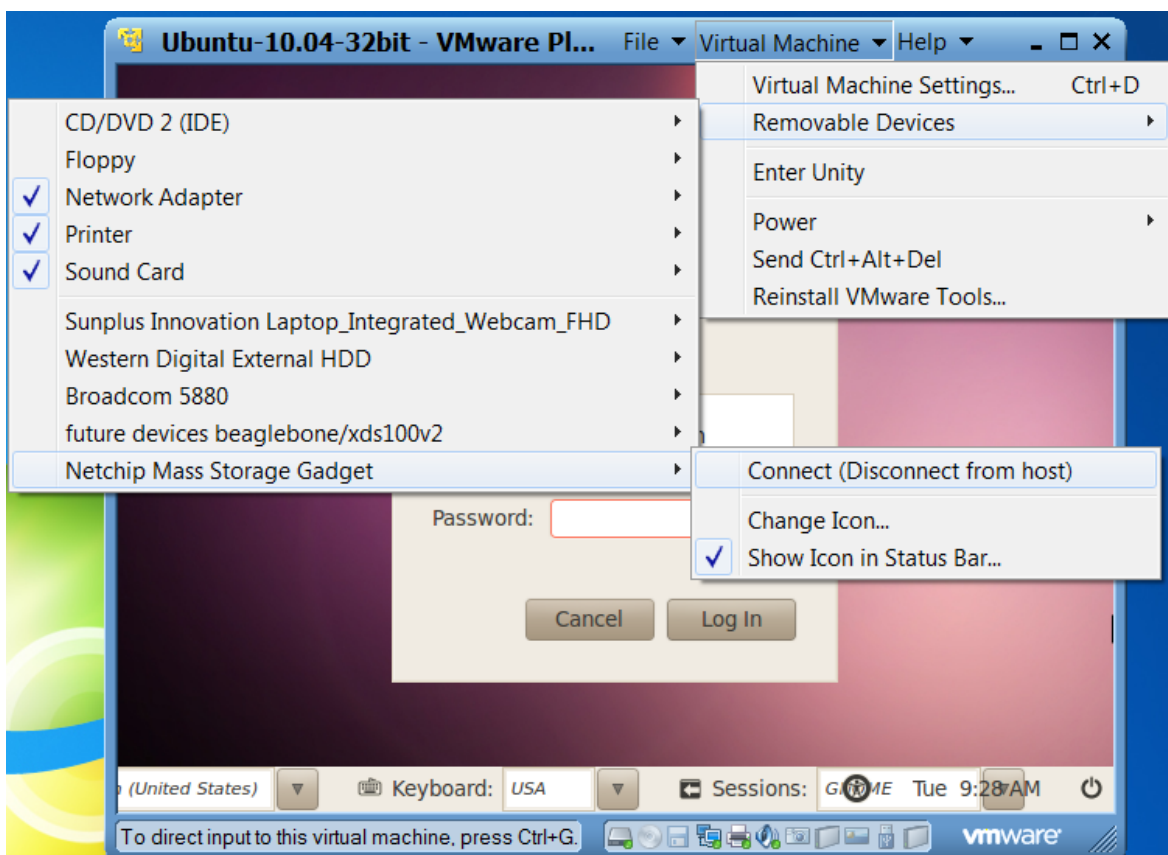
Some devices such as the BeagleBone also act as the SD card reader when they are booted. For these devices you can import the device as an SD card reader into the VMWare using the following steps:

1. Once the board is powered on with the Sitara Linux SDK SD card installed you will receive messages like the following letting you know that the device can be connected to the VMWare.





2. Click **OK**
3. To connect the mass storage device (the SD card contents) select **Virtual Machine -> Removable Devices -> Netchip Mass Storage Gadget -> Connect (Disconnect from Host)**



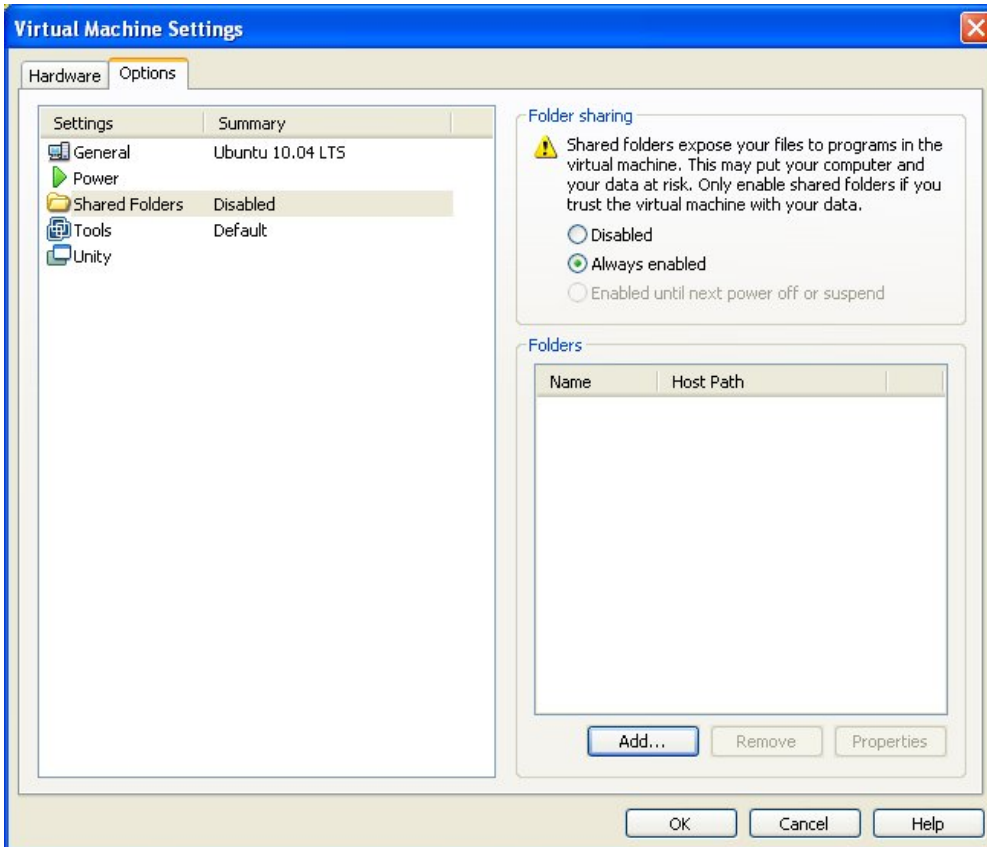
4. You will be prompted that you are about to remove a device from the Host system and connect to the virtual machine. Select **OK** to connect the device.

5. The device should now be available within the virtual machine

How to Set up Shared Folder in VMWare

The following steps show how to enable Shared Folders within VMware which allows you to easily share files between Ubuntu 10.04 running on VMware and your Windows host.

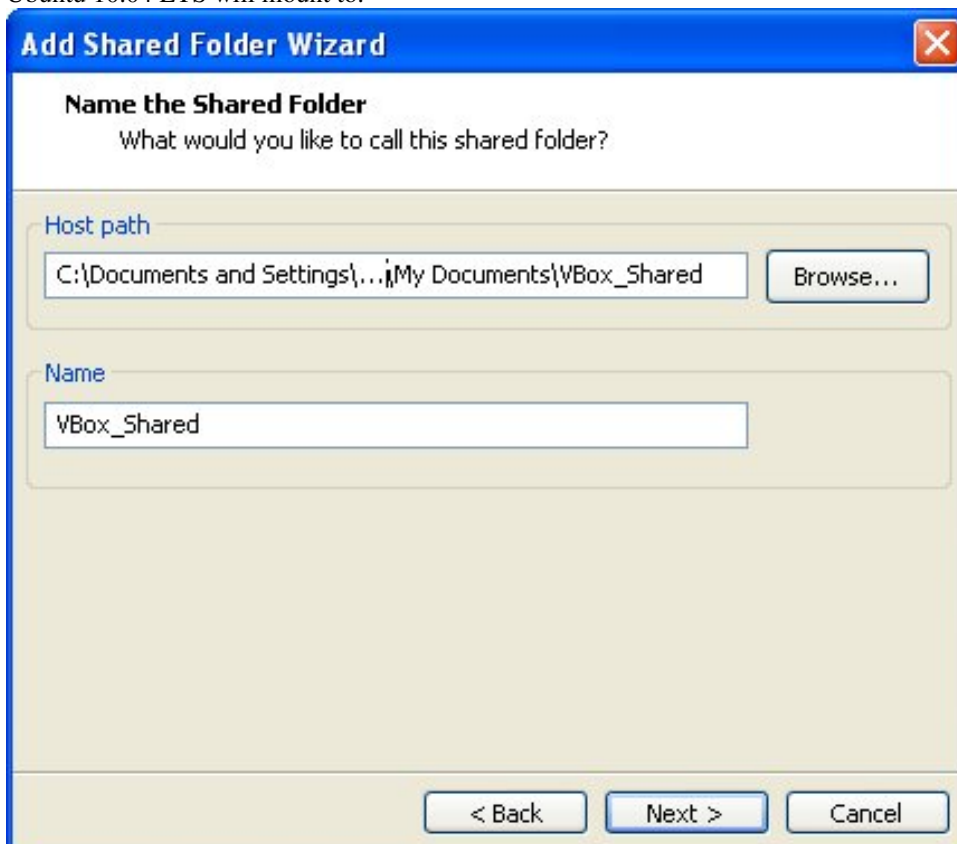
1. Under Virtual Machine Settings, the Options tab, select *Shared Folders* and *Always enabled*.



2. Click Add...and the following dialog should display.



3. **Browse the Windows folder you want to Share.** And provide a Name for that folder. This Name is what Ubuntu 10.04 LTS will mount to.



4. Ensure **Enable this share** is checked (should be default). And click Finish.



5. **Start your virtual machine and log into Ubuntu 10.04 LTS.** Create a Desktop short-cut to the Shared Folder you just set up.

- Right click on your Ubuntu 10.04 LTS desktop
- Click *Create Launcher...*
- Change Type to *Location*
- Enter a *Name*
- Next to Location:, enter */mnt/hgfs/shared_folder_name*

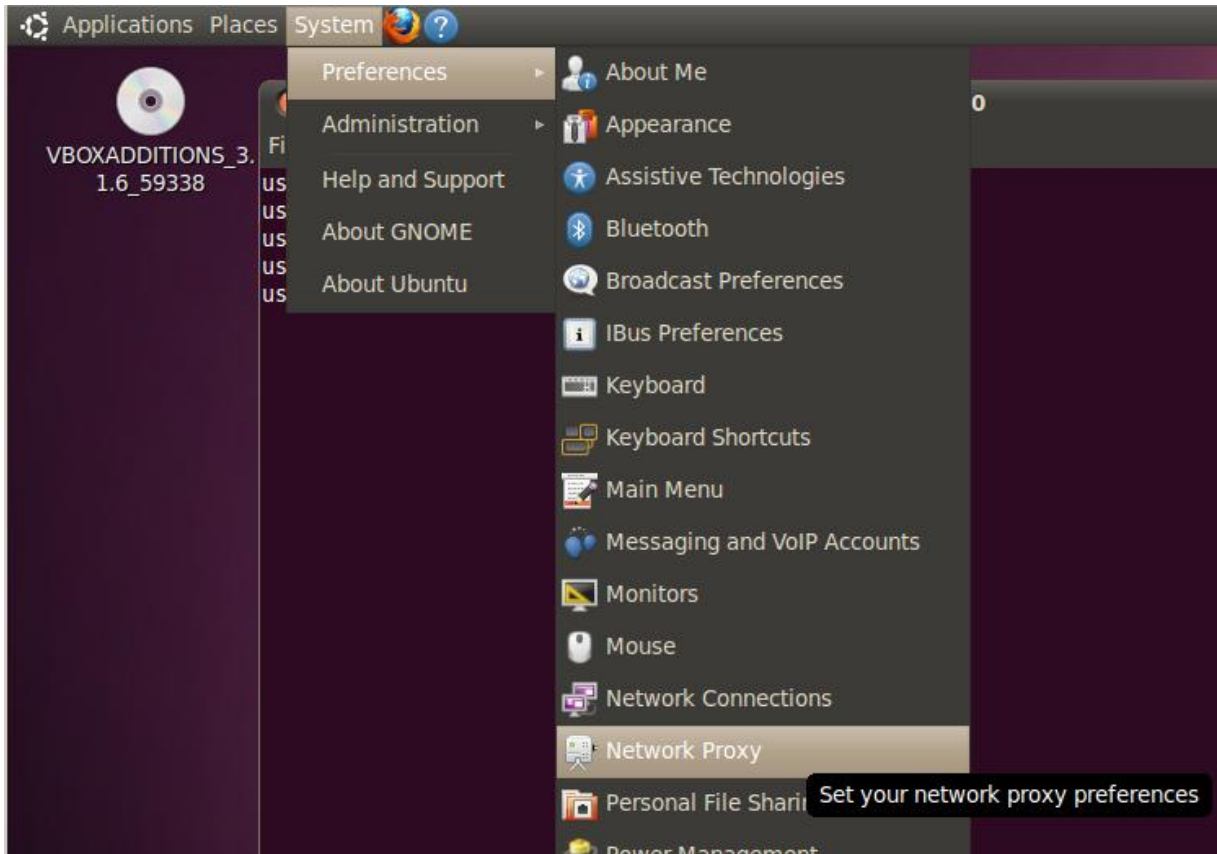
NOTE - shared_folder_name is what ever name your provide in Settings

After clicking OK, you should have a desktop shortcut to your Shared Folder.

Configuring a Proxy in Ubuntu

If your network is behind a firewall you will need to configure the network proxy for Ubuntu in order to successfully download the applications required to complete your development environment or to browse the Internet on your Linux Host machine.

To configure the network proxy in Ubuntu go to System-Preferences and click Network Proxy as seen below.



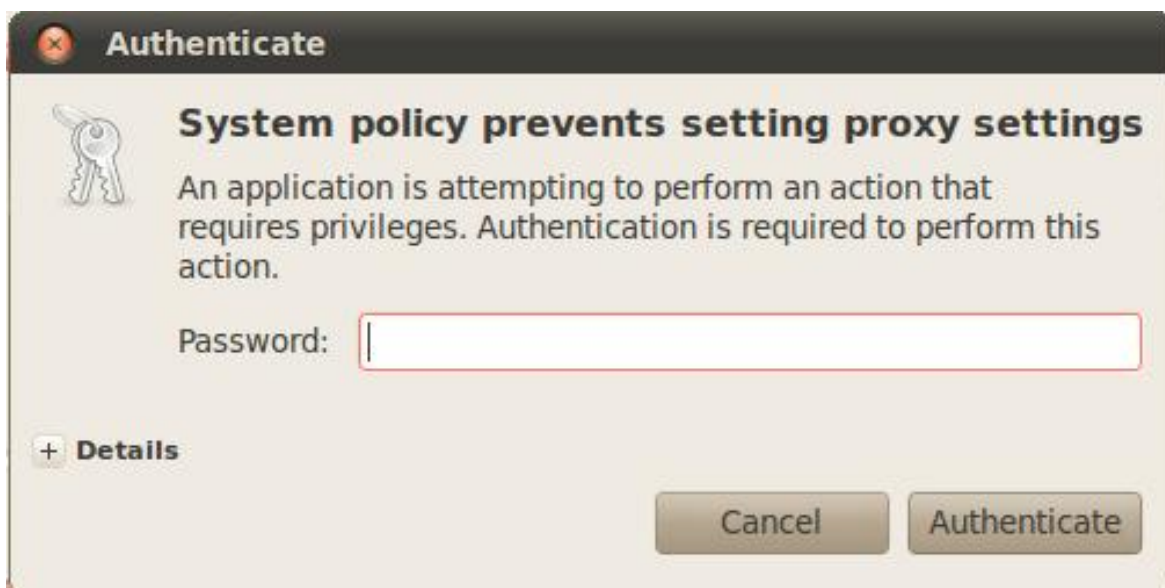
As seen in the image below, click Manual Proxy Configuration. Specify the HTTP proxy server used by your company. You may find this information under your Windows OS inside the Internet Explorer Network Connections. Be sure to specify the port.

NOTE - www.proxyserver.com is not a valid HTTP proxy. It is shown as an example only. You need use the HTTP proxy server & Port used by your company.

Make sure you check "Use the same proxy for all protocols". Also be sure to click "Apply System-Wide". Then Close



Finally you will be asked to enter your password in order to set your network proxy information. This is typically the same password you used to log into Ubuntu on boot.



You should now be able to browse the Internet using Firefox within Ubuntu.

Configuring apt-get to use the proxy explicitly

In case you are unable to use apt-get to download and install packages, then you may need to explicitly setup the proxies in /etc/apt.

Create /etc/apt/apt.conf file as root

```
host$ sudo vi /etc/apt/apt.conf
```

Now add the following text to the file.

```
Acquire::ftp::proxy "ftp://www.proxyserver.com:80";  
Acquire::http::proxy "http://www.proxyserver.com:80";  
Acquire::https::proxy "https://www.proxyserver.com:80";
```

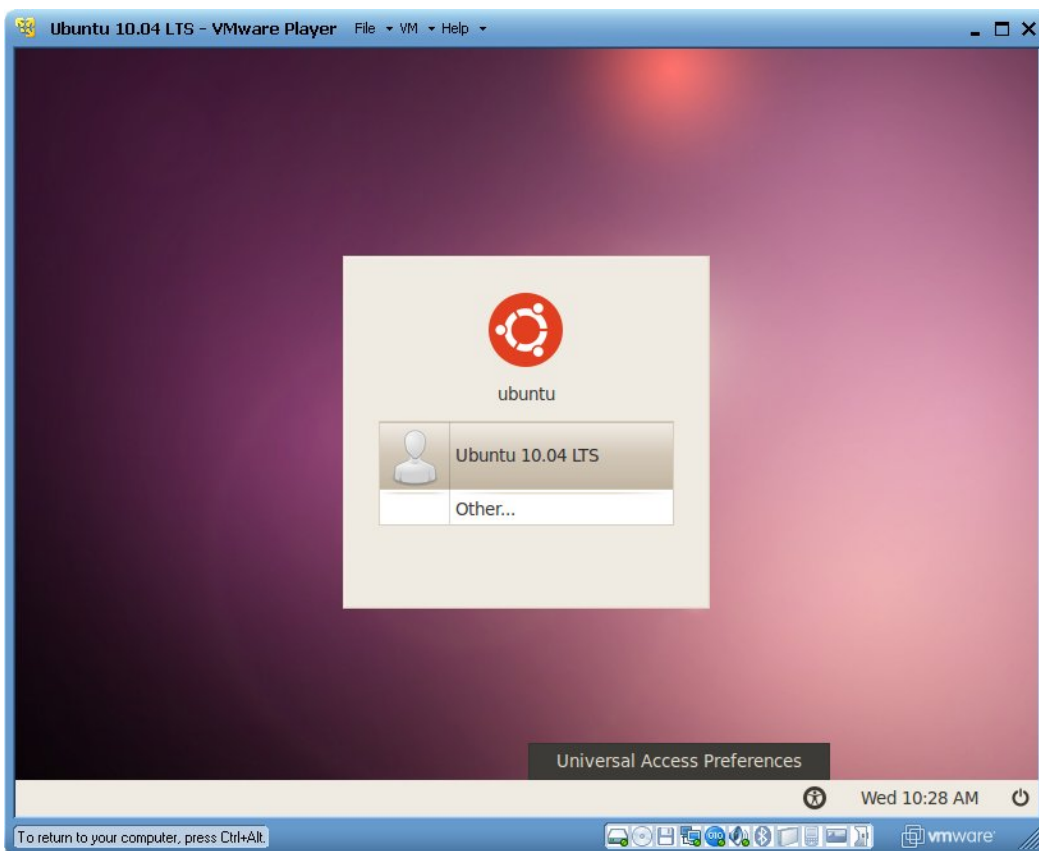
Replace www.proxyserver.com with your proxy server's address and 80 with your proxy server's port.

Keyboard does not Work in Ubuntu 10.04 LTS

If your physical keyboard does not work when you first start VMWare running Ubuntu 10.04 LTS, then you may want to try the following steps to resolve the issue. This assumes your mouse works properly.

NOTE - The following makes use of the on-screen keyboard provided in Ubuntu 10.04 LTS. Once you are able to log into Ubuntu 10.04 LTS using the on-screen keyboard, issues with the physical keyboard should be resolved.

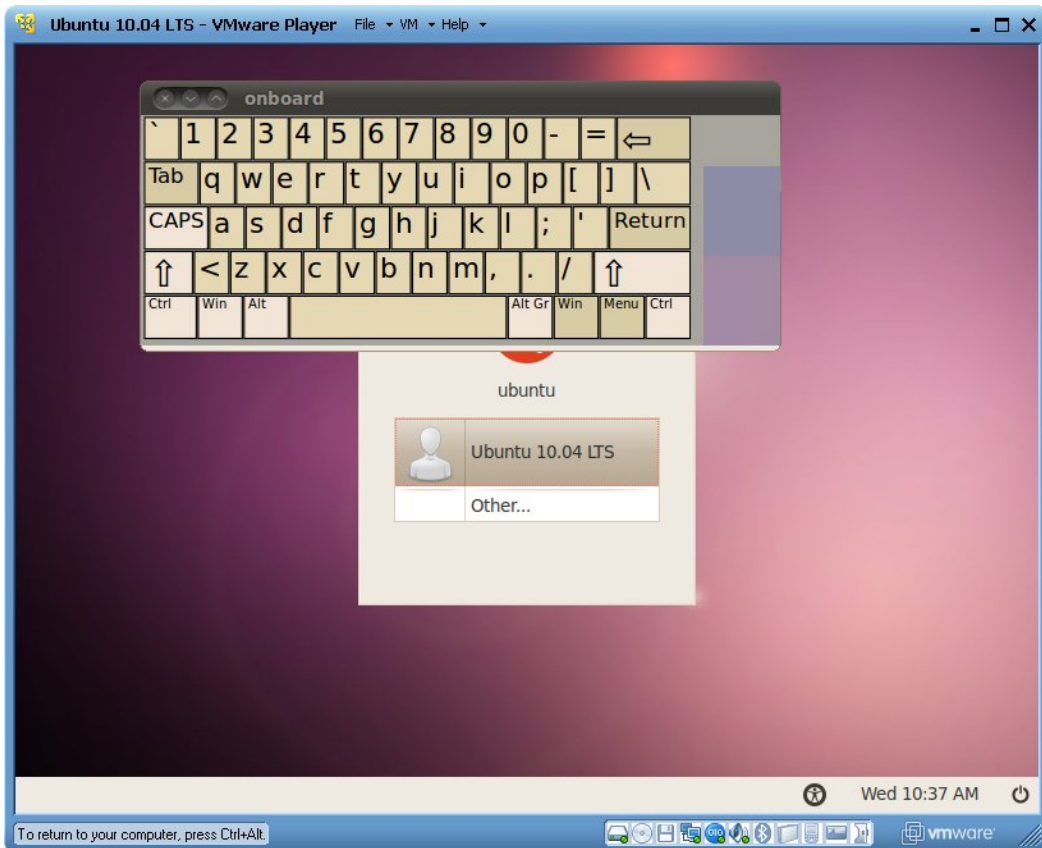
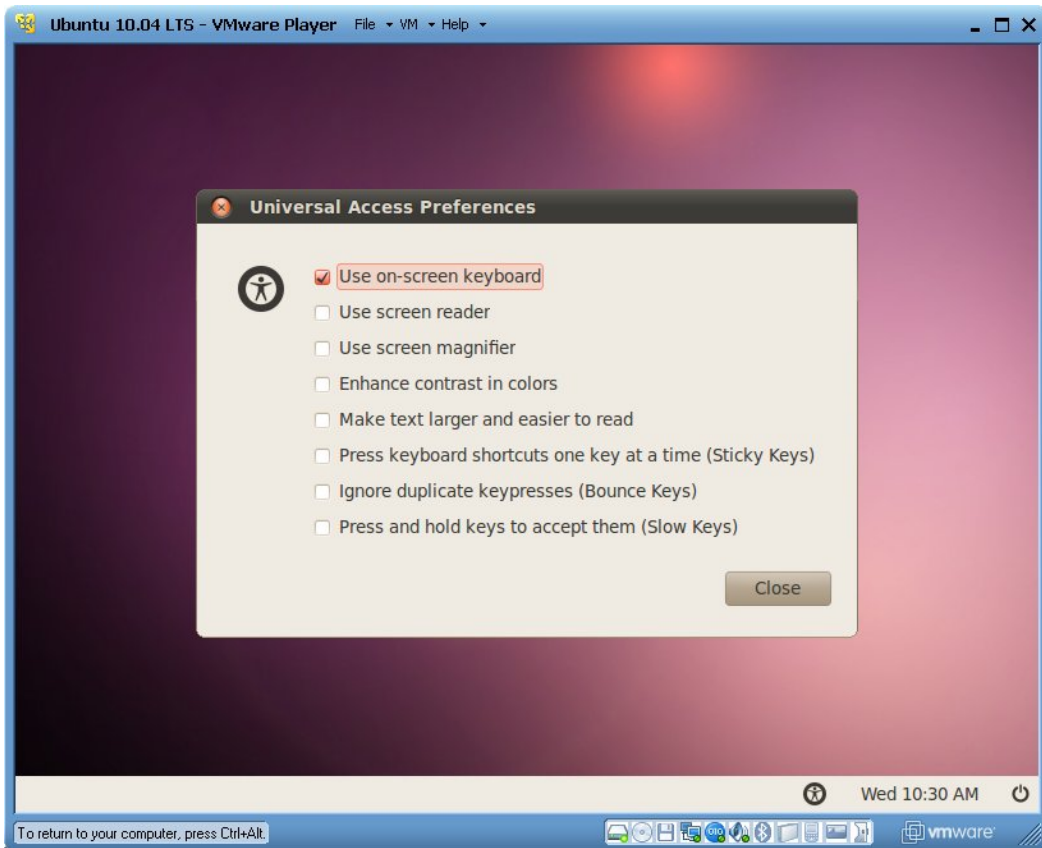
1. Open the *Universal Access Preferences* by left clicking on the icon with the man in the circle. This will display the *Universal Access Preferences* button. Click on *Universal Access Preferences*.



2. Select the first option to *Use on-screen keyboard*. Once you *Close*, the on-screen keyboard should display allowing you to log into Ubuntu 10.04 LTS.

NOTE - If the on-screen keyboard does not display, *Shutdown* Ubuntu 10.04 LTS and restart your Ubuntu 10.04 LTS virtual machine again in VMWare. Once Ubuntu 10.04 starts again the on-screen keyboard should be

displayed.



3. Modify `/etc/default/console-setup` using `gedit` to enable the physical keyboard

```
username@ubuntu:~$ sudo gedit /etc/default/console-setup
```


NOTE - You will need sudo access and therefore must enter your password to access this file.

At the bottom change the following lines:

```
<original>
XKBMODEL="SKIP"
XKBLAYOUT="us"
XKBVARIANT="U.S. English"
XKBOPTIONS=""

<new changes>
XKBMODEL="pc105"
XKBLAYOUT="us"
XKBVARIANT=""
XKBOPTIONS=""
```

The physical keyboard should now work from here out. You may also now disable the on-screen keyboard under *Universal Access Preferences*.

Adding Hard Drive space to the Virtual Machine

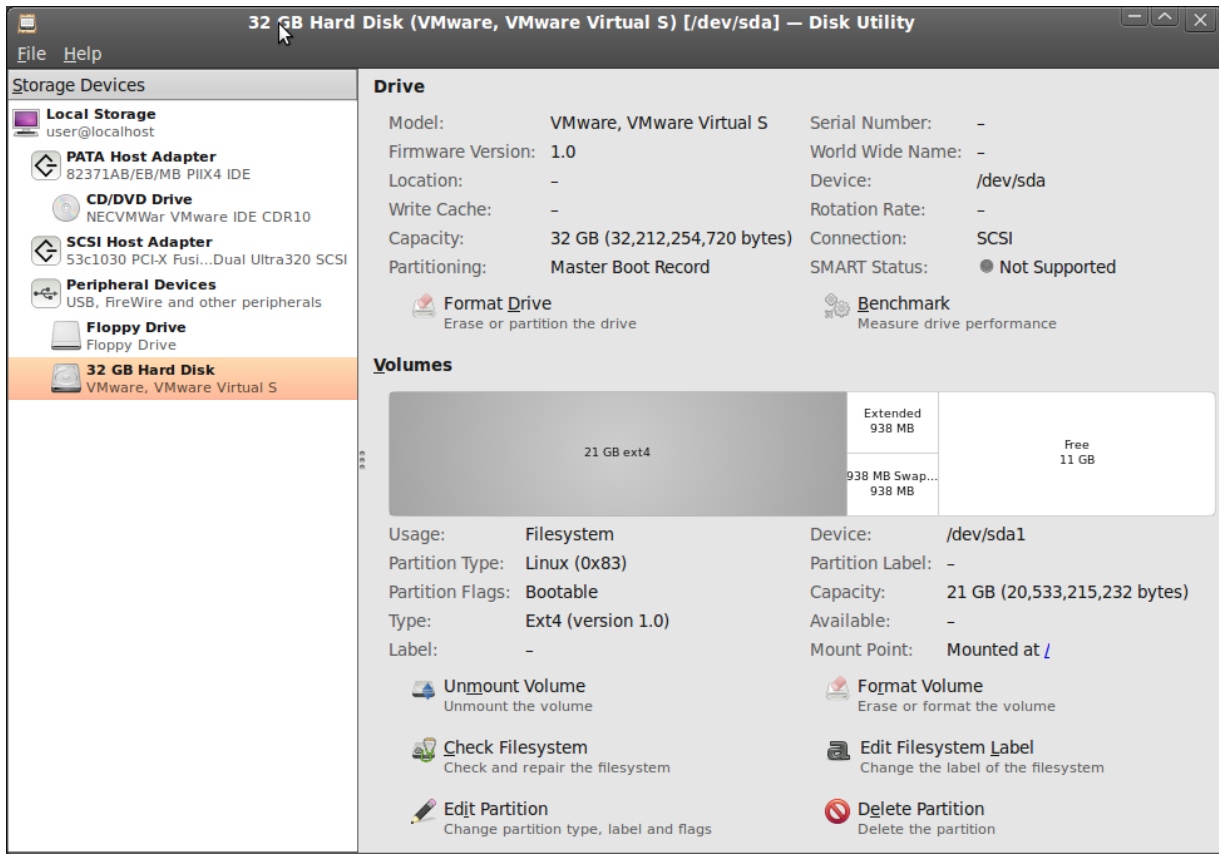
If you followed the instructions in this Wiki and created a 40G Ubuntu 10.04 machine, you will find that this is a good size for the installation of one Sitara SDK and all of its tools. However, there may come a time when you run out of space in the virtual machine. If you have plenty of hard drive space on the Windows host machine then it is very easy to expand the drive of the virtual machine.

The instructions here are for VMware player version 3.1.5 and later.

In VMware, before starting the virtual machine, click on "Edit virtual machine settings" and then click "Hard Disk" on the Hardware tab. In the Utilities menu select "Expand..." and enter a new value for the "Maximum disk size". Click "Expand" then "Ok" to close the settings menu. Now play the machine as usual.

When the machine boots up, Click System --> Administration --> Disk Utility. This will bring up a view of all the storage devices on the system. The Hard Drive here should show a value equal to the amount that was just edited in the machine settings. Click on the Hard Drive and a Volume view should be presented. Here you should see the original Hard Drive as a formatted ext volume. And you should see an additional "Free" partition that is so far unformatted. Click on this "Free" partition and then click the "Create Partition" button. Enter a name for this partition. Now click the "Mount Volume" button. The drive should now show up on the desktop and it is ready to use for additional data space.

The example below shows a machine that was originally 20G and was expanded to 30G. This creates a new 10G partition.



References

- [1] <http://releases.ubuntu.com/lucid/>
- [2] <http://www.vmware.com>
- [3] http://processors.wiki.ti.com/index.php/Setting_up_Minicom_in_Ubuntu

Sitara SDK Installer



Overview

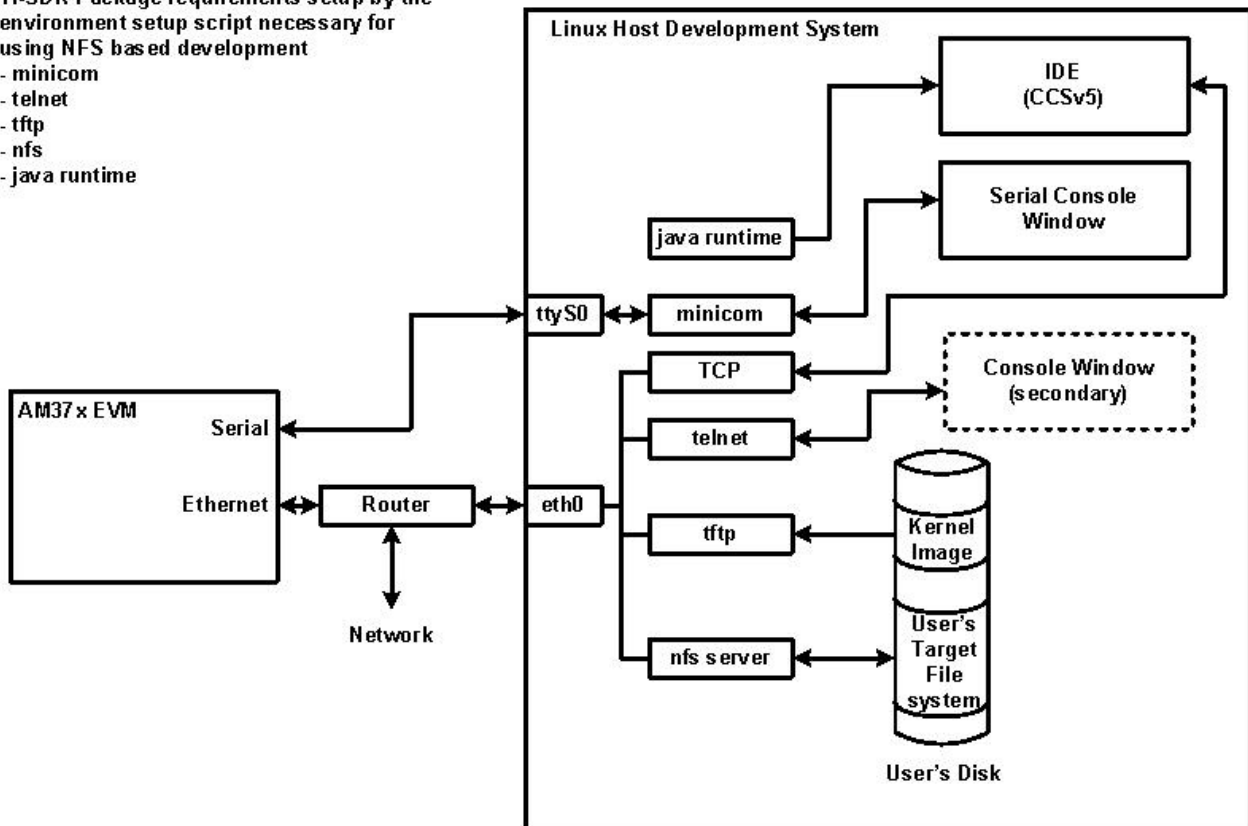
The SDK Installer (ti-sdk-amxx-vx.x.x.x) will install the necessary components to start your development on the AMxx microprocessor. The SDK consists of source for the Matrix App launcher starting point application, a development filesystem, a target filesystem, an IDE (CCSv5), Pin Mux Utility, Flash Tool applications, the PSP and documentation.

The Sitara SDK now includes the GCC toolchain from Open Embedded. The ti-sdk was built and tested against a specific Linux Distribution name and version, Ubuntu 10.04. Note this **does not** prevent the user from installing the SDK on other Linux distributions.

To assist the user in getting a to a point of starting development the installer contains an Environment Setup Script ^[1] that will run with the Linux Distribution specified for the ti-sdk. This particular script sets up several functions needed by the SDK such as the Java runtime for the CCSv5 IDE. While it is written for the current distribution the user can modify the script to fit their particular distribution. Please see the picture for a block diagram of the development environment setup for ti-sdk. Please also note that this script is specifically tied to Ubuntu 10.04 as will make specific package installations that are only known to work on this release.

TI-SDK Package requirements setup by the environment setup script necessary for using NFS based development

- minicom
- telnet
- tftp
- nfs
- java runtime



SDK Installer Execution Steps

1. Confirm

User is to confirm if loading the ti-sdk is ok. This is important to note in the user is trying to over-install on an existing directory and has made changes to the directory.

2. Directory Install Location

The user will be prompted for a location on where to put the ti-sdk. An example is given below.

3. Installation of software

The software is installed.

4. Environment Setup Script

If the user is running the supported Linux distribution they will have the option of running the environment script that will set download from Ubuntu several packages that will enable tftp, nfs, serial console and the IDE. This is shown below. This script is run after the installer completes and should only be run once. If the user decides to re-install the environment setup script does not need to be run again.

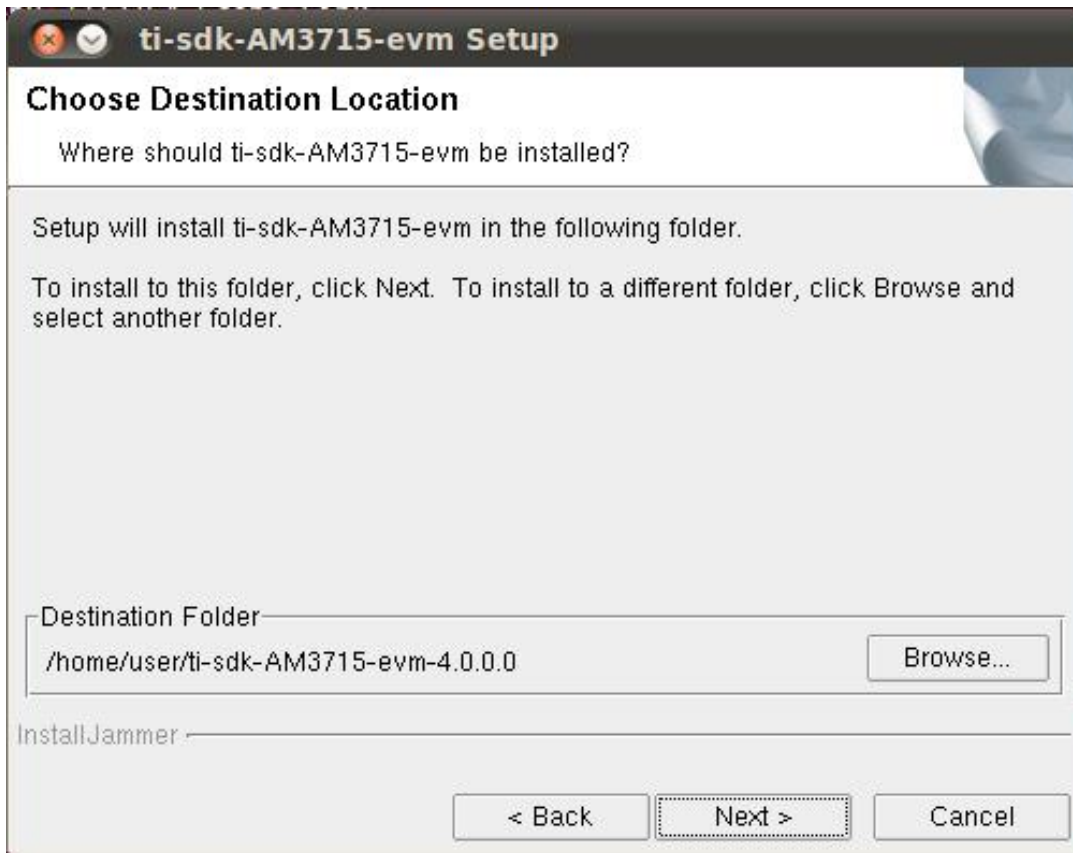
How to Run the SDK Installer

Before running the SDK Installer, the SD Card from the EVM box needs to be mounted to your Linux Host PC (using a USB SD Card reader). The SDK Installer is found in the START_HERE partition of the SD card. Run the SDK Installer by double clicking on it within your Linux host PC. Alternatively, bring up a terminal window and change directories to the where the START_HERE partition is mounted and run the SDK Installer with the following command:

```
[prompt]:~$ ./ti-sdk-amxx-evm-x.x.x.x-Linux-x86-Install
```

Where to install the ti-sdk package

The default selection of where to install is the user's home directory. In this particular example the name of the user is user.



Code Composer Studio v5 IDE Installer

If you checked the "Install Code Composer" checkbox in the SDK Installer, the CCSv5 installer will get kicked off prior to exiting from the SDK installer. The wiki page [Running_the_CCSv5_installer](#)^[2] provides information about the CCSv5 installer.

Archived Versions

Archived - Sitara Linux SDK Installer User's Guide

- Sitara SDK 5.03 - SDK Installer User's Guide (archived)^[3]
- Sitara SDK 4.01 - SDK Installer User's Guide (archived)^[4]

References

- [1] http://processors.wiki.ti.com/index.php/SDK_Setup_Script
- [2] http://processors.wiki.ti.com/index.php/Code_Composer_Studio_v5_Preview_Edition_Users_Guide#Running_the_CCSv5_installer
- [3] http://processors.wiki.ti.com/index.php?title=Sitara_SDK_Installer&oldid=85510
- [4] http://processors.wiki.ti.com/index.php?title=Sitara_SDK_Installer&oldid=50935

Sitara Linux SDK Setup Script



Return to the [Sitara Linux Software Developer's Guide](#)

Overview

After installation of the SDK on the Linux host, the setup script should be run to prepare the host for software development. Some of the tasks require administrator privileges. The script will prompt you when these administrator privileges are required. The setup script does the following things:

- Verification that the Linux host is the recommended Ubuntu LTS version
- Installation of required host packages
- Target FileSystem installation
- NFS setup
- TFTP setup
- Minicom setup
- uboot setup
- Load uboot script

How to run the setup script

The Setup Script is located in the Sitara SDK installation directory. By default, this directory has a name that has the form

"ti-sdk-<Hardware-Platform>-<Version>". Change to that ti-sdk install directory. Then run the script:

```
[host prompt]:~$ ./setup.sh
```

NOTE - The Setup Script will first check to see if the user is running the recommended Ubuntu Long Term Support (LTS) distribution, if not it will exit. If the user is running on a different Ubuntu version or another Linux distribution, they are encouraged to modify the environment setup script to match their distribution.

See which version of Ubuntu is currently supported here: [Start_your_Linux_Development](#)

Detailed step by step description through the setup script

The following sections describe in more detail how to run the script and what is doing.

Installation of Required Host Packages

This section will check to make sure you have the proper host support packages to allow you do the following tasks:

- telnet
- bring up menuconfig, the kernel configuration tool
- mounting filesystem via nfs
- tftp
- bring up minicom
- rebuild u-boot

If your host lacks any of the needed packages, they will automatically be installed in this step.

Note! This command requires you to have administrator privileges (sudo access) on your host.

The command below is an example of what this script is doing. The actual packages may vary for different releases:

```
[host prompt]:~$ sudo apt-get install xinetd tftpd nfs-kernel-server minicom build-essential libncurses5-dev uboot-mkimage autoconf automake
```

Add to Dialout Group

Note! This part requires you to have administrator privileges (sudo access)

A unique step is required for users using Ubuntu 12.04+. By default the user does not have the proper permissions to access a serial device (ex ttyS0, ttyUSB0, etc...). A user must be apart of a "dialout" group to access these serial device without root privileges.

During this step the script will check if the current Linux user is apart of the dialout group. If not the current Linux user will automatically be added to the dialout group.

The Linux user will still be required to use sudo when accessing the serial device until the user logs out and then logs back in.

Target FileSystem Installation

This step will extract the target filesystem.

Note! This part requires you to have administrator privileges (sudo access)

The default locations is: /home/user/ti-sdk-amxx-evm-x.x.x.x/targetNFS

```
In which directory do you want to install the target filesystem?(if this directory does not exist it will be created)
```

```
[ /home/user/ti-sdk-amxx-evm-x.x.x.x/targetNFS ]
```

You can override the default location by typing in another location or by hitting <Enter> you can accept the default location. This can take a little time to untar and unzip the filesystem.

If you have run this script more than once and the filesystem already exists, you will be asked to either:

- rename the filesystem
- overwrite the filesystem
- skip filesystem extraction

(see actual prompt below)

```
/home/jlance/ti-sdk-am335x-evm-05.04.00.00/targetNFS already exists
```

```
(r) rename existing filesystem (o) overwrite existing filesystem (s) skip filesystem extraction
```

NFS Setup

This step will allow you to export your filesystem which was extracted in the previous step.

Note! This command requires you to have administrator privileges (sudo access) on your host.

- This step adds the path to root filesystem from the previous step to the file /etc/exports on your host.
- The NFS kernel daemon is then stopped and then restarted to make sure the exported file system is recognized.

TFTP Setup

This section will setup tftp access on your host.

Note! This command requires you to have administrator privileges (sudo access) on your host.

```
-----
Which directory do you want to be your tftp root directory?(if this directory does not exist it will be created for you)
```

```
[ /tftpboot ]
```

The default location is /tftpboot which is off of the root directory on your linux host and requires administrator privileges. You can hit <Enter> to select the default location or type in another path to override the default. Then the following task occur:

- A tftp config file is created for you on your host at /etc/xinetd.d/tftp
- The tftp server is stopped and then restarted to insure the changes are picked up.

If you have run this script more than once or the filename already exists, you will be asked to select one of the following options.

- rename the filesystem
- overwrite the filesystem
- skip filesystem extraction

Minicom Setup

This step will set up minicom (serial communication application) for SDK development

```
Which serial port do you want to use with minicom?
[ /dev/ttyS0 ]
```

For most boards, the default /dev/ttyS0 should be selected. For Beaglebone which has a USB-to-Serial converter, just hit enter and the proper serial port will be setup in a later step.

- A minicom configuration will be saved for you at /home/user/.minirc.dfl
- The old configuration if there was one will be saved at /home/user/.minirc.dfl.old

The configuration saved to /home/user/.minirc.dfl can be changed, see the Software Development Guide for more information.

NOTE: If you are using a USB-to-Serial converter, your port should be configured for /dev/ttyUSBx

uboot Setup

This section will create the necessary u-boot commands to boot up your board based on your answers to the questions below.

The script will detect your ip address and display it. You can override the detected value by entering an alternate value. See sample below:

```
-----
This step will set up the u-boot variables for booting the EVM.
Autodetected the following ip address of your host, correct it if necessary
[ xxx.xxx.xxx.xxx ]
```

Next you will be prompted where you prefer your kernel and file system to be located.

- Kernel location
 - TFTP - located on your Host in your designated /tftpboot directory
 - SD card - located in the 1st partition named "boot" of your SD card
- Filesystem location
 - NFS - located on your Host. The location is where the file system was extracted in an earlier step.
 - SD card - located on the 2nd partition named "rootfs" of your SD card.

Next if you have selected TFTP, you will be prompted which uImage you want to boot using TFTP. You will be given a list of existing uImage's and you can type one in from the list or hit <Enter> to select the default option. The default option will be the uImage corresponding to the SDK installation. This will be used in the next step to create the necessary u-boot options to boot up your device.

Load uboot Script

This section creates a minicom script or a uEnv.txt file which will be used by u-boot to provide the necessary commands to boot up in the preferred configuration.

- For boards with straight serial connectors, a minicom script is created
- For boards like beaglebone with a USB-to-Serial configuration, then a uEnv.txt script is created and placed in the /boot partition of the SD card.

NOTE: For devices which create a uEnv.txt, the device must already be booted up with the USB-to-Serial connector attached to the Host. Further the Host must recognize the boot and START_HERE partitions.

AMSDK START HERE Script



Return to the [Sitara Linux Software Developer's Guide](#)

Overview

For some devices like the Beaglebone which use a built in USB to Serial adapter a special script called START_HERE.sh has been created to assist in running the out of box software and installing the AMSDK. This script will be found when the board is booted into Linux and the START_HERE partition of the SD card is auto mounted on your Linux host machine. At that point the board is acting as an SD card reader to provide you access to the contents of the START_HERE partition which has the AMSDK installer, Quick Start Guides, and CCS installer.

Executing the Script

After following the instructions to connect your board to your Linux host machine using USB, the board will boot into Linux. At this point you should find two partitions mounted. These are the **boot** and **START_HERE** partitions. On the START_HERE partition you will find a script called **START_HERE.sh**. To execute this script double-click the script and if prompted select **Run in Terminal**. This will open a terminal window which will prompt you for which actions you wish to perform.

Launching Matrix

At this point a terminal window should be open and you should see output like the sample below asking you if you would like to launch a browser to connect to matrix remotely.

```
BeagleBone Startup
```

```
Thank you for purchasing the BeagleBone!
```

```
The BeagleBone contains everything you need for development, including a USB card reader, USB-to-Serial converter and a XDS100v2 emulator - and it's all over the same USB cable! With optional daughter cards, components like display and Wifi can be added with ease.
```

```
The BeagleBone is fully booted, but development can't begin until the SDK is installed and the appropriate development environment is configured.
```

```
However, before beginning development, would you like to explore the remote version of Matrix GUI v2? Matrix GUI provides an easy way explore the capabilities of the BeagleBone through your host web browser. (y/n) [y]
```

Pressing *Enter* (default is yes) or entering *y* + *Enter* will start your local browser and connect to the matrix application running on the board. Entering *n* will skip this step. From this browser window you can launch additional applications on the board and see their results.

NOTE: You can only see text based output. Applications that are graphical in nature can only be viewed using the LCD or DVI output of your board if available. If a video output is not available you will not be able to see the application output and should not execute the application.

When you are done exploring the example applications available you can minimize your browser and go on to the next step.

Launching the Software Developer's Guide

The terminal window should now be prompting you to decide whether or not to view the **Software Developer's Guide**.

```
TI provides an up-to-date Software Developer's Guide (SDG) in the form of a wiki on the TI website. Would you like to open this before beginning the setup process? (y/n)? [y]
```

If you select *y* here or just press *Enter* your browser will open a new tab connecting to the **Software Developer's Guide** on the TI wiki. Entering *n* will skip this step. This guide will explain the features of the SDK and how to use it.

When you are done reviewing the **Software Developer's Guide** you can minimize your browser and go to the next step.

Installing the SDK

The terminal window should now be prompting you about whether or not to begin the SDK installation.

```
Would you like to begin the SDK installation? (y/n)? [y]
```

If you select *y* here or just press *Enter* the SDK installer will pop up and prompt you to begin installation. Entering *n* will skip SDK installation and the terminal will close, at which time the START_HERE script has finished running.

If you selected *y* above then you will be guided through the SDK installation using the graphical interface. It is generally best to select all the default values.

NOTE: If you do not install the SDK to the default location you will be prompted to run the setup.sh script inside of the SDK yourself with a message similar to:

```
BeagleBone setup.sh
```

```
The remainder of the BeagleBone setup is performed via setup.sh. Setup.sh has
```

been modified slightly for the BeagleBone so that it can autodetect and install the drivers for the built in USB-to-Serial and XDS100v2.

The setup script is located in your install directory, and requires sudo (admin) access to correctly run. To run type "sudo <sdk-install dir>/setup.sh"
Press Enter to Exit.

Running setup.sh

Assuming that you installed the SDK to the default path the script will find the setup.sh script within the SDK installation and prompt you if you would like to run that script.

```
BeagleBone setup.sh
```

The remainder of the BeagleBone setup is performed via setup.sh. Setup.sh has been modified slightly for the BeagleBone so that it can autodetect and install the drivers for the built in USB-to-Serial and XDS100v2.

```
Would you like to begin setup.sh now? (y/n)  
[y]
```

For details on what the setup.sh script does refer to the [SDK setup script](#) page.

NOTE: You will need to have sudo (administrative) privileges to run the setup.sh script. If you do not have sudo access on your Linux PC you should select *n* here and talk to your system administrator to gain sudo permissions. You can then run setup.sh manually as described on the [SDK setup script](#) page. If you do have sudo permissions you will be prompted for your sudo password.

When setup.sh has finished you will be left with a terminal that is connected over the USB to serial adapter to the board. From this terminal you can interact with the shell command line on the board.

Matrix Users Guide

Return to the [Sitara Linux Software Developer's Guide](#)



Important Note

This guide is for the latest version of Matrix that is included in the SDK starting at version 5.03. If you are looking for information about the old Matrix then this can be found at the following link [Previous Version of Matrix](#)^[1]

Supported Platforms

This version of Matrix supports **all** Sitara devices

Initial Boot Up

When you first boot up a target system containing a Sitara Software Development Kit (SDK), Matrix should be automatically started. Matrix can be either operated by touchscreen or mouse. Default startup for most SDK platforms is touchscreen. Should you encounter any problems below are some tips to get everything running smoothly. See [Matrix Startup Debug](#)

Overview

Matrix is an HTML 5 based application launcher created to highlight available applications and demos provided in new Software Development Kits. There are two forms of Matrix, local and remote Matrix. All of the example applications and demos are available using either the local or remote version. The local version launches by default when the target system is booted and uses the target system's touchscreen interface for user input. Matrix comes as a 4x3 matrix of icons or as a 4x2 matrix depending on the display resolution.



Difference from Previous Version of Matrix

Matrix has been transformed from a QT based C application to a HTML 5 application. Using the latest and greatest technologies including HTML 5, PHP and the Lighttpd web server, it is now even easier to add applications to Matrix and customize it. Matrix now supports a great new feature called Remote Matrix.

Local Matrix

Local Matrix refers to Matrix being displayed on a display device attached to the target system. The launcher for Matrix is just a simple QT application that displays a Webkit base browser that points to the URL `http://localhost:80`.

NOTE: Versions of matrix released before the Sitara Linux SDK 05.04 use port **8080** for Matrix

Remote Matrix

Remote Matrix refers to Matrix being ran in any modern day web browser not located on the target system.

The URL for Remote Matrix is `http://<target system's ip address>`

NOTE: Versions of Matrix released before the Sitara Linux SDK 05.04 use port **8080** for Matrix ie `http://<target system's ip address>:8080`

You can find the target's ip address by using local Matrix and clicking on the Settings icon and then on the Network Settings icon. Or using a terminal logged in to the target system enter the below command:

```
ifconfig
```

From the output displayed, look in the section that starts with eth0. You could see an IP address right after "inet addr". This is the IP address you should use for remote Matrix.

With Remote Matrix you can interact with Matrix on your PC, cellphone, tablet, or any device with a modern web browser. You can now launch text based applications or scripts and have the output streamed back to your web browser! Launching a gui application from Matrix requires you to still look at the display device connected to the target system.

Matrix Project Webpage

The official website for Matrix is located at gforge.ti.com/gf/project/matrix-gui-v2/ ^[2] Any comments or bug reports for Matrix should be posted there.

How to Use the Matrix

Matrix is based on HTML 5 and is designed to be easily customizable. All applications and submenus for Matrix can be found in the target system's `usr/share/matrix-gui-2.0/apps/` directory. Matrix utilizes the `.desktop` standard along with some additional parameters to easily allow modifying, adding and removing an application or directory.

Matrix Components

Below is a summary of all the Matrix web pages:

Menu Pages

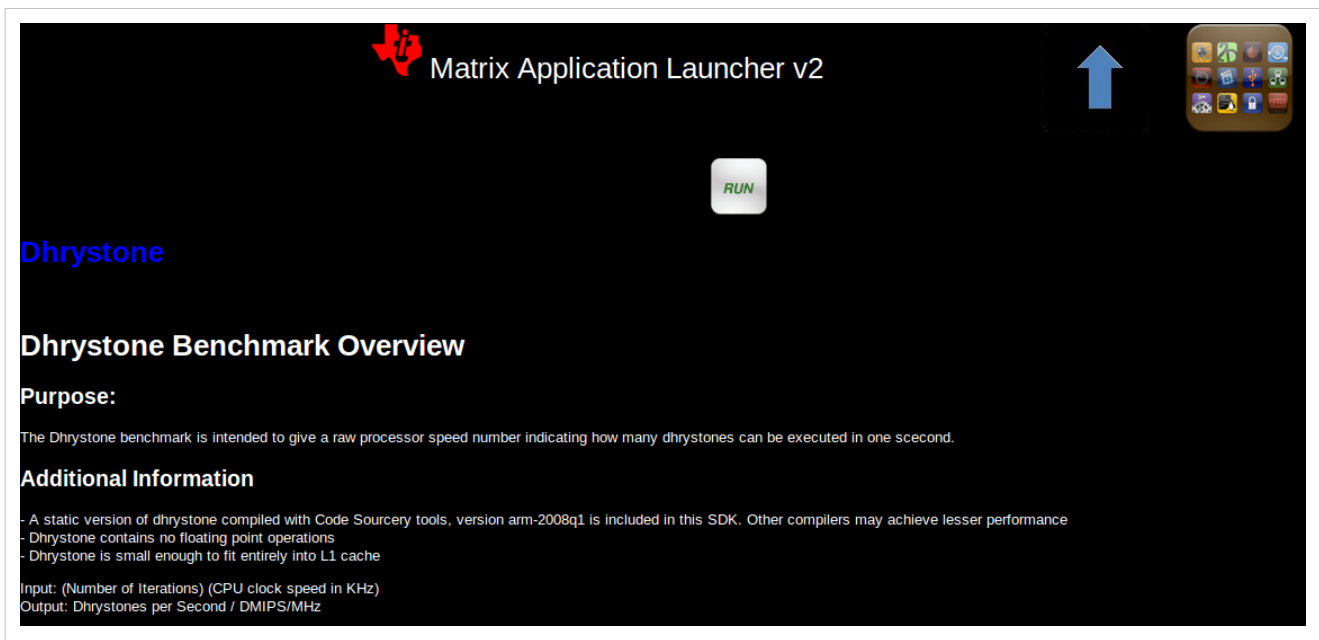
- Contains all the directories or application that belongs to each directory level.

Application Description

- Optional and associated with a particular application.
- Provide additional information which can be useful for various reasons
- Displayed when the associated application icon is pressed.

Example Application Description Page

Below is an example application description page. Description pages can be used to add additional information that may not be obvious.



Coming Soon Page

- Displayed for Matrix directories that doesn't contain any applications within it.

Application/Script Execution Page

- For console based application, displays the output text of the application

Icons

- 96x96 png image files which are associated to a submenu or an application.
- Can be re-used by many applications

Applications

- Any application can be launched by Matrix
- Local Matrix uses the graphics display layer. If a launched application also uses the graphics display layer there will be a conflict.

Updating Matrix

Matrix 2 utilizes a caching system that caches the information read from the .desktop files and also the html that is generated from the various php pages. While this provides a substantial performance boost, developers must be aware that any changes to the Matrix apps folder which includes adding, deleting and modifying files can result in many problems within Matrix. To properly update Matrix with the latest information, Matrix's caches need to be cleared.

Automatically Clearing Matrix Cache

The simplest way to clear Matrix's cache is to use the Refresh Matrix application found within Matrix's Settings submenu. Simply running the application will cause Matrix to clear all the cached files and regenerate the .desktops cache file. Once the application is done running, Matrix will be updated with the latest information found from within the apps folder.

Manually Clearing Matrix Cache

Matrix caching system consists of 1 file and 1 directory. Within Matrix's root directory there contains a file called json.txt. Json.txt is a JSON file that contains information gathered from all the .desktops located within the apps directory. This file is generated by executing the generate.php file.

Also located in Matrix's root directory is a folder called cache. This folder contains all of the html files cached from the various dynamic php webpages.

To clear Matrix's caches you need to perform only two steps:

1. Execute the generate.php file.

In the terminal of the target system, enter the following line of code.

```
php generate.php
```

or

In a browser enter the following url. Note replace <target ip> with the IP address of the target system.

```
http://<target ip>:80/generate.php
```

Viewing generate.php in the browser should display a blank page. There is no visual output to this webpage.

NOTE: Versions of matrix released before the Sitara Linux SDK 05.04 use port **8080** for Matrix

2. You need to clear the files located within Matrix's cache folder. Enter the following commands.

```
cd /usr/share/matrix-gui-2.0/cache
rm -r *
```

Once the above steps are completed, Matrix will be updated.

Launching Matrix

Use the following shell script in the target's terminal window to run Matrix as a background task:

```
/etc/init.d/matrix-gui-2.0 start
```

This script ensures that the touchscreen has been calibrated and that the Qt Window server is running.

Alternatively, Matrix can be launched manually with this full syntax:

```
matrix_browser -qws http://localhost:80
```

NOTE: Versions of matrix released before the Sitara Linux SDK 05.04 use port **8080** for Matrix

The “-qws” parameter is required to start the Qt window server if this is the only/first Qt application running on the system.

The third parameter is the URL that you want the application's web browser to go to. http://localhost:80 points to the web server on the target system that is hosting Matrix.

NOTE: Versions of matrix released before the Sitara Linux SDK 05.04 use port **8080** for Matrix

For the AM37x EVM then the following command must be used:

```
matrix_browser -qws -display transformed:Rot90 http://localhost:80
```

This adds a new rotation parameter which is required due to the attached LCD orientation.

The “-display transformed:Rot90” parameter causes the Qt windowing system to rotate the display 90 degrees.

Matrix Startup Debug

The following topics cover debugging Matrix issue at startup or disabling Matrix at start up.

Touchscreen not working

Please see this wiki page to recalibrate the touch screen: [How to Recalibrate the Touchscreen](#)

Matrix is running but I don't want it running

1. Exit Matrix by going to the Settings submenu and running the Exit Matrix application. Note that exiting Matrix only shuts down local Matrix. Remote Matrix can still be used.
2. Or if the touchscreen is not working, from the console, type:

```
/etc/init.d/matrix-gui-2.0 stop
```


I don't want Matrix to run on boot up

From the console type the following commands:

```
cd /etc/rc5.d
mv S99matrix-gui-2.0 K99matrix-gui-2.0
```

This will cause local Matrix to not automatically start on boot up.

How to Enable Mouse Instead of Touchscreen for the Matrix

You can enable mouse by referring to the following: [How to Enable Mouse for the Matrix GUI](#)

How to Switch Display from LCD to DVI out for the Matrix

You can switch the display output by referring to the following: [How to Switch Display Output for the Matrix GUI](#)

Adding a New Application/Directory to Matrix

Below are step by step instructions.

1. Create a new folder on your target file system at `/usr/share/matrix-gui-2.0/apps/`. The name should be a somewhat descriptive representation of the application or directory. The folder name must be different than any existing folders at that location.
2. Create a `.desktop` file based on the parameters discussed below. It is recommended the name of the desktop file match the name of the newly created folder. No white spaces can be used for the `.desktop` filename. The `.desktop` file parameters should be set depending on if you want to add a new application or a new directory to Matrix. The `Type` field must be set according to your decision. The `.desktop` file must have the `.desktop` suffix.
3. Update the `Icon` field in the `.desktop` to reference any existing Icon in the `/usr/share/matrix-gui-2.0` directory or subdirectories. You can also add a new 96x96 png image and place it into your newly created folder.
4. Optionally for applications you can add a HTML file that contains the application description into your newly created directory. If you add a description page then update the `X-Matrix-Description` field in the `.desktop` file.
5. Refresh Matrix using the application "Refresh Matrix" located in the Settings submenu.

Run your new application from Matrix!

See reference examples below: [Examples](#)

Blank template icons for Matrix can be found here:

gforge.ti.com/gf/download/frsrelease/712/5167/blank_icons_1.1.tar.gz ^[3]

Creating the .Desktop File

The `.desktop` file is based on standard specified at the standards.freedesktop.org/desktop-entry-spec/latest/ ^[4]
Additional fields were added that are unique for Matrix.

Format for each parameter:

`<Field>=<Value>`

The fields and values are case sensitive.

Examples

Creating a New Matrix Directory

You can get all the files including the image discussed below from the following file: [Ex_directory.tar.gz](#)

Create a directory called `ex_directory`

Create a new file named `hello_world_dir.desktop`

Fill the contents of the file with the text shown below:

```
#!/usr/bin/env xdg-open
[Desktop Entry]
Name=Ex Demo
Icon=/usr/share/matrix-gui-2.0/apps/ex_directory/example-icon.png
Type=Directory
X-MATRIX-CategoryTarget=ex_dir
X-MATRIX-DisplayPriority=5
```

This `.desktop` above tells Matrix that this `.desktop` is meant to create a new directory since `Type=Directory`. The directory should be named "Ex Demo" and will use the icon located within the `ex_directory` directory. This new directory should be the 5th icon displayed as long as there aren't any other `.desktop` files that specify `X-MATRIX-DisplayPriority=5` and will be displayed in the Matrix Main Menu. Now any applications that wants to be displayed in this directory should have their `.desktop` `Category` parameter set to `ex_dir`.

- Note that sometimes Linux will rename the `.desktop` file to the name specified in the `Name` field. If this occurs don't worry about trying to force it to use the file name specified.

Now move the `.desktop` file and image into the `ex_directory` directory that was created.

Moving the Newly created Directory to the Target's File System

Open the Linux terminal and go to the directory that contains the `ex_directory`.

Enter the below command to copy `ex_directory` to the `/usr/share/matrix-gui-2.0/apps/` directory located in the target's file system. Depending on the `targetNFS` directory premissions you might have to include `sudo` before the `cp` command.

```
host $ cp ex_directory ~/ti-sdk-XXXX-evm-x.x.x.x/targetNFS/usr/share/matrix-gui-2.0/apps/
```

If NFS isn't being used then you need to copy the `ex_directory` to the `/usr/share/matrix-gui-2.0/apps/` directory in the target's filesystem.

Updating Matrix

Now in either local or remote Matrix go to the Settings directory and click on and then run the Refresh Matrix application. This will delete all the cache files that Matrix generates and regenerates all the needed files which will include any updates that you have made.

Now if you go back to Matrix's Main Menu the 5th icon should be the icon for your Ex Demo.

Creating a New Application

This example is assuming that you completed the [Creating a New Matrix Directory](#) example.

You can get all the files including the image discussed below from the following file: [Ex_application.tar.gz](#)

Create a new directory called `ex_application`

Create a file named `test.desktop`

Fill the contents of the file with the below text:

```
#!/usr/bin/env xdg-open
[Desktop Entry]
Name=Test App
Icon=/usr/share/matrix-gui-2.0/apps/ex_application/example-icon.png
Exec=/usr/share/matrix-gui-2.0/apps/ex_application/test_script.sh
Type=Application
ProgramType=console
Categories=ex_dir
X-Matrix-Description=/usr/share/matrix-gui-2.0/apps/ex_application/app_desc.html
X-Matrix-Lock=test_app_lock
```

`Type=Application` lets Matrix know that this `.desktop` is for an application. The name of the application is "Test App". The icon `example-icon.png` can be found within the `ex_application` directory. The command to execute is a shell script that will be located within `ex_application`. The script that is being ran is a simply shell script that output text to the terminal. Therefore, the `ProgramType` should be set to `console`. This application should be added to the Ex Demo directory from the previous example. Therefore, `Categories` will be set to `ex_dir` which is the same value that `X-MATRIX-CategoryTarget` is set to. You could optionally remove the `Categories` field to have this application displayed in Matrix's Main Menu. This application will also have a description page. The html file to be used is located within the `ex_application` directory. A lock is also being used. Therefore, any other application including itself that has the same lock can't run simultaneously.

Create a file named `test_script.sh`

```
echo "You are now running you first newly created application in Matrix"
echo "I am about to go to sleep for 30 seconds so you can test out the lock feature if you want"
sleep 30
echo "I am finally awake!"
```

The newly created script needs to have its permission set to be executable. Enter the below command to give read, write and execute permission to all users and groups for the script:

```
host $ chmod 777 test_script.sh
```

Create a new file called `app_desc.html`

```
<h1>Test Application Overview</h1>
<h2>Purpose:</h2>
<p>The purpose of this application is to demonstrate the ease in adding a new application to Matrix.</p>
```

Now move the `.desktop` file, script file, the png image located in the `Ex_application.tar.gz` file and the html file into the `ex_application` folder.

Moving the newly created Directory to the Target System

Open the Linux terminal and go to the directory that contains the `ex_application` directory.

Enter the below command to copy the `ex_application` directory to `/usr/share/matrix-gui-2.0/apps/` located in the target's file system. Depending on the targetNFS directory permissions you might have to include `sudo` before the `cp` command.

```
host $ cp ex_application ~/ti-sdk-XXXX-evm-x.x.x.x/targetNFS/usr/share/matrix-gui-2.0/apps/
```

If your not using NFS but instead are using a SD card then copy `ex_application` into the `/usr/share/matrix-gui-2.0/apps/` directory in the target's filesystem.

Updating Matrix

Now in either local or remote Matrix go to the Settings directory and click and then run the Refresh Matrix application. This will delete all the cache files that Matrix generates and regenerate all the needed files which will include any updates that you have made.

Now if you go back to the Matrix's Main Menu and click on the Ex Demo directory you should see your newly created application. Click on the application's icon and you will see the application's description page. Click the Run button and your application will execute. If you try to run two instances of this application simultaneously via local and remote Matrix you will get a message saying that the program can't run because a lock exists. Because of X-Matrix-Lock being set to `test_app_lock`, Matrix knows not to run two instances of a program simultaneously that share the same lock. You can run the application again when the previous application is done running.

You have just successfully added a new application to Matrix using all the possibly parameters!

Matrix Advanced Developers Guide

Coming Soon

Archived Versions

- [Sitara Linux SDK 05.03](#) ^[5]

References

[1] http://processors.wiki.ti.com/index.php?title=Matrix_Users_Guide&oldid=74107

[2] <https://gforge.ti.com/gf/project/matrix-gui-v2/>

[3] https://gforge.ti.com/gf/download/frsrelease/712/5167/blank_icons_1.1.tar.gz

[4] <http://standards.freedesktop.org/desktop-entry-spec/latest/>

[5] http://processors.wiki.ti.com/index.php?title=Matrix_Users_Guide&oldid=97571

Power Management Users Guide

Return to the [Sitara Linux Software Developer's Guide](#)



Overview

This article provides a description of the example applications under the Power page of the Matrix application that comes with the Sitara SDK. This page is labeled "Power" in the top-level Matrix GUI. NOTE: Not all power applications can be supported for AM335x class devices. This is due to differences in the underlying kernel implementation. The location of the Power icon on the main Matrix app list may be different than shown here, depending on screen size.



Supported Platforms

- AM37x
- Beagleboard-xM
- AM335x: power management support is being phased in for this device. Beginning with SDK 05.04.01.00, cpufreq, cpuidle, and Suspend-to-RAM (Deep Sleep 0 state) support will be available. Please refer to http://processors.wiki.ti.com/index.php/AM335x_Power_Management_User_guide for full details. Future releases will add support for more aggressive suspend modes (RTC), and OPP 50 support.

PLEASE NOTE: cpufreq may cause I2C lockups on AM335x EVM boards. Beaglebone is not affected. This is a known issue related to the CPLD firmware. If the CPLD firmware on your EVM is detected to be the wrong version, the Matrix application output will inform you of the version mismatch and continue.

Once updated CPLD firmware is available, this documentation will be updated to teach users how to upgrade their CPLD if necessary/desired. This procedure will require an Altera programming pod.

Power Examples

Several power examples exist to provide users the ability to dynamically switch the CPU clock frequency. The frequencies shown are those available for your system. Upon making a selection, you will be presented a confirmation page. The readout number "BogoMIPS" will confirm the new clock frequency. Please note that the frequency will read out with a slight margin compare to the intended frequency. For example, if you select 1GHz, you may see a number like 998.84 (in MHz). This is normal. After reviewing the confirmation page, press the Close button to return to normal Matrix operation.

Other power examples are provided which may be useful for power management developers and power users. These have been included in Matrix in part to make users aware that these valuable debugging tools exist, in addition to the convenience of executing each application from the GUI. In depth descriptions for each application follow. Similar descriptions are also given via description pages in Matrix, which will be displayed when clicking the button. Where appropriate, the documentation will point out the corresponding command line operation.

The Suspend/Resume button demonstrates the ability to put the machine into a suspended state. See below for complete documentation of this feature.

Please note that the order of applications which appear on your screen may differ from the picture below, due to devices with different screen sizes, and differences between Matrix v1 and v2.

Further note that AM335x supports a smaller number of power examples than on AM37x. Screen capture below is from AM37x.



Set ___ MHz

```
(command line equivalent)
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
(view options, select one for next step)
echo <selected frequency, in KHz> > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
cat /proc/cpuinfo
```

The cpufreq kernel interface allows the user to select the OPP (operating point / clock frequency selection). Cpubfreq provides an opportunity to save power by adjusting/scaling voltage and frequency based on the current cpu load.

Power Statistics and Active Clocks (AM37x only)

```
(command line equivalent)
mount -t debugfs debugfs /debug
cat /debug/pm_debug/count
```

The first set of information that is displayed in the dump gives the state statistics for each power domain. Each power domain state is shown with an associated off-mode counter in parenthesis. The RET_LOGIC_OFF counter refers to the logic off counter, and the RET_MEMBANKN_OFF counter refers to the per-bank memory off counter.

Following that, each of the clock domain active clocks is displayed along with the associated use count. The use count refers to the number of enabled downstream clocks from this clock domain. Once the count reaches 0, the clock domain can be put to sleep.

PRCM Register Dump (AM37x only)

```
(command line equivalent)
mount -t debugfs debugfs /debug
cat /debug/pm_debug/registers/current
```

This program allows the user a simple method to quickly display the current state of the the PRCM registers via an exhaustive dump.

PRCM Snapshot (pre-suspend) (AM37x only)

```
(command line equivalent)
mount -t debugfs debugfs /debug
cat /debug/pm_debug/registers/1
```

This program allows the user a simple method to quickly display a snapshot of the state of the PRCM registers that was taken just prior to suspend. The snapshot is taken before the jump occurs into SRAM idle code. Used to debug suspend/resume problems.

PRCM Snapshot (post-resume) (AM37x only)

```
(command line equivalent)
mount -t debugfs debugfs /debug
cat /debug/pm_debug/registers/2
```

This program allows the user a simple method to quickly display a snapshot of the state of the PRCM registers that was taken just after resume. Used to debug suspend/resume problems.

Suspend/Resume

```
(command line equivalent)
mkdir /debug
mount -t debugfs debugfs /debug
echo 1 > /debug/pm_debug/sleep_while_idle (am37x only)
echo 1 > /debug/pm_debug/enable_off_mode (am37x only)
echo mem > /sys/power/state
```

This program will put the platform into suspend mode.

Sleep_while_idle enables the kernel (for AM37x only) to attempt to reach the retention or off state while idle. Enable_off_mode enables the kernel (for AM37x only) to transition to off mode (deeper sleep state than retention). The final command initiates the suspend.

IMPORTANT NOTE: When running this from Matrix, the system will only properly resume if the user sends a keypress to the UART. If the user presses the touchscreen or a button on the EVM, resume will not complete normally. This issue will be fixed in a future release. If you run these commands from the terminal - all of the normal wakeup events (UART keypress, touchscreen press, EVM keypad press) will operate correctly.

SmartReflex (AM37x and AM335x)

SmartReflex is an active power management technique which optimizes voltage based on silicon process ("hot" vs. "cold" silicon), temperature, and silicon degradation effects. In most cases, SmartReflex provides significant power savings by lowering operating voltage.

On AM335x, SmartReflex is enabled by default in Sitara SDK releases since 05.05.00.00. The SmartReflex driver requires the use of either the TPS65217 or TPS65910 PMIC. Furthermore, SmartReflex is currently supported only on the ZCZ package. Please note that SmartReflex may not operate on AM335x sample devices which were not programmed with voltage targets. To disable SmartReflex, type the following commands at the target terminal:

```
mkdir /debug
mount -t debugfs debugfs /debug
cd /debug/smartreflex ==> NOTE: You may not see 'smartreflex' node
if you have early silicon. In this case SmartReflex operation is not
possible.
echo 0 > autocomp
(Performing "echo 1 > autocomp" will re-enable SmartReflex)
```

On AM37x, SmartReflex is generally disabled by default. To enable SmartReflex, type the following commands at the target terminal:

```
mkdir /debug
mount -t debugfs debugfs /debug
cd /debug/voltage/vdd_core/smartreflex
```



```
echo 1 > autocomp
cd /debug/voltage/vdd_mpu/smartreflex
echo 1 > autocomp
(Note the "echo 0 > autocomp" would be used to disable SmartReflex)
```

On AM37x, to compile SmartReflex support out of the kernel, follow this procedure to modify the kernel configuration:

```
cd <kernel source directory>
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- menuconfig
<the menuconfig interface should appear>
Select "System Type"
Select "TI OMAP Common Features"
Deselect "SmartReflex Support"
Select "Exit" until you are prompted to save the configuration changes, and save them.
Rebuild the kernel.
```

On AM335x, to compile SmartReflex support out of the kernel, follow this procedure to modify the kernel configuration:

```
cd <kernel source directory>
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- menuconfig
<the menuconfig interface should appear>
Select "System Type"
Select "TI OMAP Common Features"
Deselect "AM33xx SmartReflex Support"
Select "Exit" until you are prompted to save the configuration changes, and save them.
Rebuild the kernel.
```

Power Management Reference

- [**AM335x Power Management User guide**](#)

Refer to this page for more information about cpufreq, cpuidle, and Suspend/Resume on AM335x devices.

- [**PSP Power Management User Guide**](#)

(AM/DM37x) Refer to this page for discussion of cpuidle framework architecture, C-states, Dynamic Tick Suppression, Suspend/Resume, Linux kernel configuration for PM, idle state transition, cpufreq, and SmartReflex.

- [**Power Management Performance Guide**](#)

(AM/DM37x) This page builds upon the previous page, and adds VDD1 measurements performed against certain identified scenarios.

- [**AM/DM37x Low Power Standby Support**](#)

Discusses how to reach the lowest possible power, including measured results on AM37X for comparison purposes.

- [**AM/DM37x Power Estimation Spreadsheet**](#)

Advanced information on power consumption modeling, featuring a downloadable spreadsheet.

- [**AM18x: Changing the Operating Point \(OPP\)**](#)

On AM18x devices (aka OMAPL1), changing the OPP is done in a different way, documented here.

ARM Multimedia Users Guide

Return to the [Sitara Linux Software Developer's Guide](#)



Overview

Multimedia codecs on ARM based platforms could be optimised for better performance using the tightly coupled **Neon** co-processor. Neon architecture works with its own independent pipeline and register file. Neon technology is a 128 bit SIMD architecture extension for ARM Cortex-A series processors. It is designed to provide acceleration for multimedia applications.

Supported Platforms

- AM37x
- Beagleboard-xM
- AM35x
- AM335x EVM

Multimedia on Cortex-A8

Cortex-A8 Features and Benefits

- Support ARM v7 with Advanced SIMD (NEON)
- Support hierarchical cache memory
- Up to 1 MB L2 cache
- Up to 128-bit memory bandwidth
- 13-stage pipeline and enhanced branch prediction engine
- Dual-issue of instructions

Neon Features and Benefits

- Independent HW block to support advanced SIMD instructions
 - Comprehensive instruction set with support of 8, 16 & 32-bit signed & unsigned data types
 - 256 byte register file (dual 32x64/16x128 view) with hybrid 32/64/128 bit modes
 - Large register files enables efficient data handling and minimizes access to memory, thus enhancing data throughput
 - Processor can sleep sooner which leads to an overall dynamic power saving
 - Independent 10-stage pipeline
 - Dual-issue of limited instruction pairs
 - Significant code size reduction
-

Neon support on opensource community

NEON is currently supported in the following Open Source projects.

- ffmpeg/libav
 - LGPL media player used in many Linux distros
 - NEON Video: MPEG-4 ASP, H.264 (AVC), VC-1, VP3, Theora
 - NEON Audio: AAC, Vorbis, WMA
- x264 –Google Summer Of Code 2009
 - GPL H.264 encoder –e.g. for video conferencing
- Bluez –official Linux Bluetooth protocol stack
 - NEON sbc audio encoder
- Pixman (part of cairo 2D graphics library)
 - Compositing/alpha blending
 - X.Org, Mozilla Firefox, fennec, & Webkit browsers
 - e.g. fbCompositeSolidMask_nx8x0565neon 8xfaster using NEON
- Ubuntu 09.04 & 09.10 –fully supports NEON
 - NEON versions of critical shared-libraries
- Android –NEON optimizations
 - Skia library, S32A_D565_Opaque 5xfaster using NEON
 - Available in Google Skia tree from 03-Aug-2009

For additional details, please refer the [NEON - ARM website](#) ^[1].

SDK Example Applications

This application can be executed by selecting the "Multimedia" icon at the top-level matrix.

NOTE

The very first GStreamer launch takes some time to initialize outputs or set up decoders.



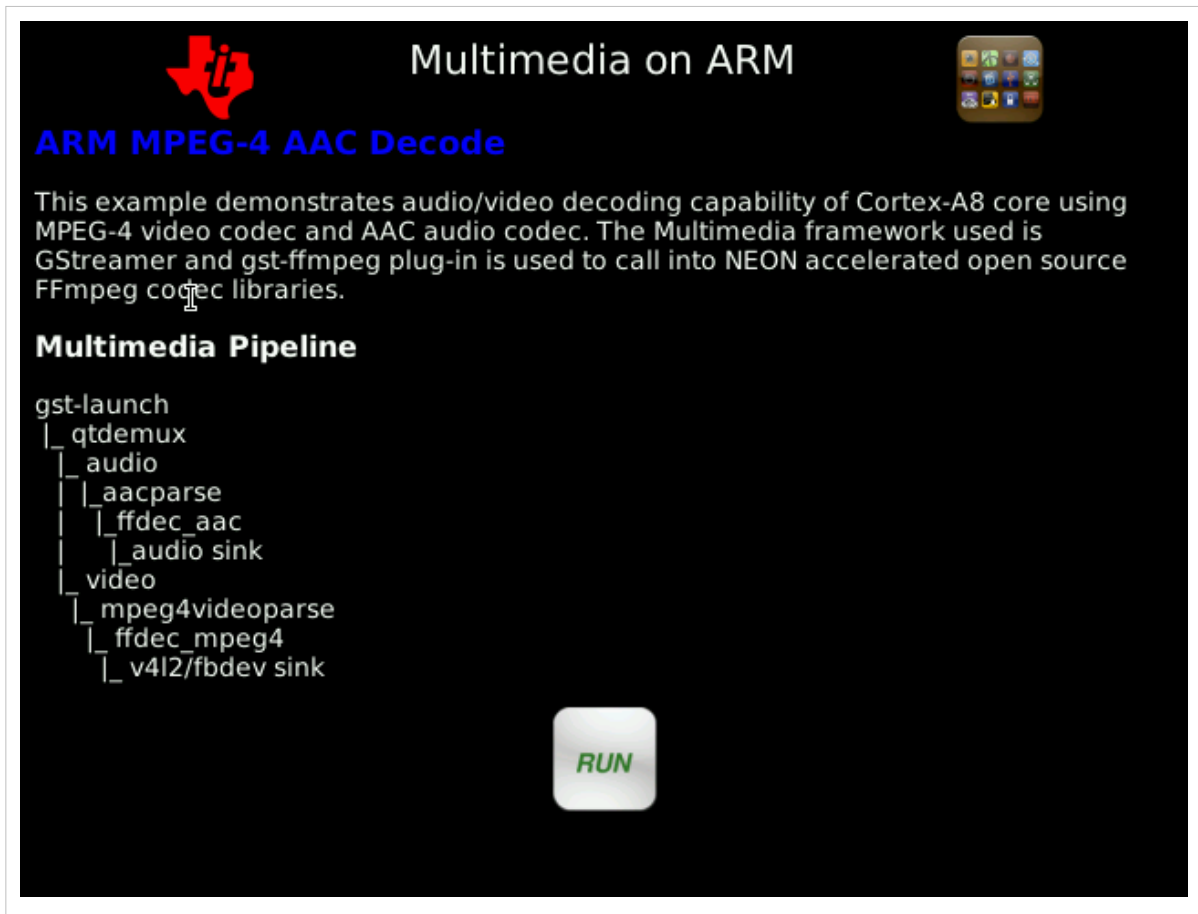
Codec portfolio

AM SDK includes ARM based multimedia using opensource GPLv2+ FFmpeg/Libav codecs, the codec portfolio includes MPEG-4, H.264 for video in VGA/WQVGA/480p resolution and AAC codec for audio.

The script file to launch multimedia demo detects the display enabled and accordingly decodes VGA or 480p video. In AM37x platform VGA clip is decoded when LCD is enabled and 480p is decoded when DVI out is enabled. Scripts in "Settings" menu can be used to switch between these two displays.

MPEG4 + AAC Decode

MPEG-4 + AAC Dec example application demonstrates use of MPEG-4 video and AAC audio codec as mentioned in the description page below.



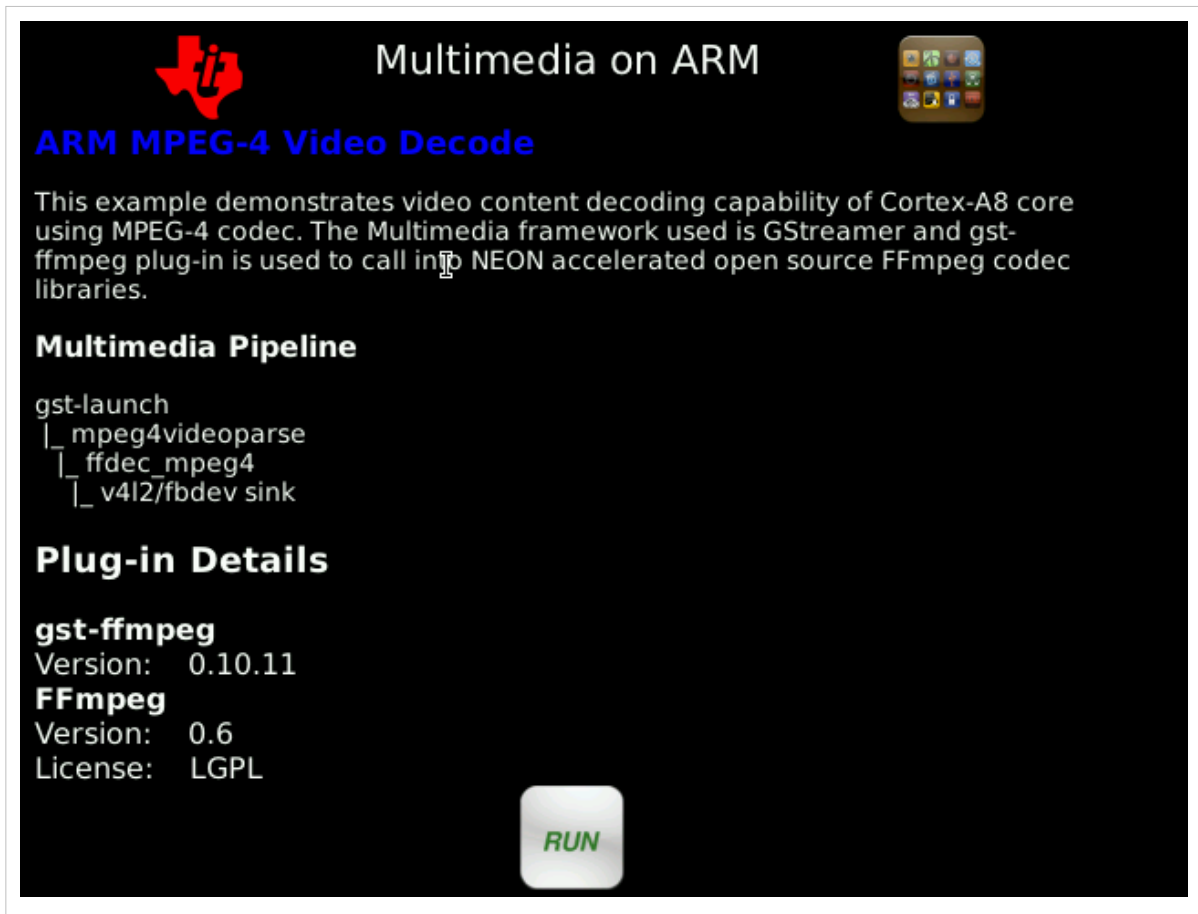
The multimedia pipeline is constructed using `gst-launch`, GStreamer elements such as `qtdemux` is used for demuxing AV content. Parsers are elements with single source pad and can be used to cut streams into buffers, they do not modify the data otherwise.

```
gst-launch-0.10 filesrc location=$filename ! qtdemux name=demux
demux.audio_00 ! queue ! ffdec_aac ! alsasink sync=false demux.video_00
! queue ! ffdec_mpeg4 ! ffmpegcolorspace ! fbdevsink device=/dev/fb0
```

"filename" is defined based on the selected display device which could be LCD or DVI.

MPEG4 Decode

MPEG-4 decode example application demonstrates use of MPEG-4 video codec as mentioned in the description page below.



The screenshot shows a terminal window titled "Multimedia on ARM" with a Texas Instruments logo in the top left and a grid of application icons in the top right. The main content is as follows:

ARM MPEG-4 Video Decode

This example demonstrates video content decoding capability of Cortex-A8 core using MPEG-4 codec. The Multimedia framework used is GStreamer and gst-ffmpeg plug-in is used to call into NEON accelerated open source FFmpeg codec libraries.

Multimedia Pipeline

```
gst-launch
|_ mpeg4videoparse
|_ ffdec_mpeg4
|_ v4l2/fbdev sink
```

Plug-in Details

gst-ffmpeg
Version: 0.10.11

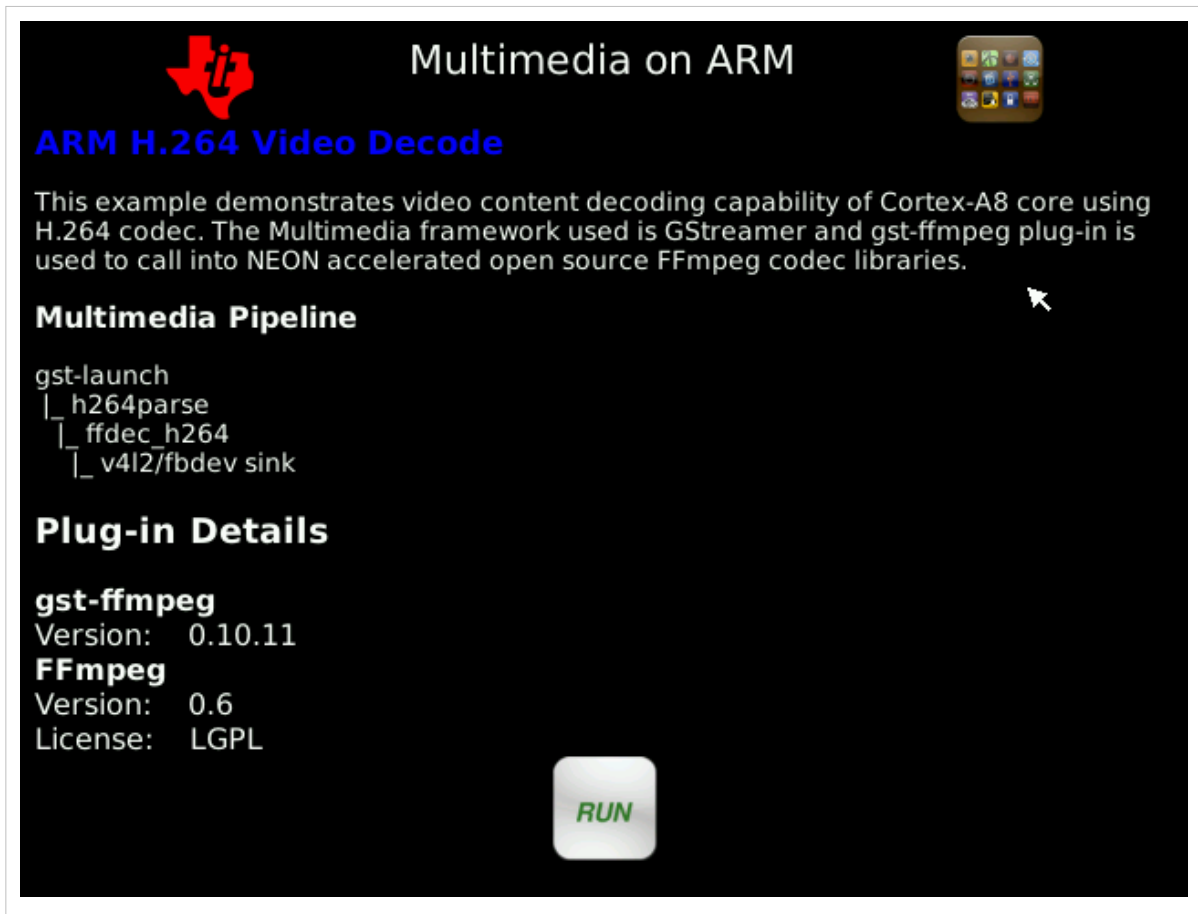
FFmpeg
Version: 0.6
License: LGPL

At the bottom center of the terminal window is a white rounded square button with the word "RUN" in green capital letters.

```
gst-launch-0.10 filesrc location=$filename ! mpeg4videoparse ! ffdec_mpeg4 ! ffmpegcolorspace ! fbdevsink device=/dev/fb0
```

H.264 Decode

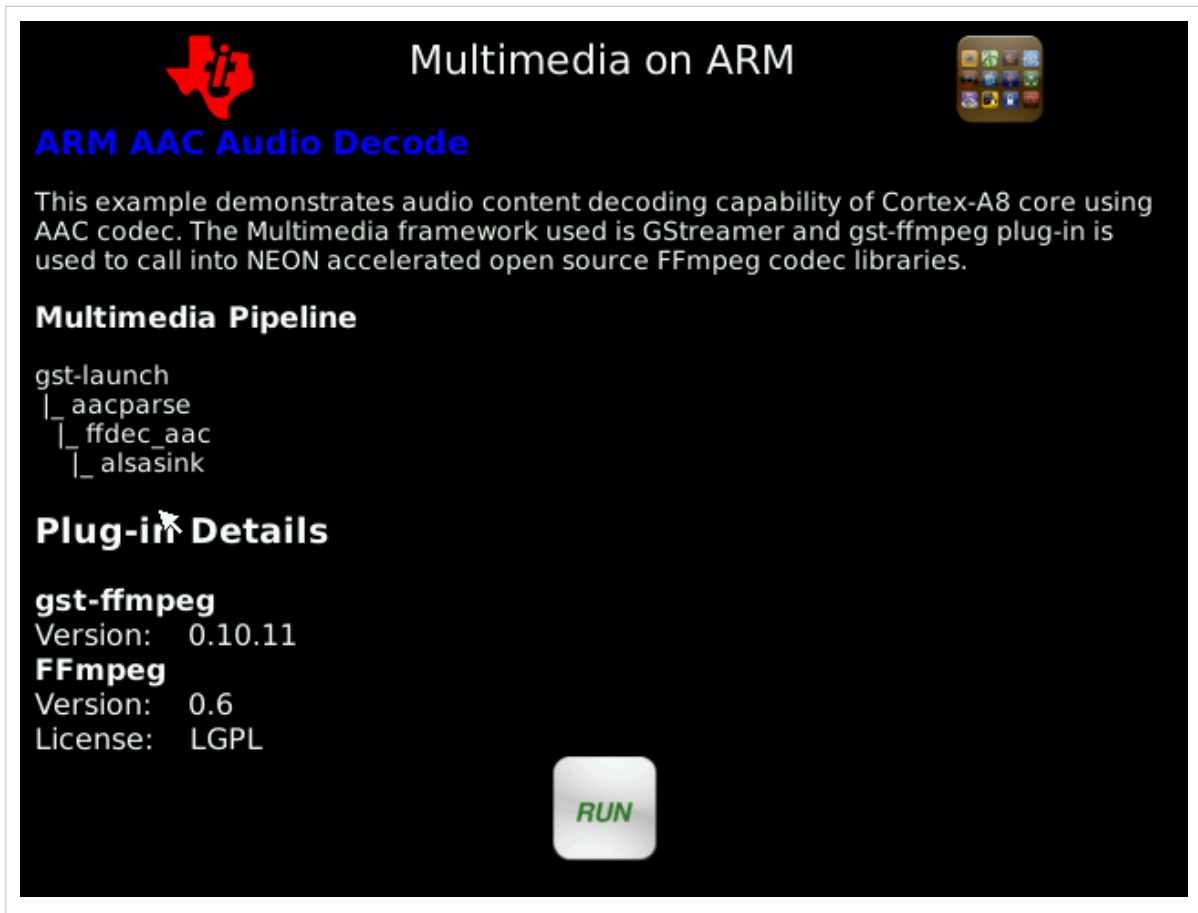
H.264 decode example application demonstrates use of H.264 video codec as mentioned in the description page below.



```
gst-launch-0.10 filesrc location=$filename ! h264parse ! ffdec_h264 ! ffmpegcolorspace ! fbdevsink device=/dev/fb0
```

AAC Decode

AAC decode example application demonstrates use of AAC video codec as mentioned in the description page below.



```
gst-launch-0.10 filesrc location=$filename ! aacparse ! faad ! alsasink
```

Streaming

Audio/Video data can be streamed from a server using souphttpsrc. For example to stream audio content, if you set-up an apache server on your host machine you can stream the audio file HistoryOfTI.aac located in the files directory using the pipeline

```
gst-launch souphttpsrc location=http://<ip address>/files/HistoryOfTI.aac ! aacparse ! faad ! alsasink
```

Multimedia Peripheral Examples

Examples of how to use several different multimedia peripherals can be found on the ARM Multimedia Peripheral Examples page.

MPEG4 + AAC WQVGA Clip: HistoryOfTIAV-WQVGA.mp4 Video: mpeg4, yuv420p, 480x272, 1326 kb/s, 24 fps Audio: aac, 48000 Hz, stereo, s16, 69 kb/s 480p Clip: HistoryOfTIAV-480p.mp4 Video: mpeg4, yuv420p, 720x405, 1778 kb/s, 24 fps Audio: aac, 48000 Hz, stereo, s16, 79 kb/s WVGA Clip: HistoryOfTIAV-WVGA.mp4 Video: mpeg4, yuv420p, 800x450, 1326 kb/s, 24 fps Audio: aac, 48000 Hz, stereo, s16, 69 kb/s	720M	23	44	86	91	22	41	73	80
	600M		52	98	97		49	88	87
	500M		58	97	97		61	97	96
	275M		95	NA	NA		96	NA	NA
MPEG4 WQVGA Clip: HistoryOfTI-WQVGA.m4v Video: mpeg4, yuv420p, 480x272, 24 fps Audio: NA 480p Clip: HistoryOfTI-480p.m4v Video: mpeg4, yuv420p, 720x405, 28 fps Audio: NA WVGA Clip: HistoryOfTI-WVGA.m4v Video: mpeg4, yuv420p, 800x450, 24 fps Audio: NA	720M	16	42	88	71	15	42	76	60
	600M		45	98	78		46	88	69
	500M		55	97	88		53	97	79
	275M		88	96	95		87	96	96
H.264 WQVGA Clip: HistoryOfTI-WQVGA.264 Video: h264, yuv420p, 480x272, 29.97 fps Audio: NA 480p Clip: HistoryOfTI-480p.264 Video: h264, yuv420p, 720x406, 29.97 fps Audio: NA WVGA Clip: HistoryOfTI-WVGA.264 Video: h264, yuv420p, 800x450, 29.97 fps Audio: NA	720M	16	62	98	98	16	63	98	98
	600M		73	98	97		70	98	98
	500M		81	97	97		79	98	97
	275M		95	96	92		96	96	93
AAC Clip: HistoryOfTL.aac Video: NA Audio: aac, 48000 Hz, stereo, s16, 76 kb/s	720M	10	9	Same as VGA	Same as VGA	10	2	Same as VGA	Same as VGA
	600M		2				4		
	500M		3				2		
	275M		26				25		

oProfile

OProfile is a common profiling tool used on Linux platforms, a kernel driver and a user daemon is used to collect samples of data. This tool can be used to profile a huge variety of system-level statistics. Please refer [OProfile](#)^[2] for additional details.

Listed below are the profiling results of Mpeg4 + AAC in VGA resolution at 300MHz and 1GHz via NFS server. Profiling results are obtained from the serial terminal.

In order to profile the multimedia application kernel has to be compiled with oprofile enabled. Followed by this, opcontrol has to be enabled for the first time

```
root@am37x-evm:~# opcontrol --vmlinux=/boot/vmlinux-`uname -r`
```

Setopp power/clocks application can be used to scale the clock settings, the example below shows the settings for opp2, similarly clock can be set to 1GHz using the setopp1.sh application.

```
root@am37x-evm:~# setopp2.sh
Set to operating point 2: CPU = 300 MHz
-----
Processor      : ARMv7 Processor rev 2 (v7l)
BogoMIPS      : 299.95
Features      : swp half thumb fastmult vfp edsp thumbee neon vfpv3
CPU implementer : 0x41
CPU architecture: 7
CPU variant   : 0x3
CPU part      : 0xc08
CPU revision  : 2

Hardware      : OMAP3 EVM
Revision     : 0020
Serial       : 0000000000000000
```

profiling can be calibrated by executing the following set of applications in the given sequence

```
root@am37x-evm:~# opcontrol --reset
root@am37x-evm:~# opcontrol --start
root@am37x-evm:~# runMpeg4AacDec.sh
root@am37x-evm:~# opcontrol --stop
root@am37x-evm:~# oprofile -l
root@am37x-evm:~# oprofile
```

opcontrol is used to initiate the profiling and oprofile to generate the profiling results

300MHz	1GHz
3732 53.7675 vmlinux-2.6.37	10157 89.4653 vmlinux-2.6.37
1291 18.5996	415 3.6554
libgstffmpegcsp.so	libgstffmpegcsp.so
831 11.9723 libavcodec.so.52.113.2	253 2.2285 libavcodec.so.52.113.2
479 6.9010 libfaad.so.2.0.0	241 2.1228 libc-2.9.so
273 3.9332 libc-2.9.so	142 1.2508 libfaad.so.2.0.0
72 1.0373 libm-2.9.so	22 0.1938
69 0.9941 libgstreamer-0.10.so.0.26.0	libgstreamer-0.10.so.0.26.0
38 0.5475 libglib-2.0.so.0.2400.1	18 0.1585 libm-2.9.so
37 0.5331 ld-2.9.so	17 0.1497 ld-2.9.so
33 0.4754 libgobject-2.0.so.0.2400.1	15 0.1321 libgobject-2.0.so.0.2400.1
16 0.2305 libgstbase-0.10.so.0.26.0	14 0.1233 libgstbase-0.10.so.0.26.0
12 0.1729 libpthread-2.9.so	13 0.1145 libglib-2.0.so.0.2400.1
12 0.1729 libgstaudio-0.10.so.0.21.0	10 0.0881 libpthread-2.9.so
8 0.1153 libgstqtdemux.so	6 0.0528 libgstffmpeg.so
8 0.1153 libasound.so.2.0.0	4 0.0352 busybox
6 0.0864 busybox	4 0.0352 libgstqtdemux.so
5 0.0720 libgstfbdevsink.so	4 0.0352 libgstaudio-0.10.so.0.21.0
5 0.0720 libgthread-2.0.so.0.2400.1	3 0.0264 libgstcoreelements.so
4 0.0576 libgstalsa.so	3 0.0264 libasound.so.2.0.0
4 0.0576 libavutil.so.50.39.0	3 0.0264 libavutil.so.50.39.0
2 0.0288 oprofiled	3 0.0264 libgthread-2.0.so.0.2400.1
2 0.0288 libgstcoreelements.so	2 0.0176 oprofiled
	2 0.0176 libgstfaad.so
	2 0.0176 libgstfbdevsink.so
1 0.0144 gst-launch-0.10	
1 0.0144 libgstfaad.so	

Power Benchmark

The power measurements below are analyzed on the AM3730 ES1.2 processor on a Rev.G. EVM. Here we indicate the power measurement after first boot, in suspend mode and with additional power features enabled.

The jumper settings for power measurements are:

J6	VDD_MPU_IVA
J5	VDD_CORE
J28	VDDS, VDDS_MEM, VDDS_WKUP_BG, VDDS_SRAM, POP (DDR and Flash)
J20	VDDS_DPLL_DLL, VDDS_DPLL_PER

Suspend Mode

The Suspend/Resume application in Power/Clocks SDK example application drives the system in suspend mode. The measurements below indicate total power consumption in suspend mode.

Jumper	Pin No	Register Value	Voltage Drop [mV]	Voltage [V]	Current [mA]	Power [mW]
J6	2	0.05	1.00	1.425	20.00	28.50
J5	2	0.1	1.00	1.241	10.00	12.41
J28	2	0.1	0.00	1.808	0.00	0.00
J20	1	0.1	1.00	1.803	10.00	18.03
Total						58.94

After Resume

After the system is resumed from sleep mode, when sleep_while_idle and enable_off_mode features are enabled.

Jumper	Pin No	Register Value	Voltage Drop [mV]	Voltage [V]	Current [mA]	Power [mW]
J6	2	0.05	14.00	1.376	280.00	385.28
J5	2	0.1	11.00	1.317	110.00	144.87
J28	2	0.1	3.00	1.803	30.00	54.09
J20	1	0.1	2.00	1.795	20.00	35.90
Total						620.14

MPEG4 Decode

After the system resume with enhanced power save features enabled, the total power consumption for MPEG4 decode is as shown below.

Jumper	Pin No	Register Value	Voltage Drop [mV]	Voltage [V]	Current [mA]	Power [mW]
J6	2	0.05	17.00	1.349	340.00	458.66
J5	2	0.1	12.00	1.316	120.00	157.92
J28	2	0.1	4.00	1.800	40.00	72.00
J20	1	0.1	2.00	1.795	20.00	35.90
Total						724.48

MPEG4 Decode with Scaling Governor Enabled

These power measurements can be further optimized by setting the scaling governor feature to 'ondemand'. Here the power values are dynamically scaled at run time depending on the system level requirements. It can be enabled using sysfs, the resultant improvement in power values are as shown below.

```
echo "ondemand" > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

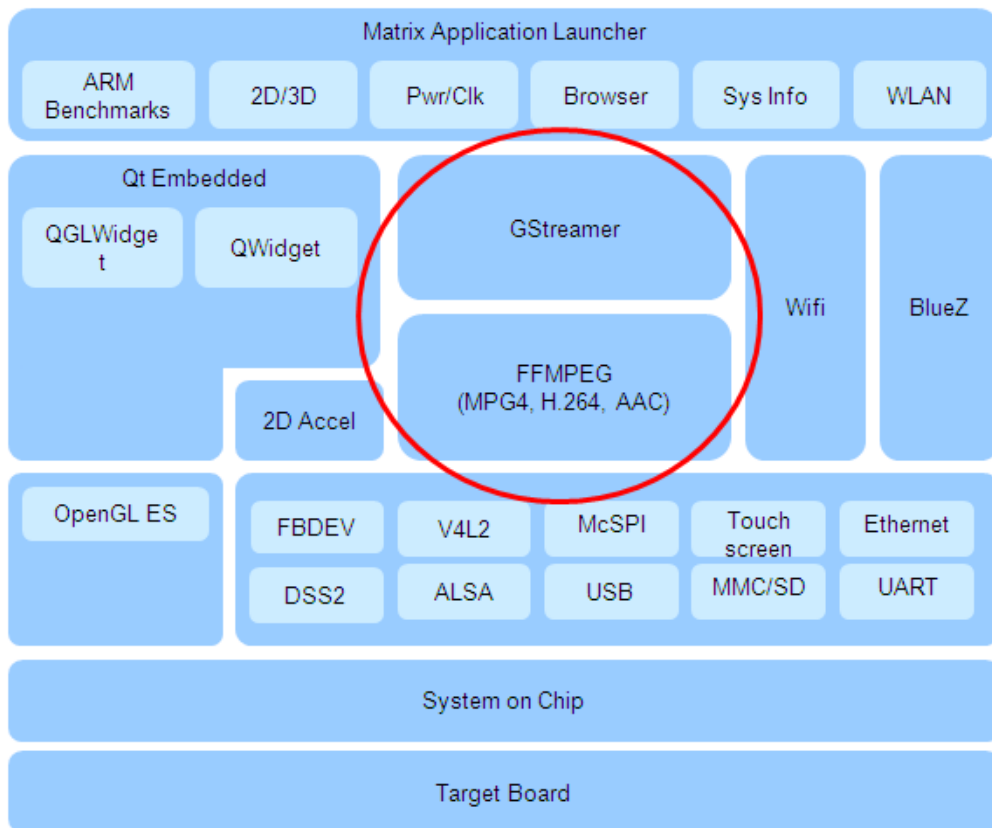
Jumper	Pin No	Register Value	Voltage Drop [mV]	Voltage [V]	Current [mA]	Power [mW]
J6	2	0.05	10.75	1.376	215.00	295.84
J5	2	0.1	12.00	1.317	120.00	158.04
J28	2	0.1	5.00	1.803	50.00	90.15
J20	1	0.1	1.00	1.795	10.00	17.95
Total						561.98

Further power optimization

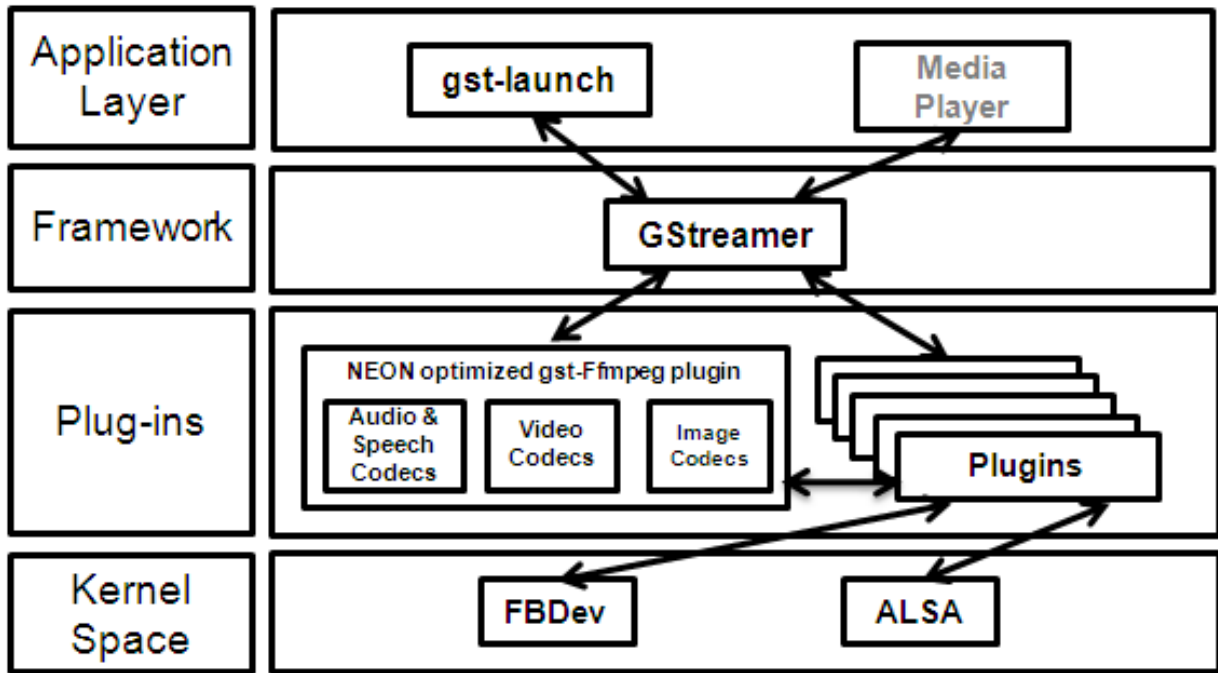
The system level power consumption can be further optimized by disabling clocks of unused modules and enabling Smart Reflex. Additional details of power optimization can be obtained from **OMAP power management guide** ^[3] and **PSP user guide for 2.6.37 kernel**

SDK Multimedia Framework

Multimedia framework for cortex-a8 SDK will leverage GStreamer multimedia stack with gst-ffmpeg plug-in's to support GPLv2+ FFmpeg/libav library code.



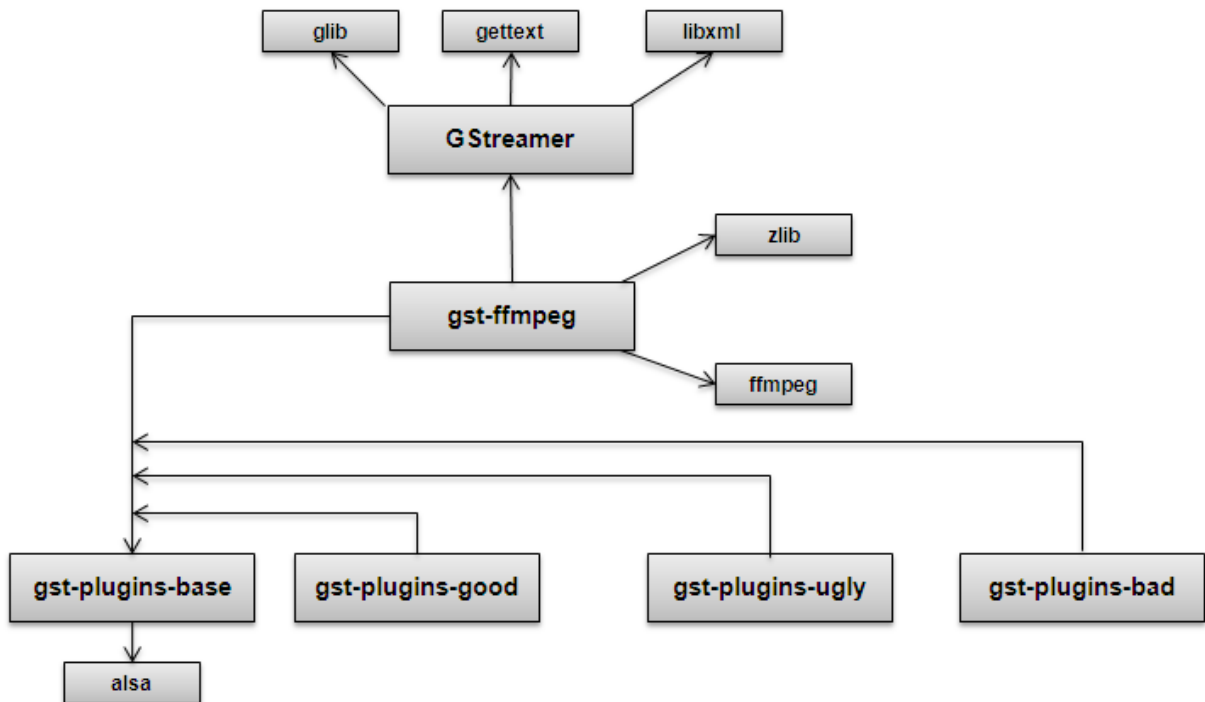
gst-launch is used to build and run basic multimedia pipelines to demonstrate audio/avideo decoding examples.



GStreamer

- Multimedia processing library
- Provides uniform framework across platforms
- Includes parsing & A/V sync support
- Modular with flexibility to add new functionality via plugins
- Easy bindings to other frameworks

Some of the build dependencies for GStreamer are shown here:



Open Source FFmpeg Codecs

FFmpeg^[4] is an open source project which provides a cross platform multimedia solution.

- Free audio and video decoder/encoder code licensed under GPLv2+ (GPLv3 licensed codecs can be build separately)
- A comprehensive suite of standard compliant multimedia codecs
 - Audio
 - Video
 - Image
 - Speech
- Codec software package
- Codec libraries with standard C based API
- Audio/Video parsers that support popular multimedia content
- Use of SIMD/NEON instructions **cortex-A8 neon architecture**
- Neon provides 1.6x-2.5x performance on complex video codecs

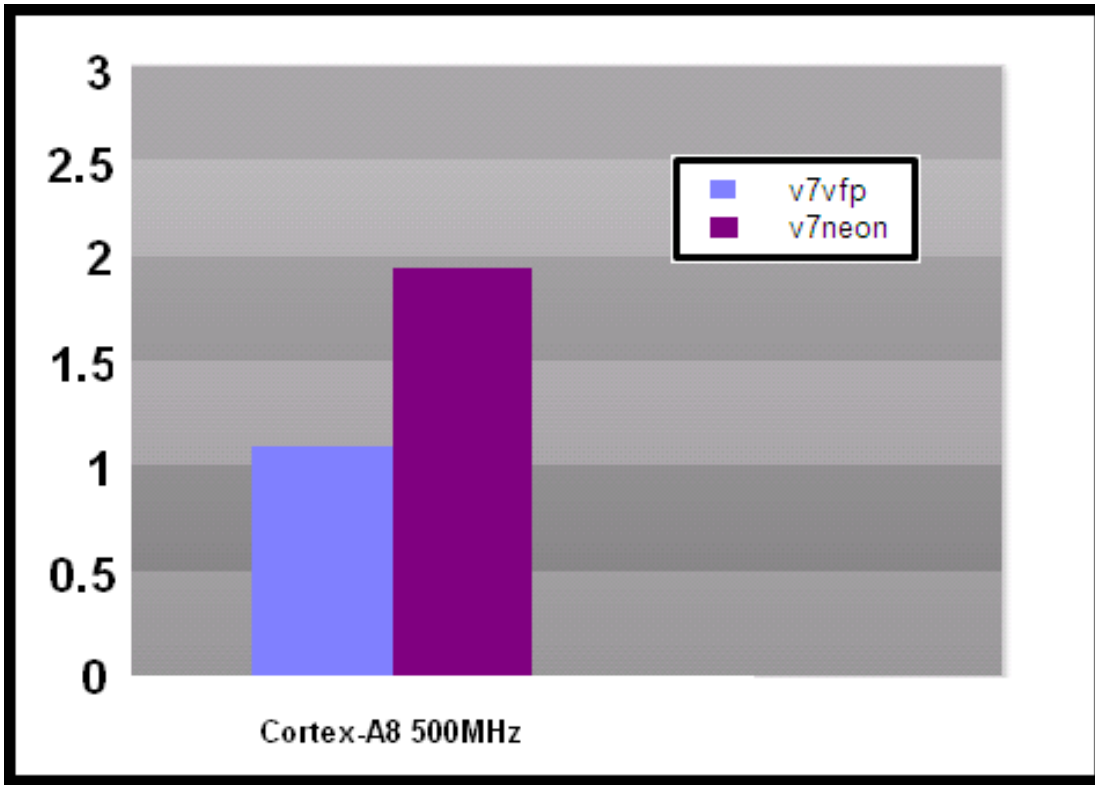
Multimedia Neon Benchmark

Test Parameters:

- Sep 21 2009 snapshot of gst-ffmpeg.org
- Real silicon measurements on Omap3 Beagleboard

Resolution	480x270
Frame Rate	30fps
Audio	44.1KHz
Video Codec	H.264
Audio Codec	AAC

- Benchmarks released by ARM demonstrating an overall performance improvement of ~2x.



FFmpeg Codecs List

FFmpeg Codec Licensing

FFmpeg libraries include LGPL, GPLv2, GPLv3 and other license based codecs, enabling GPLv3 codecs subjects the entire framework to GPLv3 license. In the Sitara SDK GPLv2+ licensed codecs are enabled. Enabling Additional details of **legal and license** ^[5] of these codecs can be found on the FFmpeg/libav webpage.

GPLv2+ codecs list

Codec	Description
ffenc_a64multi	FFmpeg Multicolor charset for Commodore 64 encoder
ffenc_a64multi5	FFmpeg Multicolor charset for Commodore 64, extended with 5th color (colram) encoder
ffenc_asv1	FFmpeg ASUS V1 encoder
ffenc_asv2	FFmpeg ASUS V2 encoder
ffenc_bmp	FFmpeg BMP image encoder
ffenc_dnxhd	FFmpeg VC3/DNxHD encoder
ffenc_dvvideo	FFmpeg DV (Digital Video) encoder
ffenc_ffv1	FFmpeg FFmpeg video codec #1 encoder
ffenc_ffvhuff	FFmpeg Huffiyuv FFmpeg variant encoder
ffenc_flashsv	FFmpeg Flash Screen Video encoder
ffenc_flv	FFmpeg Flash Video (FLV) / Sorenson Spark / Sorenson H.263 encoder
ffenc_h261	FFmpeg H.261 encoder
ffenc_h263	FFmpeg H.263 / H.263-1996 encoder
ffenc_h263p	FFmpeg H.263+ / H.263-1998 / H.263 version 2 encoder

ffenc_huffyuv	FFmpeg Huffiyuv / HuffYUV encoder
ffenc_jpegls	FFmpeg JPEG-LS encoder
ffenc_ljpeg	FFmpeg Lossless JPEG encoder
ffenc_mjpeg	FFmpeg MJPEG (Motion JPEG) encoder
ffenc_mpeg1video	FFmpeg MPEG-1 video encoder
ffenc_mpeg4	FFmpeg MPEG-4 part 2 encoder
ffenc_msmpeg4v1	FFmpeg MPEG-4 part 2 Microsoft variant version 1 encoder
ffenc_msmpeg4v2	FFmpeg MPEG-4 part 2 Microsoft variant version 2 encoder
ffenc_msmpeg4	FFmpeg MPEG-4 part 2 Microsoft variant version 3 encoder
ffenc_pam	FFmpeg PAM (Portable AnyMap) image encoder
ffenc_pbm	FFmpeg PBM (Portable BitMap) image encoder
ffenc_pcx	FFmpeg PC Paintbrush PCX image encoder
ffenc_pgm	FFmpeg PGM (Portable GrayMap) image encoder
ffenc_pgmyuv	FFmpeg PGM YUV (Portable GrayMap YUV) image encoder
ffenc_png	FFmpeg PNG image encoder
ffenc_ppm	FFmpeg PPM (Portable PixelMap) image encoder
ffenc_qtrle	FFmpeg QuickTime Animation (RLE) video encoder
ffenc_roqvideo	FFmpeg id RoQ video encoder
ffenc_rv10	FFmpeg RealVideo 1.0 encoder
ffenc_rv20	FFmpeg RealVideo 2.0 encoder
ffenc_sgi	FFmpeg SGI image encoder
ffenc_snow	FFmpeg Snow encoder
ffenc_svk1	FFmpeg Sorenson Vector Quantizer 1 / Sorenson Video 1 / SVQ1 encoder
ffenc_targa	FFmpeg Truevision Targa image encoder
ffenc_tiff	FFmpeg TIFF image encoder
ffenc_wmv1	FFmpeg Windows Media Video 7 encoder
ffenc_wmv2	FFmpeg Windows Media Video 8 encoder
ffenc_zmbv	FFmpeg Zip Motion Blocks Video encoder
ffenc_aac	FFmpeg Advanced Audio Coding encoder
ffenc_ac3	FFmpeg ATSC A/52A (AC-3) encoder
ffenc_alac	FFmpeg ALAC (Apple Lossless Audio Codec) encoder
ffenc_mp2	FFmpeg MP2 (MPEG audio layer 2) encoder
ffenc_nellymoser	FFmpeg Nellymoser Asao encoder
ffenc_real_144	FFmpeg RealAudio 1.0 (14.4K) encoder encoder
ffenc_sonic	FFmpeg Sonic encoder
ffenc_sonics	FFmpeg Sonic lossless encoder
ffenc_wmav1	FFmpeg Windows Media Audio 1 encoder
ffenc_wmav2	FFmpeg Windows Media Audio 2 encoder
ffenc_roq_dpcm	FFmpeg id RoQ DPCM encoder

ffenc_adpcm_adx	FFmpeg SEGA CRI ADX ADPCM encoder
ffenc_g722	FFmpeg G.722 ADPCM encoder
ffenc_g726	FFmpeg G.726 ADPCM encoder
ffenc_adpcm_ima_qt	FFmpeg ADPCM IMA QuickTime encoder
ffenc_adpcm_ima_wav	FFmpeg ADPCM IMA WAV encoder
ffenc_adpcm_ms	FFmpeg ADPCM Microsoft encoder
ffenc_adpcm_swf	FFmpeg ADPCM Shockwave Flash encoder
ffenc_adpcm_yamaha	FFmpeg ADPCM Yamaha encoder
ffenc_ass	FFmpeg Advanced SubStation Alpha subtitle encoder
ffenc_dvbsub	FFmpeg DVB subtitles encoder
ffenc_dvbsub	FFmpeg DVD subtitles encoder
ffenc_xsub	FFmpeg DivX subtitles (XSUB) encoder
ffdec_aasc	FFmpeg Autodesk RLE decoder
ffdec_amv	FFmpeg AMV Video decoder
ffdec_anm	FFmpeg Deluxe Paint Animation decoder
ffdec_ansi	FFmpeg ASCII/ANSI art decoder
ffdec_asv1	FFmpeg ASUS V1 decoder
ffdec_asv2	FFmpeg ASUS V2 decoder
ffdec_aura	FFmpeg Auravision AURA decoder
ffdec_aura2	FFmpeg Auravision Aura 2 decoder
ffdec_avs	FFmpeg AVS (Audio Video Standard) video decoder
ffdec_bethsoftvid	FFmpeg Bethesda VID video decoder
ffdec_bfi	FFmpeg Brute Force & Ignorance decoder
ffdec_binkvideo	FFmpeg Bink video decoder
ffdec_bmp	FFmpeg BMP image decoder
ffdec_c93	FFmpeg Interplay C93 decoder
ffdec_cavs	FFmpeg Chinese AVS video (AVS1-P2, JiZhun profile) decoder
ffdec_cdgraphics	FFmpeg CD Graphics video decoder
ffdec_cinepak	FFmpeg Cinepak decoder
ffdec_cljr	FFmpeg Cirrus Logic AccuPak decoder
ffdec_camstudio	FFmpeg CamStudio decoder
ffdec_cyuv	FFmpeg Creative YUV (CYUV) decoder
ffdec_dnxhd	FFmpeg VC3/DNxHD decoder
ffdec_dpx	FFmpeg DPX image decoder
ffdec_dsicinvideo	FFmpeg Delphine Software International CIN video decoder
ffdec_dvvideo	FFmpeg DV (Digital Video) decoder
ffdec_dxa	FFmpeg Feeble Files/ScummVM DXA decoder
ffdec_eacmv	FFmpeg Electronic Arts CMV video decoder
ffdec_eamad	FFmpeg Electronic Arts Madcow Video decoder

ffdec_eatgq	FFmpeg Electronic Arts TGQ video decoder
ffdec_eatgv	FFmpeg Electronic Arts TGV video decoder
ffdec_eatqi	FFmpeg Electronic Arts TQI Video decoder
ffdec_8bps	FFmpeg QuickTime 8BPS video decoder
ffdec_8svx_exp	FFmpeg 8SVX exponential decoder
ffdec_8svx_fib	FFmpeg 8SVX fibonacci decoder
ffdec_escape124	FFmpeg Escape 124 decoder
ffdec_ffv1	FFmpeg FFmpeg video codec #1 decoder
ffdec_ffvhuff	FFmpeg Huffiyuv FFmpeg variant decoder
ffdec_flashsv	FFmpeg Flash Screen Video v1 decoder
ffdec_flic	FFmpeg Autodesk Animator Flic video decoder
ffdec_flv	FFmpeg Flash Video (FLV) / Sorenson Spark / Sorenson H.263 decoder
ffdec_4xm	FFmpeg 4X Movie decoder
ffdec_fraps	FFmpeg Fraps decoder
ffdec_FRWU	FFmpeg Forward Uncompressed decoder
ffdec_h261	FFmpeg H.261 decoder
ffdec_h263	FFmpeg H.263 / H.263-1996, H.263+ / H.263-1998 / H.263 version 2 decoder
ffdec_h263i	FFmpeg Intel H.263 decoder
ffdec_h264	FFmpeg H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 decoder
ffdec_huffyuv	FFmpeg Huffiyuv / HuffYUV decoder
ffdec_idcinvideo	FFmpeg id Quake II CIN video decoder
ffdec_iff_byterun1	FFmpeg IFF ByteRun1 decoder
ffdec_iff_ilbm	FFmpeg IFF ILBM decoder
ffdec_indeo2	FFmpeg Intel Indeo 2 decoder
ffdec_indeo3	FFmpeg Intel Indeo 3 decoder
ffdec_indeo5	FFmpeg Intel Indeo Video Interactive 5 decoder
ffdec_interplayvideo	FFmpeg Interplay MVE video decoder
ffdec_jpegls	FFmpeg JPEG-LS decoder
ffdec_kgv1	FFmpeg Kega Game Video decoder
ffdec_kmvc	FFmpeg Karl Morton's video codec decoder
ffdec_loco	FFmpeg LOCO decoder
ffdec_mdec	FFmpeg Sony PlayStation MDEC (Motion DECoder) decoder
ffdec_mimic	FFmpeg Mimic decoder
ffdec_mjpeg	FFmpeg MJPEG (Motion JPEG) decoder
ffdec_mjpegb	FFmpeg Apple MJPEG-B decoder
ffdec_mmvideo	FFmpeg American Laser Games MM Video decoder
ffdec_motionpixels	FFmpeg Motion Pixels video decoder
ffdec_mpeg4	FFmpeg MPEG-4 part 2 decoder
ffdec_mpegvideo	FFmpeg MPEG-1 video decoder

ffdec_msmpeg4v1	FFmpeg MPEG-4 part 2 Microsoft variant version 1 decoder
ffdec_msmpeg4v2	FFmpeg MPEG-4 part 2 Microsoft variant version 2 decoder
ffdec_msmpeg4	FFmpeg MPEG-4 part 2 Microsoft variant version 3 decoder
ffdec_msrle	FFmpeg Microsoft RLE decoder
ffdec_msvideo1	FFmpeg Microsoft Video 1 decoder
ffdec_mszh	FFmpeg LCL (LossLess Codec Library) MSZH decoder
ffdec_nuv	FFmpeg NuppelVideo/RTJPEG decoder
ffdec_pam	FFmpeg PAM (Portable AnyMap) image decoder
ffdec_pbm	FFmpeg PBM (Portable BitMap) image decoder
ffdec_pcx	FFmpeg PC Paintbrush PCX image decoder
ffdec_pgm	FFmpeg PGM (Portable GrayMap) image decoder
ffdec_pgmyuv	FFmpeg PGM YUV (Portable GrayMap YUV) image decoder
ffdec_pictor	FFmpeg Pictor/PC Paint decoder
ffdec_png	FFmpeg PNG image decoder
ffdec_ppm	FFmpeg PPM (Portable PixelMap) image decoder
ffdec_ptx	FFmpeg V.Flash PTX image decoder
ffdec_qdraw	FFmpeg Apple QuickDraw decoder
ffdec_qpeg	FFmpeg Q-team QPEG decoder
ffdec_qtrle	FFmpeg QuickTime Animation (RLE) video decoder
ffdec_r10k	FFmpeg AJA Kona 10-bit RGB Codec decoder
ffdec_rl2	FFmpeg RL2 video decoder
ffdec_roqvideo	FFmpeg id RoQ video decoder
ffdec_rpza	FFmpeg QuickTime video (RPZA) decoder
ffdec_rv10	FFmpeg RealVideo 1.0 decoder
ffdec_rv20	FFmpeg RealVideo 2.0 decoder
ffdec_rv30	FFmpeg RealVideo 3.0 decoder
ffdec_rv40	FFmpeg RealVideo 4.0 decoder
ffdec_sgi	FFmpeg SGI image decoder
ffdec_smackvid	FFmpeg Smacker video decoder
ffdec_smc	FFmpeg QuickTime Graphics (SMC) decoder
ffdec_snow	FFmpeg Snow decoder
ffdec_sp5x	FFmpeg Sunplus JPEG (SP5X) decoder
ffdec_sunrast	FFmpeg Sun Rasterfile image decoder
ffdec_svk1	FFmpeg Sorenson Vector Quantizer 1 / Sorenson Video 1 / SVQ1 decoder
ffdec_svk3	FFmpeg Sorenson Vector Quantizer 3 / Sorenson Video 3 / SVQ3 decoder
ffdec_targa	FFmpeg Truevision Targa image decoder
ffdec_thp	FFmpeg Nintendo Gamecube THP video decoder
ffdec_tiertexseqvideo	FFmpeg Tiertex Limited SEQ video decoder
ffdec_tiff	FFmpeg TIFF image decoder

ffdec_tmv	FFmpeg 8088flex TMV decoder
ffdec_truemotion1	FFmpeg Duck TrueMotion 1.0 decoder
ffdec_truemotion2	FFmpeg Duck TrueMotion 2.0 decoder
ffdec_camtasia	FFmpeg TechSmith Screen Capture Codec decoder
ffdec_txd	FFmpeg Renderware TXD (TeXture Dictionary) image decoder
ffdec_ultimotion	FFmpeg IBM UltiMotion decoder
ffdec_vb	FFmpeg Beam Software VB decoder
ffdec_vc1	FFmpeg SMPTE VC-1 decoder
ffdec_vcr1	FFmpeg ATI VCR1 decoder
ffdec_vmdvideo	FFmpeg Sierra VMD video decoder
ffdec_vmnc	FFmpeg VMware Screen Codec / VMware Video decoder
ffdec_vp3	FFmpeg On2 VP3 decoder
ffdec_vp5	FFmpeg On2 VP5 decoder
ffdec_vp6	FFmpeg On2 VP6 decoder
ffdec_vp6a	FFmpeg On2 VP6 (Flash version, with alpha channel) decoder
ffdec_vp6f	FFmpeg On2 VP6 (Flash version) decoder
ffdec_vp8	FFmpeg On2 VP8 decoder
ffdec_vqavideo	FFmpeg Westwood Studios VQA (Vector Quantized Animation) video decoder
ffdec_wmv1	FFmpeg Windows Media Video 7 decoder
ffdec_wmv2	FFmpeg Windows Media Video 8 decoder
ffdec_wmv3	FFmpeg Windows Media Video 9 decoder
ffdec_wmv1	FFmpeg Winnov WNV1 decoder
ffdec_xan_wc3	FFmpeg Wing Commander III / Xan decoder
ffdec_xl	FFmpeg Miro VideoXL decoder
ffdec_yop	FFmpeg Psygnosis YOP Video decoder
ffdec_zlib	FFmpeg LCL (LossLess Codec Library) ZLIB decoder
ffdec_zmbv	FFmpeg Zip Motion Blocks Video decoder
ffdec_aac	FFmpeg Advanced Audio Coding decoder
ffdec_aac_latm	FFmpeg AAC LATM (Advanced Audio Codec LATM syntax) decoder
ffdec_ac3	FFmpeg ATSC A/52A (AC-3) decoder
ffdec_alac	FFmpeg ALAC (Apple Lossless Audio Codec) decoder
ffdec_als	FFmpeg MPEG-4 Audio Lossless Coding (ALS) decoder
ffdec_amrnb	FFmpeg Adaptive Multi-Rate NarrowBand decoder
ffdec_ape	FFmpeg Monkey's Audio decoder
ffdec_atrac1	FFmpeg Atrac 1 (Adaptive TRansform Acoustic Coding) decoder
ffdec_atrac3	FFmpeg Atrac 3 (Adaptive TRansform Acoustic Coding 3) decoder
ffdec_binkaudio_dct	FFmpeg Bink Audio (DCT) decoder
ffdec_binkaudio_rdf	FFmpeg Bink Audio (RDFT) decoder
ffdec_cook	FFmpeg COOK decoder

ffdec_dca	FFmpeg DCA (DTS Coherent Acoustics) decoder
ffdec_dsicinaudio	FFmpeg Delphine Software International CIN audio decoder
ffdec_eac3	FFmpeg ATSC A/52B (AC-3, E-AC-3) decoder
ffdec_flac	FFmpeg FLAC (Free Lossless Audio Codec) decoder
ffdec_gsm	FFmpeg GSM decoder
ffdec_gsm_ms	FFmpeg GSM Microsoft variant decoder
ffdec_imc	FFmpeg IMC (Intel Music Coder) decoder
ffdec_mace3	FFmpeg MACE (Macintosh Audio Compression/Expansion) 3
ffdec_mace6	FFmpeg MACE (Macintosh Audio Compression/Expansion) 6
ffdec_mlp	FFmpeg MLP (Meridian Lossless Packing) decoder
ffdec_mp1float	FFmpeg MP1 (MPEG audio layer 1) decoder
ffdec_mp2float	FFmpeg MP2 (MPEG audio layer 2) decoder
ffdec_mpc7	FFmpeg Musepack SV7 decoder
ffdec_mpc8	FFmpeg Musepack SV8 decoder
ffdec_nellymoser	FFmpeg Nellymoser Asao decoder
ffdec_qcelp	FFmpeg QCELP / PureVoice decoder
ffdec_qdm2	FFmpeg QDesign Music Codec 2 decoder
ffdec_real_144	FFmpeg RealAudio 1.0 (14.4K) decoder
ffdec_real_288	FFmpeg RealAudio 2.0 (28.8K) decoder
ffdec_shorten	FFmpeg Shorten decoder
ffdec_sipr	FFmpeg RealAudio SIPR / ACELP.NET decoder
ffdec_smackaud	FFmpeg Smacker audio decoder
ffdec_sonic	FFmpeg Sonic decoder
ffdec_truehd	FFmpeg TrueHD decoder
ffdec_truespeech	FFmpeg DSP Group TrueSpeech decoder
ffdec_tta	FFmpeg True Audio (TTA) decoder
ffdec_twinvq	FFmpeg VQF TwinVQ decoder
ffdec_vmdaudio	FFmpeg Sierra VMD audio decoder
ffdec_wmapro	FFmpeg Windows Media Audio 9 Professional decoder
ffdec_wmav1	FFmpeg Windows Media Audio 1 decoder
ffdec_wmav2	FFmpeg Windows Media Audio 2 decoder
ffdec_wmavoice	FFmpeg Windows Media Audio Voice decoder
ffdec_ws_snd1	FFmpeg Westwood Audio (SND1) decoder
ffdec_pcm_lxf	FFmpeg PCM signed 20-bit little-endian planar decoder
ffdec_interplay_dpcm	FFmpeg DPCM Interplay decoder
ffdec_roq_dpcm	FFmpeg DPCM id RoQ decoder
ffdec_sol_dpcm	FFmpeg DPCM Sol decoder
ffdec_xan_dpcm	FFmpeg DPCM Xan decoder
ffdec_adpcm_4xm	FFmpeg ADPCM 4X Movie decoder

ffdec_adpcm_adx	FFmpeg SEGA CRI ADX ADPCM decoder
ffdec_adpcm_ct	FFmpeg ADPCM Creative Technology decoder
ffdec_adpcm_ea	FFmpeg ADPCM Electronic Arts decoder
ffdec_adpcm_ea_maxis_xa	FFmpeg ADPCM Electronic Arts Maxis CDROM XA decoder
ffdec_adpcm_ea_r1	FFmpeg ADPCM Electronic Arts R1 decoder
ffdec_adpcm_ea_r2	FFmpeg ADPCM Electronic Arts R2 decoder
ffdec_adpcm_ea_r3	FFmpeg ADPCM Electronic Arts R3 decoder
ffdec_adpcm_ea_xas	FFmpeg ADPCM Electronic Arts XAS decoder
ffdec_g722	FFmpeg G.722 ADPCM decoder
ffdec_g726	FFmpeg G.726 ADPCM decoder
ffdec_adpcm_ima_amv	FFmpeg ADPCM IMA AMV decoder
ffdec_adpcm_ima_dk3	FFmpeg ADPCM IMA Duck DK3 decoder
ffdec_adpcm_ima_dk4	FFmpeg ADPCM IMA Duck DK4 decoder
ffdec_adpcm_ima_ea_eacs	FFmpeg ADPCM IMA Electronic Arts EACS decoder
ffdec_adpcm_ima_ea_sead	FFmpeg ADPCM IMA Electronic Arts SEAD decoder
ffdec_adpcm_ima_iss	FFmpeg ADPCM IMA Funcom ISS decoder
ffdec_adpcm_ima_qt	FFmpeg ADPCM IMA QuickTime decoder
ffdec_adpcm_ima_smjpeg	FFmpeg ADPCM IMA Loki SDL MJPEG decoder
ffdec_adpcm_ima_wav	FFmpeg ADPCM IMA WAV decoder
ffdec_adpcm_ima_ws	FFmpeg ADPCM IMA Westwood decoder
ffdec_adpcm_ms	FFmpeg ADPCM Microsoft decoder
ffdec_adpcm_sbpro_2	FFmpeg ADPCM Sound Blaster Pro 2-bit decoder
ffdec_adpcm_sbpro_3	FFmpeg ADPCM Sound Blaster Pro 2.6-bit decoder
ffdec_adpcm_sbpro_4	FFmpeg ADPCM Sound Blaster Pro 4-bit decoder
ffdec_adpcm_swf	FFmpeg ADPCM Shockwave Flash decoder
ffdec_adpcm_thp	FFmpeg ADPCM Nintendo Gamecube THP decoder
ffdec_adpcm_xa	FFmpeg ADPCM CDROM XA decoder
ffdec_adpcm_yamaha	FFmpeg ADPCM Yamaha decoder
ffdec_ass	FFmpeg Advanced SubStation Alpha subtitle decoder
ffdec_dvbsub	FFmpeg DVB subtitles decoder
ffdec_dvdsup	FFmpeg DVD subtitles decoder
ffdec_pgssub	FFmpeg HDMV Presentation Graphic Stream subtitles decoder
ffdec_xsub	FFmpeg XSUB decoder

Third Party Solutions

Third parties like Ittiam and VisualON provide highly optimized ARM only codecs on Linux, WinCE and Android OS.

Software Components & Dependencies

The following lists some of the software components and dependencies associated with the Sitara SDK.

Dependencies: Required packages to build Gstreamer on Ubuntu:

```
sudo apt-get install automake autoconf libtool docbook-xml docbook-xsl fop libxml2 gnome-doc-utils
```

- build-essential
- libtool
- automake
- autoconf
- git-core
- svn
- liboil0.3-dev
- libxml2-dev
- libglib2.0-dev
- gettext
- corkscrew
- socket
- libfaad-dev
- libfaac-dev

Software components for Sitara SDK Release:

- glib
- gstreamer
- liboil
- gst-plugins-good
- gst-ffmpeg
- gst-plugins-bad
- gst-plugins-base

Re-enabling Mp3 and Mpeg2 decode in the AM SDK

Starting with version 05.05.01.00, mp3 and mpeg2 codecs are no longer distributed as part of the SDK. These plugins can be re-enabled by the end user through rebuilding the `gst-plugins-ugly` package. The following instructions have been tested with `gst-plugins-ugly-0.10.19` which can be found at gstreamer.freedesktop.org ^[6].

Note that these instructions will work for any of the gstreamer plugin packages found in the sdk.

- Source environment-setup at the terminal
- Navigate into the example-applications path under the SDK install directory
- Extract the GStreamer plug-in source archive
- Navigate into the folder that was created
- On the command line type `./configure --host=arm-arago-linux-gnueabi --prefix=/usr`
- Notice that some components are not built because they have dependencies that are not part of our SDK
- Run `make` to build the plugins.
- Run `make install DESTDIR=<PATH TO TARGET ROOT>`

Other

- For technical support please post your questions at <http://e2e.ti.com> or search forum post Database.

References

- [1] <http://www.arm.com/products/processors/technologies/neon.php>
- [2] <http://oprofile.sourceforge.net/about/>
- [3] http://elinux.org/OMAP_Power_Management
- [4] <http://ffmpeg.org/>
- [5] <http://ffmpeg.org/legal.html>
- [6] <http://gstreamer.freedesktop.org/>

Camera Users Guide

Return to the [Sitara Linux Software Developer's Guide](#)



Introduction

This users guide provides an overview of camera loopback example integrated in matrix application on AM37x platform. This application can be executed by selecting the "Camera" icon at the top-level matrix.



Supported Platforms

- AM37x

Camera Loopback Example

Camera example application demonstrates streaming of YUV 4:2:2 parallel data from Leopard Imaging LI-3M02 daughter card with MT9T111 sensor on AM37x EVM. Here, driver allocated buffers are used to capture data from the sensor and display to any active display device (LCD/DVI/TV) in VGA resolution (640x480).

Since the LCD is at inverted VGA resolution video buffer is rotated by 90 degree. Rotation is enabled using the hardware rotation engine called Virtual Rotated Frame Buffer (VRFB), VRFB provides four virtual frame buffers: 0, 90, 180 and 270. For statically compiled drivers, VRFB buffer can be declared in the bootloader. In this example, data is displayed through video1 pipeline and "omap_vout.vid1_static_vrfb_alloc=y" bootargument is included in the bootargs.

Sensor Daughter card

Smart sensor modules have control algorithms such as AE, AWB and AF are implemented in the module. Third parties like Leopard Imaging and E-Con Systems provide smart sensor solutions for OMAP35x/AM37x Rev G EVM and Beagle Board. SDK 5.02 integrates support for LI-3M02 daughter card which has MT9T111 sensor, the sensor is configured in 8-bit YUV4:2:2 mode. **Leopard Imaging** ^[1] daughter card LI-3M02 can be ordered from their website.

For additional support on daughter cards or sensor details please contact the respective third parties

- **Leopard Imaging** ^[2]

Contact: support@leopardimaging.com

- **E-Con Systems** ^[3]

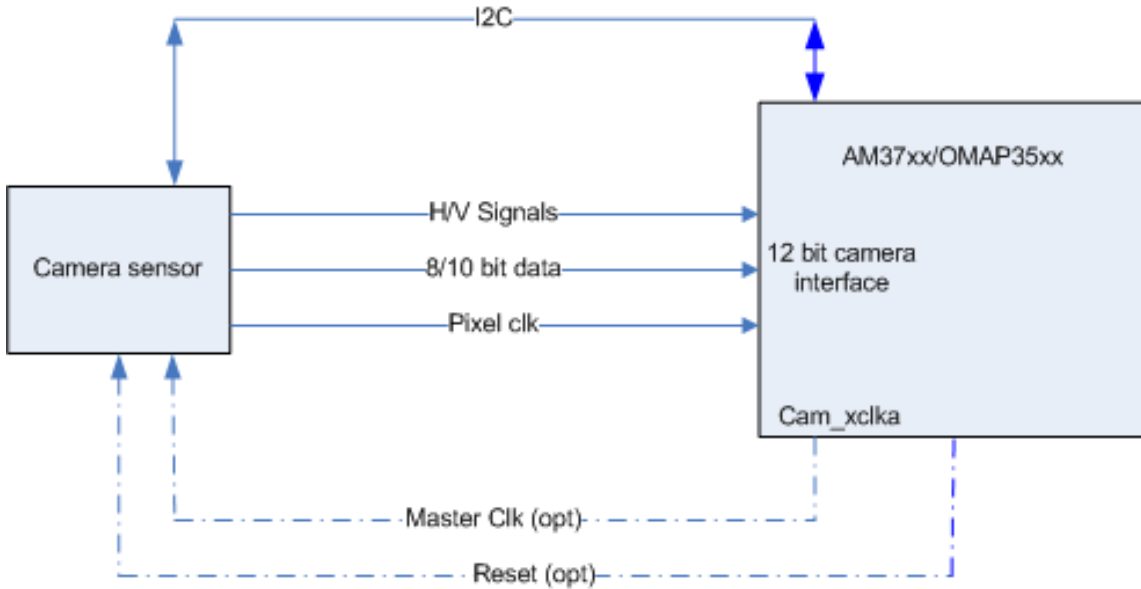
Contact: **Harishankkar** ^[4]

Connectivity

OMAP35xx/AM/DM37xx	Signal Description	Comment	IO Type
cam_hs	Camera Horizontal Sync		IO
cam_vs	Camera Vertical Sync		IO
cam_xclka	Camera Clock Output a	Optional	O
cam_xclkb	Camera Clock Output b	Optional	O
cam_[d0:12]	Camera/Video Digital Image Data		I
cam_fld	Camera Field ID	Optional	IO
cam_pclk	Camera Pixel Clock		I
cam_wen	Camera Write Enable	Optional	I
cam_strobe	Flash Strobe Control Signal	Optional	O
cam_global_reset	Global Reset - Strobe Synchronization	Optional	IO
cam_shutter	Mechanical Shutter Control Signal	Optional	O

LI-3M02 daughter card is connected to the 26-pin J31 connector on the EVM. 24MHz oscillator on the LI daughter card is used to drive the sensor device, alternatively camera clock output generated by processor ISP block can be used to drive the sensor pixel clock. Horizontal sync (HS) and vertical sync (VS) lines are used to detect end of line

and field respectively. Sensor data line CMOS D4:D11 are connected to the processor D4:D11.



Connectivity of LI-3M02 daughter card to AM/DM37x

- Note: H/V signal connectivity not required for ITU-R BT.656 input since synchronization signals are extracted from BT.656 bit stream

Media Controller Framework

SDK 5.02 includes support for media controller framework, for additional details of media controller framework and its usage please refer the PSP capture driver users guide ^[5]

In order to enable streaming each links between each entity is established. Currently only MT9T111/TV5146 => CCDC => Memory has been validated.

```
Media: Opened Media Device
Enumerating media entities
[1]:OMAP3 ISP CCP2
[2]:OMAP3 ISP CCP2 input
[3]:OMAP3 ISP CSI2a
[4]:OMAP3 ISP CSI2a output
[5]:OMAP3 ISP CCDC
[6]:OMAP3 ISP CCDC output
[7]:OMAP3 ISP preview
[8]:OMAP3 ISP preview input
[9]:OMAP3 ISP preview output
[10]:OMAP3 ISP resizer
[11]:OMAP3 ISP resizer input
[12]:OMAP3 ISP resizer output
[13]:OMAP3 ISP AEWB
[14]:OMAP3 ISP AF
[15]:OMAP3 ISP histogram
[16]:mt9t111 2-003c
[17]:tvp514x 3-005c
Total number of entities: 17
Enumerating links/pads for entities
```

```
pads for entity 1=(0 INPUT) (1 OUTPUT)
[1:1]===>[5:0] INACTIVE

pads for entity 2=(0 OUTPUT)
[2:0]===>[1:0] INACTIVE

pads for entity 3=(0 INPUT) (1 OUTPUT)
[3:1]===>[4:0] INACTIVE
[3:1]===>[5:0] INACTIVE

pads for entity 4=(0 INPUT)

pads for entity 5=(0 INPUT) (1 OUTPUT) (2 OUTPUT)
[5:1]===>[6:0] ACTIVE
[5:2]===>[7:0] INACTIVE
[5:1]===>[10:0] INACTIVE
[5:2]===>[13:0] ACTIVE
[5:2]===>[14:0] ACTIVE
[5:2]===>[15:0] ACTIVE

pads for entity 6=(0 INPUT)

pads for entity 7=(0 INPUT) (1 OUTPUT)
[7:1]===>[9:0] INACTIVE
[7:1]===>[10:0] INACTIVE

pads for entity 8=(0 OUTPUT)
[8:0]===>[7:0] INACTIVE

pads for entity 9=(0 INPUT)

pads for entity 10=(0 INPUT) (1 OUTPUT)
[10:1]===>[12:0] INACTIVE

pads for entity 11=(0 OUTPUT)
[11:0]===>[10:0] INACTIVE

pads for entity 12=(0 INPUT)

pads for entity 13=(0 INPUT)

pads for entity 14=(0 INPUT)

pads for entity 15=(0 INPUT)

pads for entity 16=(0 OUTPUT)
[16:0]===>[5:0] ACTIVE
```

```

pads for entity 17=(0 OUTPUT)
[17:0]===>[5:0] INACTIVE

Enabling link [MT9T111]===>[ccdc]
[MT9T111]===>[ccdc]     enabled
Enabling link [ccdc]===>[video_node]
[ccdc]===>[video_node]  enabled

Capture: Opened Channel
successfully format is set on all pad [WxH] - [640x480]
Capture: Capable of streaming
Capture: Number of requested buffers = 3
Capture: Init done successfully

Display: Opened Channel
Display: Capable of streaming
Display: Number of requested buffers = 3
Display: Init done successfully

Display: Stream on...
Capture: Stream on...

```

AM/DM37x ISP Configuration

ISP CCDC block should be configured to enable 8-bit YUV4:2:2 parallel data input, the registers below provide details of ISP and CCDC registers in this mode.

ISP Registers:

```

ISP_CTRL: 0x480BC040
29C14C
ISP_SYSCONFIG: 0x480BC004
2000
ISP_SYSSTATUS: 0x480BC008
1
ISP_IRQ0ENABLE: 0x480BC00C
811B33F9
ISP_IRQ0STATUS: 0x480BC010
0
ISP_IRQ1ENABLE: 0x480BC014
0
ISP_IRQ1STATUS: 0x480BC018
80000300

```

CCDC Registers:

```

CCDC_PID: 0x480BC600
1FE01

```

```
CCDC_PCR
0
CCDC_SYN_MODE: 0x480BC604
31000
CCDC_HD_VD_WID: 0x480BC60C
0
CCDC_PIX_LINES: 0x480BC610
0
CCDC_HORZ_INFO: 0x480BC614
27F
CCDC_VERT_START: 0x480BC618
0
CCDC_VERT_LINES: 0x480BC61C
1DF
CCDC_CULLING: 0x480BC620
FFFF00FF
CCDC_HSIZE_OFF: 0x480BC624
500
CCDC_SDOFST: 0x480BC628
0
CCDC_SDR_ADDR: 0x480BC62C
1C5000
CCDC_CLAMP: 0x480BC630
10
CCDC_DCSUB: 0x480BC634
40
CCDC_COLPTN: 0x480BC63
0
CCDC_BLKCMP: 0x480BC63C
0
CCDC_FPC: 0x480BC640
0
CCDC_FPC_ADDR: 0x480BC644
0
CCDC_VDINT: 0x480BC648
1DE0140
CCDC_ALAW: 0x480BC64C
0
CCDC_REC: 0x480BC650
0
CCDC_CFG: 0x480BC65
8800
CCDC_FMTCFG: 0x480BC658
0
CCDC_FMT_HORZ: 0x480BC65C
280
CCDC_FMT_VERT: 0x480BC660
```

```
1E0
CCDC_PRGEVEN0: 0x480BC684
0
CCDC_PRGEVEN1: 0x480BC688
0
CCDC_PRGODD0: 0x480BC68C
0
CCDC_PRGODD1: 0x480BC690
0
CCDC_VP_OUT: 0x480BC694
3BE2800
CCDC_LSC_CONFIG: 0x480BC698
6600
CCDC_LSC_INITIAL: 0x480BC69C
0
CCDC_LSC_TABLE_BA: 0x480BC6A0
0
CCDC_LSC_TABLE_OF: 0x480BC6A4
0
```

Other

- For technical support please post your questions at <http://e2e.ti.com>^[6] or search forum post Database.

References

- [1] https://www.leopardimaging.com/3M_Camera_Module_Board.html
- [2] <https://www.leopardimaging.com>
- [3] <http://www.e-consystems.com>
- [4] <mailto:harishankkar@e-consystems.com>
- [5] http://processors.wiki.ti.com/index.php/UserGuideOmap35xCaptureDriver_PSP_04.02.00.07
- [6] <http://e2e.ti.com>

Cryptography Users Guide

Return to the [Sitara Linux Software Developer's Guide](#)



Overview

This article provides a description of the example applications under the cryptography page of the Matrix application that comes with the Sitara SDK. This page is labeled "Cryptos" in the top-level Matrix GUI.



Devices Supported

All Sitara SDK's provide cryptography through OpenSSL, but the following devices have cryptographic hardware accelerators and drivers:

- AM3517 - AES, DES, 3DES, SHA, MD5
- AM37x - AES, DES, 3DES, SHA, MD5
- AM335x - AES, SHA, MD5, RNG

The example applications have been designed to use hardware accelerators when they are available and fall back to a pure software implementation when hardware accelerators are not available.

Cryptography Examples

All of the examples under the Cryptos page use the OpenSSL command line application to perform cryptographic functions. A comprehensive list of cryptographic functionality using OpenSSL is beyond the scope of the out-of-box experience intended with the Matrix GUI. However, the examples present a nice variety of cryptographic functions that are available with OpenSSL on the Sitara platform.



OpenSSL Performance

This example executes the OpenSSL built-in speed test for a variety of cryptographic algorithms. The results of the test are displayed on the screen and also written to the file `OpenSSLspeedResults.txt` in the top level directory of the target filesystem.

Certificate Generation

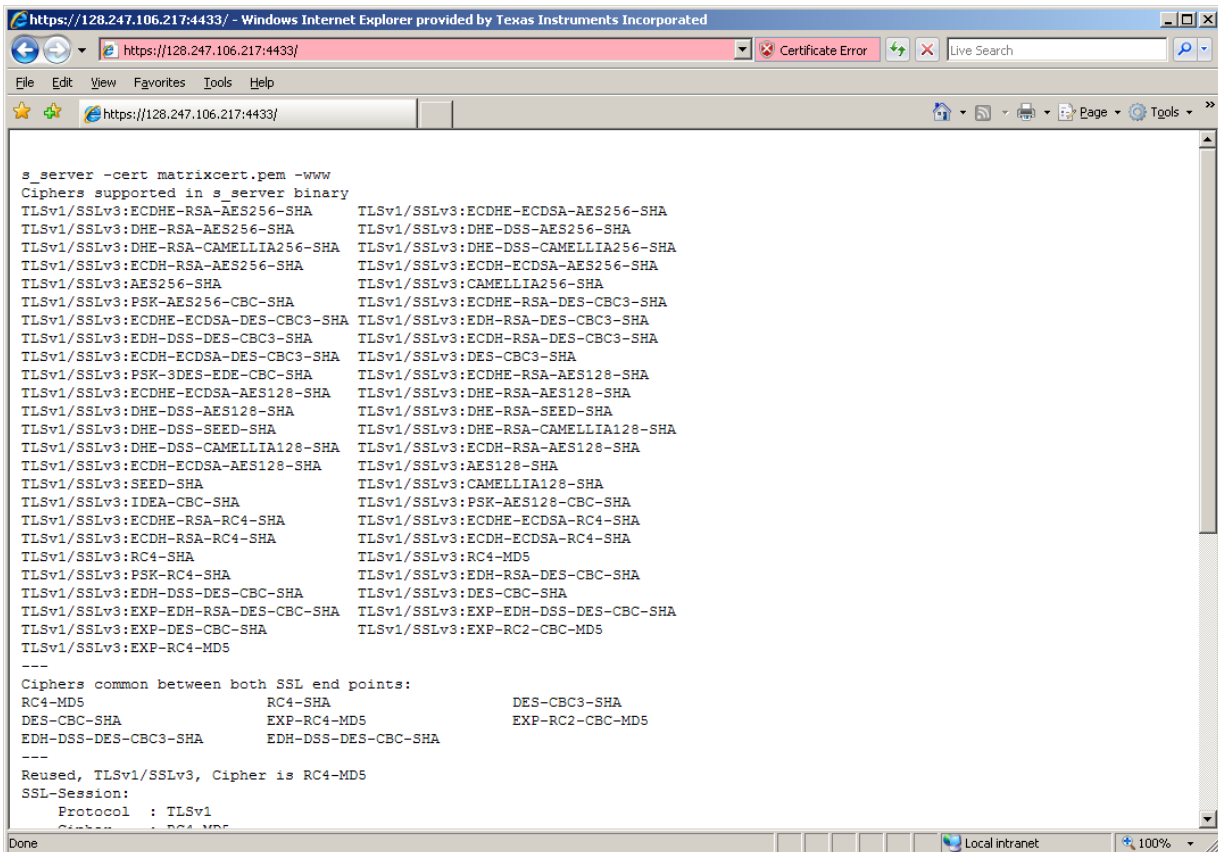
This example generates a web page certificate for use with a secure web server. The certificate is held in the file `matrixcert.pem`. This file will appear in the top level directory of the target file system. If the certificate already exists, then the example will fail and prompt the user to delete the existing certificate (`matrixcert.pem`) before generating a new one.

Public Key Generation

This example generates a public key based on the certificate (matrixcert.pem) generated in the previous example. If the certificate does not exist the example will fail and prompt the user to first generate the certificate before trying to generate the public key. The public key will be saved to a file pubkey.pem in the top level directory of the target file system.

Secure Server

Once the web certificate (matrixcert.pem) has been generated, the secure web server can be started on the target board. Pointing a modern web browser at the target should generate a warning that the certificate is self-signed. This means that the certificate has not been verified by a trusted third party such as Verisign. Depending on the browser, you can view details of the certificate. In the example below the target board has an IP address of 128.247.106.217. When Internet Explorer is pointed to the URL https://128.247.106.247:4433, it first warns the user that there is a problem with the website's security certificate. Make sure that you use https:// and the port number :4433 with the IP address in the URL of the browser. Clicking the link to continue anyway provides the page below. And clicking the "Certificate Error" button at the top of the page will provide details of the certificate.



Certificate Info

This example simply prints out details of the generated certificate (matrixcert.pem).

AES 256

This example will perform an encrypt/decrypt cycle on a 10M file of random data with the AES 256 algorithm. The 10M file called rnddata will be generated if it doesn't already exist. The result of the decryption is compared to the original file and the results are displayed to the screen.

SHA Hash

This example will perform a SHA1 hash function on the 10M file of random data (rnddata). If the file doesn't exist, it is generated. The result of the hash is displayed to the screen.

Building the Driver

For devices with available cryptographic hardware accelerators, a Linux driver and additionally an OCF kernel module (for OpenSSL) is needed to access them. Other devices use the pure software implementation of OpenSSL for the crypto demos.

Note that currently only the AM35x/37x GP device and AM335x GP devices use Open Cryptographic Framework (OCF) to access HW accelerators under OpenSSL. This section covers how to build the drivers for devices with crypto accelerators. Following this we cover how to build the OCF kernel module. See the Devices Supported section above for a list of the supported devices.

AM35x/37x, AM335x - AES, SHA/MD5 Drivers

Starting with SDK 5.05.00.00, the driver for AM335x is completely integrated into the kernel source. The pre-built kernel that comes with the SDK already has the AES and SHA/MD5 drivers built-in to the kernel. The kernel configuration has already been set up in the SDK and no further configuration is needed for the drivers to be built-in to the kernel. The configuration of the random number generator does require an extra step and this is detailed in the next section.

For reference, the configuration details are shown below. The configuration of the AES and SHA/MD5 driver is done under the Hardware crypto devices sub-menu of the Cryptographic API menu in the kernel configuration.

```
--- Cryptographic API
  [*] Hardware crypto devices --->
    --- Hardware crypto devices
      <*> Support for OMAP4 AES hw engine
      <*> Support for OMAP4 SHA/MD5 hw engine
```

Messages printed during bootup will indicate that initialization of the crypto modules has taken place.

```
[ 1.695495] omap4_aes_mod_init: loading AM33X AES driver
[ 1.701202] omap4-aes omap4-aes: AM33X AES hw accel rev: 3.02
[ 1.707733] omap4_aes_probe: probe() done
[ 1.712402] omap4_sham_mod_init: loading AM33X SHA/MD5 driver
[ 1.718536] omap4-sham omap4-sham: AM33X SHA/MD5 hw accel rev: 4.03
[ 1.733276] omap4_sham_probe: probe() done
```

Build the OCF kernel module using SDK

For using OpenSSL to access the Crypto Hardware Accelerator Drivers above, the Open Cryptographic Framework (OCF) is required (can be built as module). The framework is not officially in the kernel and was ported to Linux under the name "ocf-linux". Starting with the Sitara Linux SDK 05.04 there is now a top-level Makefile target for building the crypto driver . The following commands can be used to initiate the build:

```
host# cd <SDK INSTALL DIR>
host# make ti-ocf-crypto-module
```

To install the driver into the /lib/modules/<kernel version>/crypto/ocf directory the following command can be used:

```
host# cd <SDK INSTALL DIR>
host# make ti-ocf-crypto-module_install
```

NOTE: The above install instruction assumes that you have run the SDK setup script or have modified the Rules.make file in the Sitara Linux SDK installation directory to have a valid value for the DESTDIR variable that points to the root of your target file system, or a temporary directory where you want the drivers to be installed.

For a more detailed description on how the cryptographic software is built and installed in the SDK, please see the following links:

How to [Build OCF for Sitara](#)

How to [Build OpenSSL for Sitara](#)

Using Cryptographic Hardware Accelerators

Using the TRNG Hardware Accelerator

For the True Random Number Generator (TRNG) in the AM335x there is an extra step needed in the Linux kernel configuration that needs to be performed to get the driver included in the kernel.

Use the menuconfig command according to the "Customizing the Configuration" section to get into the configuration menu for the kernel.

processors.wiki.ti.com/index.php/AMSDK_Linux_User%27s_Guide#Customizing_the_Configuration^[1]

In the configuration menu, scroll down to Device Drivers and hit enter. Now scroll to Character devices and hit enter.

```
Device Drivers --->
  Character devices --->
    < > Hardware Random Number Generator Core support
```

Use the spacebar to select the Hardware Random Number Generator support and also select OMAP4 Random Number Generator support. The screen should look like below.

```
<*> Hardware Random Number Generator Core support
  < > Timer IOMEM HW Random Number Generator support (NEW)
  <*> OMAP4 Random Number Generator support (NEW)
```

Now rebuild the kernel according to the User's Guide and boot up the board with the resulting kernel. The following message should be part of the boot-up messages.

```
[ 0.944152] omap4_rng omap4_rng: OMAP4 Random Number Generator ver. 2.00
```

Once the system is booted up, the hwrng device should now show up in the filesystem.

```
root@am335x-evm:~# ls -l /dev/hwrng
crw----- 1 root root 10, 183 Jan 1 2000 /dev/hwrng
root@am335x-evm:~#
```

Use cat on this device to generate random numbers.

```
root@am335x-evm:~# cat /dev/hwrng | od -x
0000000 b2bd ae08 4477 be48 4836 bf64 5d92 01c9
0000020 0cb6 7ac5 16f9 8616 a483 7dfd 6bf4 3aa5
0000040 d693 db24 d917 5ee7 feb7 34c3 34e9 e7a5
0000060 36b7 ea85 fc17 0e66 555c 0934 7a0c 4c69
0000100 523b 9f21 1546 fddb d58b e5ed 142a 6712
0000120 8d76 8f80 a6d2 30d8 d107 32bc 7f45 f997
0000140 9d5d 0d0c f1f0 64f9 a77f 408f b0c1 f5a0
0000160 39c6 f0ae 4b59 1a76 84a7 a364 8964 f557
root@am335x-evm:~#
```

AES, SHA, TRNG Hardware Accelerators using OpenSSL (requires OCF-linux kernel support)

The device drivers for AES and SHA/MD5 hardware acceleration is configured and built into the kernel by default in SDK 5.05.00.00. No other special setup is needed for OpenSSL to access the crypto modules.

First, the kernel from the SDK must be configured and built according to the SDK User's Guide.

processors.wiki.ti.com/index.php/AMSDK_Linux_User%27s_Guide ^[2]

The General Purpose (GP) EVMs on TI SoCs allows access to built in cryptographic accelerators. Inorder to use these drivers from OpenSSL, the drivers on their own have no contact with userspace. For this, a special driver is available which abstracts the access to these accelerators through the Open Cryptographic Framework for Linux (OCF-Linux).

The demo application under the crypto menu of Matrix will load and use the OCF driver kernel modules automatically to perform hardware accelerated crypto functions. The process of manually loading the kernel modules and using the driver is explained below.

OCF-Linux is itself a special device driver which provides a general interface for higher level applications such as OpenSSL to access hardware accelerators.

Kernel modules are required for OCF-linux, OCF cryptosoft and OCF cryptodev. All these 3 modules are a part of the OCF-linux package.

The filesystem which comes with the SDK comes built with the OCF-Linux kernel modules and the TI driver which directly accesses the hardware accelerators is built into the kernel.

From the target boards perspective the drivers are located in the following directories:

```
/lib/modules/3.8.4-01427-g1eb3bbb-dirty/kernel/crypto/ocf/cryptosoft.ko
/lib/modules/3.8.4-01427-g1eb3bbb-dirty/kernel/crypto/ocf/cryptodev.ko
/lib/modules/3.8.4-01427-g1eb3bbb-dirty/kernel/crypto/ocf/ocf.ko
```

To use the drivers they must first be installed. Use the insmod command to install the drivers. The TI crypto driver allows a parameter to be passed in to indicate if DMA should be used. The following log shows the commands used to install the modules and query the system for the state of all system modules.

```
root@am37x-evm:~# lsmod
Module                               Size  Used by
```

```
cryptosoft          14350  0
cryptodev           11962  0
ocf                 25277  2 cryptosoft,cryptodev
root@am37x-evm:~#
```

After the modules are installed, OpenSSL commands may be executed which take advantage of the hardware accelerators through the OCF-Linux driver. The following example demonstrates the OpenSSL built-in speed test to demonstrate performance. The addition of the parameter **-engine cryptodev** tells OpenSSL to use the OCF-Linux driver if it exists.

```
root@am37x-evm:~# openssl speed -evp aes-128-cbc -engine cryptodev

engine "cryptodev" set.

Doing aes-128-cbc for 3s on 16 size blocks: 108107 aes-128-cbc's in 0.16s
Doing aes-128-cbc for 3s on 64 size blocks: 103730 aes-128-cbc's in 0.20s
Doing aes-128-cbc for 3s on 256 size blocks: 15181 aes-128-cbc's in 0.03s
Doing aes-128-cbc for 3s on 1024 size blocks: 15879 aes-128-cbc's in 0.03s
Doing aes-128-cbc for 3s on 8192 size blocks: 4879 aes-128-cbc's in 0.02s

OpenSSL 1.0.0b 16 Nov 2010

built on: Thu Jan 20 10:23:44 CST 2011

options:bn(64,32) rc4(ptr,int) des(idx,riscl,2,long) aes(partial) idea(int) blowfish(idx)

compiler: arm-none-linux-gnueabi-gcc -march=armv7-a -mtune=cortex-a8 -mfpu=neon -mfloat-abi=softfp -mthumb-interwork -mno-thumb -fPS

The 'numbers' are in 1000s of bytes per second processed.

type 16 bytes 64 bytes 256 bytes 1024 bytes 8192 bytes
aes-128-cbc 10810.70k 33193.60k 129544.53k 542003.20k 1998438.40k

root@am37x-evm:~#
root@am37x-evm:~#
root@am37x-evm:~#
```

Using the Linux `time -v` function gives more information about CPU usage during the test.

```
root@am37x-evm:~# time -v openssl speed -evp aes-128-cbc -engine cryptodev

engine "cryptodev" set.

Doing aes-128-cbc for 3s on 16 size blocks: 108799 aes-128-cbc's in 0.17s
Doing aes-128-cbc for 3s on 64 size blocks: 102699 aes-128-cbc's in 0.18s
Doing aes-128-cbc for 3s on 256 size blocks: 16166 aes-128-cbc's in 0.03s
Doing aes-128-cbc for 3s on 1024 size blocks: 15080 aes-128-cbc's in 0.03s
Doing aes-128-cbc for 3s on 8192 size blocks: 4838 aes-128-cbc's in 0.03s

OpenSSL 1.0.0b 16 Nov 2010

built on: Thu Jan 20 10:23:44 CST 2011

options:bn(64,32) rc4(ptr,int) des(idx,riscl,2,long) aes(partial) idea(int) blowfish(idx)

compiler: arm-none-linux-gnueabi-gcc -march=armv7-a -mtune=cortex-a8 -mfpu=neon -mfloat-abi=softfp -mthumb-interwork -mno-thumb -fPS

The 'numbers' are in 1000s of bytes per second processed.

type 16 bytes 64 bytes 256 bytes 1024 bytes 8192 bytes
aes-128-cbc 10239.91k 36515.20k 137949.87k 514730.67k 1321096.53k

Command being timed: "openssl speed -evp aes-128-cbc -engine cryptodev"

User time (seconds): 0.46
System time (seconds): 5.89
Percent of CPU this job got: 42%

Elapsed (wall clock) time (h:mm:ss or m:ss): 0m 15.06s
```

```

Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 7104
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 479
Voluntary context switches: 36143
Involuntary context switches: 211570
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0

```

When the cryptodev driver is removed, OpenSSL reverts to the software implementation of the crypto algorithm. The performance using the software only implementation can be compared to the previous test.

```

root@am37x-evm:~# rmmod cryptodev
root@am37x-evm:~# time -v openssl speed -evp aes-128-cbc
Doing aes-128-cbc for 3s on 16 size blocks: 697674 aes-128-cbc's in 2.99s
Doing aes-128-cbc for 3s on 64 size blocks: 187556 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 256 size blocks: 47922 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 1024 size blocks: 12049 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 8192 size blocks: 1509 aes-128-cbc's in 3.00s
OpenSSL 1.0.0b 16 Nov 2010
built on: Thu Jan 20 10:23:44 CST 2011
options:bn(64,32) rc4(ptr,int) des(idx,risc1,2,long) aes(partial) idea(int) blowfish(idx)
compiler: arm-none-linux-gnueabi-gcc -march=armv7-a -mtune=cortex-a8 -mfpu=neon -mfloat-abi=softfp -mthumb-interwork -mno-thumb -fPS
The 'numbers' are in 1000s of bytes per second processed.
type 16 bytes 64 bytes 256 bytes 1024 bytes 8192 bytes
aes-128-cbc 3733.37k 4001.19k 4089.34k 4112.73k 4120.58k
Command being timed: "openssl speed -evp aes-128-cbc"
User time (seconds): 15.03
System time (seconds): 0.00
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0m 15.07s
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 7216
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 1

```



```
Minor (reclaiming a frame) page faults: 484
Voluntary context switches: 13
Involuntary context switches: 35
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

Crypto Performance

- AM335x Crypto Performance using openssl ^[3]

Read AM3517 performance tests using cryptotest ^[4]

References

- [1] http://processors.wiki.ti.com/index.php/AMSDK_Linux_User%27s_Guide#Customizing_the_Configuration
- [2] http://processors.wiki.ti.com/index.php/AMSDK_Linux_User%27s_Guide
- [3] http://processors.wiki.ti.com/index.php/AM335x_Crypto_Performance
- [4] http://processors.wiki.ti.com/index.php/Build_Crypto_Module_for_Sitara#Test_the_Module

Oprofile User's Guide

Return to the [Sitara Linux Software Developer's Guide](#)



Supported Platforms

Oprofile is supported on **all** ARM devices

Oprofile Overview

- OProfile is a statistical continuous profiler. In other words, profiles are generated by regularly sampling the current registers on each CPU (from an interrupt handler, the saved PC value at the time of interrupt is stored), and converting that runtime PC value into something meaningful to the programmer.
- Regularly sampling the PC value like this approximates what actually was executed and how often - more often than not, this statistical approximation is good enough to reflect reality. In common operation, the time between each sample interrupt is regulated by a fixed number of clock cycles. This implies that the results will reflect where the CPU is spending the most time; this is obviously a very useful information source for performance analysis.
- Taken verbatim from <http://oprofile.sourceforge.net/doc/internals/introduction.html>

This is only a small set of examples to help show how Oprofile can be used. Anyone wishing more information should look here: <http://oprofile.sourceforge.net/>

Oprofile Limitations

Due to a bug in the Cortex-A8, only the internal timer mode is used for Oprofile. While Oprofile provides the capability to utilize ARM hardware performance counters, due to the bug, this feature has been disabled.

Oprofile Applications Overview

The sample applications are intended to show an example of how Oprofile may be used to discover where the CPU is spending the most time in an application. In this series of examples, a bottleneck is identified in a "signal_parent" application and better solution is implemented showing dramatic differences in CPU utilization in an optimized version of the same application.

There are 3 types of applications in this Oprofile example.

- Init - Initialize Oprofile to be able to use kernel debug information. This only needs to be done once.
- Profile session - run an application and collect profiling data.
- Reports - Run one of the variations of report generation.

The `signal_parent` application source code can be found in your SDK on your host at `$(SDK_HOME)/ti-sdk-amxxx-evm-xx.xx.xx.xx/example-applications/am-sysinfo-1.0/oprofile_example`. It is a single application which can be compiled two different ways. One where the parent uses polling or another where the parent uses SIGNALS. When building two executable are generated:

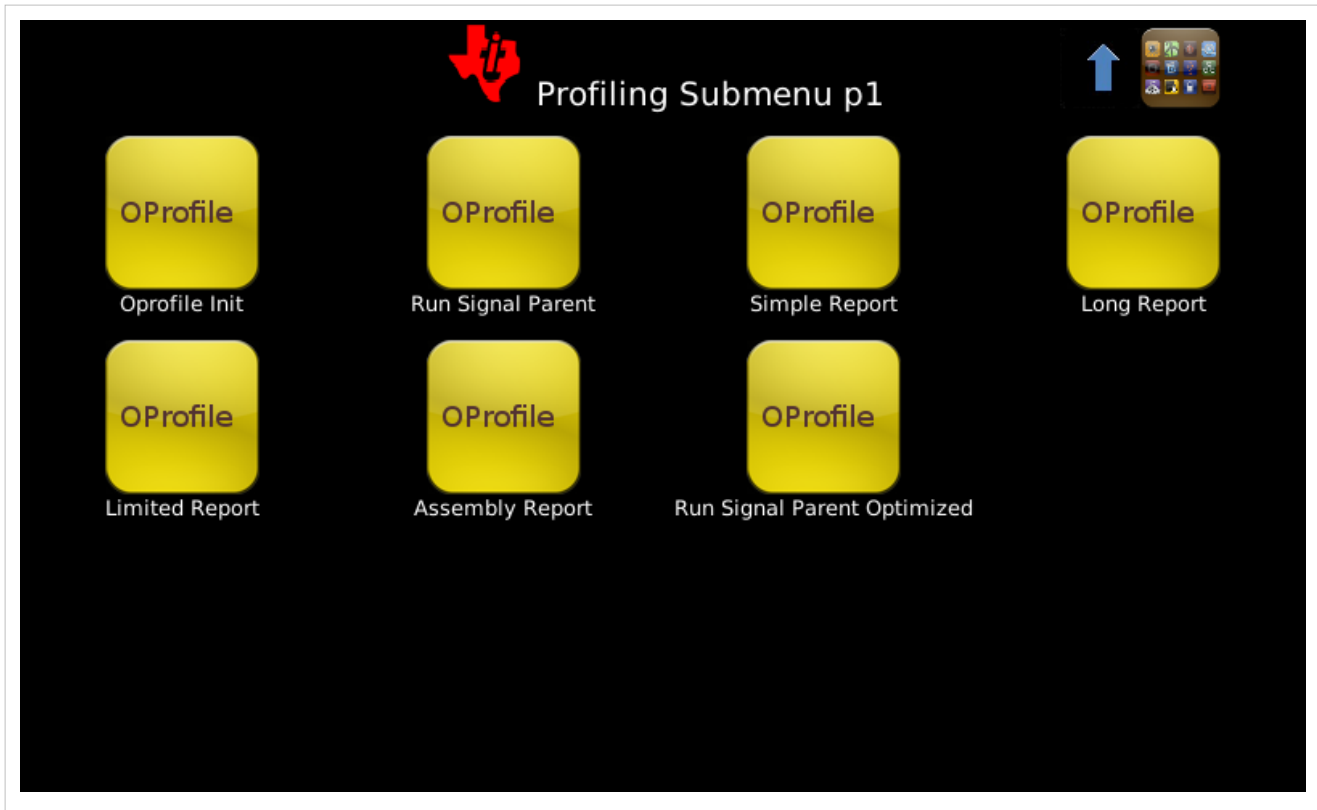
1. `Debug/signal_parent` - The polling version
2. `Debug1/signal_parent.opt` - The optimized version which uses SIGNALS rather than polling.

The scripts used to support these apps are found in your target file system at `/usr/bin/`

```
ls /usr/bin/runOp*
```

Initial Boot Up

When you first boot up a target system containing a Sitara Software Development Kit (SDK), Matrix should be automatically started. Select the Oprofile button to bring up all the Oprofile applications. Matrix Oprofile Applications Page:



Applications Detailed Information

- **Oprofile Init** - This command is required only the first time you run Oprofile or if you build and boot a new kernel. You need to point Oprofile to the vmlinux file that matches your kernel.
 - vmlinux is found by default in the /boot directory in your target file system. If you rebuild the kernel yourself, you should run this step again to point to the vmlinux from your build that corresponds to the kernel you are using
 - **If you use a vmlinux that was not built along with the kernel you are using, then the debug information will be incorrect.**
 - Actual commands

```
opcontrol --vmlinux=/boot/vmlinux*
```

- **Run Signal Parent** - Running this demo will profile a simple unoptimized application called "Signal Parent".
 - Running this application causes profile data to be stored internally. To view the data you must generate a report.
 - In this application a parent process forks a child process which does a little work and then signals the parent when it is complete.
 - The parent busy waits by constantly polling status waiting for the child to complete.
 - Actual commands:

```
opcontrol --reset
opcontrol --start
signal_parent
opcontrol --stop
```

- **Simple Report** This will generate a simple report based on the last application you profiled.
 - Notice below how signal parent takes up 86.2% of the total samples taken. Your numbers may vary depending platform differences.

```
Profiling through timer interrupt
      TIMER:0 |
samples|      %|
-----|-----|
      909 86.2429 signal_parent
      136 12.9032 vmlinux-3.2.0
         6  0.5693 ld-2.12.2.so
         1  0.0949 busybox
         1  0.0949 libc-2.12.2.so
         1  0.0949 thttpd
```

- **Actual command**

```
opreport
```

- **Run Signal Parent Optimized** - This is the same application as "Run Signal Parent", however after seeing how much time is wasted polling, it was redesigned. The parent now goes to sleep and waits for a signal from the child to complete.
 - After running this application, you can run the Simple Report again to see that now the "Run Signal Parent" application is only sampled 0.38% of the time.

```
Profiling through timer interrupt
      TIMER:0 |
samples|      %|
-----|-----|
     1035 98.7595 vmlinux-3.2.0
         4  0.3817 ld-2.12.2.so
         4  0.3817 signal_parent.opt
         2  0.1908 busybox
         2  0.1908 libc-2.12.2.so
         1  0.0954 lighttpd
```

- **Actual commands:**

```
opcontrol --reset
opcontrol --start
signal_parent.opt
opcontrol --stop
opreport
```

- **Additional reports** - There are additional reports available to show the flexibility of the report tool, however these variations are only a small subset of the capabilities of Oprofile.

Open Source Wireless Connectivity WLAN Bluetooth User Guide

Return to the [Sitara Linux Software Developer's Guide](#)



Supported Platforms

- AM335x EVM
- AM37x
- AM180x

WLAN Bluetooth User Guide

Production Updates

Before ramping to production, customers should review the Connectivity Wiki for the latest updates.

Wiki page address: http://processors.wiki.ti.com/index.php/ARM_Processor_Open_Source_Wireless_Connectivity^[1]

Board Bringup (The board bringup phase includes: hardware bringup, switches settings and system setup)

- AM18x/AM37x/AM335x - Board Bring Up^[2]

Demo - WLAN Station

- Step 1: Calibration (refer to: <Calibration Process>).
- Step 2: Setting unique MAC address (refer to: <MAC Change>).
- Step 3: Connect to non secured AP using WPA supplicant configuration file (refer to: <Connect to non secured AP using WPA Supplicant configuration file>).

Demo usecase - WLAN SoftAP

- Step 1: perform Calibration - (see WLAN Station section)
- Step 2: Define "no security" mode using hostAPD configuration file (refer to: <OMAP_Wireless_Connectivity_WLAN_AP_No_Security>).

Demo usecase - WiFi Direct

For WiFi Direct usecases refer to Scripts for WiFi Direct Configuration^[3]

Demo usecases - Bluetooth

- Bluetooth Demos using GUI (A2DP, HID, SPP)^[4].
 - Bluetooth Demo using command line and Windows PC Terminal^[5]
 - Bluetooth Demo using command line and Linux PC Terminal^[6]
-

Build Instructions Guide

- For WLAN build options refer to: <WLAN driver build options>
- For Bluetooth build options refer to: <Bluetooth build options>

References

- [1] http://processors.wiki.ti.com/index.php/ARM_Processor_Open_Source_Wireless_Connectivity
- [2] http://processors.wiki.ti.com/index.php/Open_Source_Wireless_Connectivity_baord_bring_up
- [3] http://processors.wiki.ti.com/index.php/OMAP_Wireless_Connectivity_NLCP_WiFi_Direct_Configuration_Scripts
- [4] http://processors.wiki.ti.com/index.php/OMAP_Wireless_Connectivity_Bluetooth_GUI
- [5] http://processors.wiki.ti.com/index.php/AM37x_Wireless_Connectivity_NLCP_Bluetooth_Terminal
- [6] http://processors.wiki.ti.com/index.php/AM37x_Wireless_Connectivity_NLCP_Bluetooth_Terminal_linux

OMAP Wireless Connectivity OpenSource Compat Wireless Build

Before starting

Before starting you need to complete the following steps:

- Setup build environment variables <Setting build environment>
- Make sure that you have the target linux kernel source extracted to your PC and built locally.

Download and Build the TI compat-wireless package

- Download the compat-wireless package from R3M1RC2 Compat wireless ^[1] location
- Extract the zip file to your PC
- Change to the compat-wireless-2.6 source directory
- Configure the build for the w12xx family:

```
./scripts/driver-select w12xx
```

you have a kernel you do not have installed but yet want to build the compat-wireless-2.6 drivers for it you can use this syntax:

- export the following variables:

KLIB: point to the location of your root file system (NFS_ROOT). See Setting build environment section

KLIB_BUILD: points to the location of your linux kernel source

for example:

```
export KLIB_BUILD=~/.oe/arago-tmp/work/am37x-evm-arago-linux-gnueabi/linux-omap3-2.6.37-psp04.02.00.07.sdk-r97/git/
export KLIB=${NFS_ROOT}
```

- Ensure that configuration of the kernel the w12xx driver is compiled with contains the following switches ^[2].
- Now build the compat-wireless package and install it to your root file system (pointed by NFS_ROOT)

```
make
make install-modules
```

You should now have all the relevant modules installed in your root file system under **`\${NFS_ROOT}/lib/modules/<kernel version>/updates/..`**

For more details regarding the concepts of compat-wireless please refer to the linux wireless ^[3] home page

Download and install the wl12xx Firmware Files

- Download the wl12xx firmware zip file from this ^[4] location
- Copy the firmware files to the EVM under the /lib/firmware/ti-connectivity directory in your root file system.

The files are:

```
LICENCE wl128x-fw-ap.bin wl1271-fw-2.bin wl128x-fw.bin  
wl1271-fw-ap.bin wl128x-fw-multirole-plt.bin wl1271-fw-multirole-plt.bin  
wl128x-fw-multirole-roc.bin wl1271-fw-multirole-roc.bin
```

Comment: some other files may exist, so copy all the fw files

- Copy the ini files to the EVM under the /lib/firmware/ti-connectivity/ini_files/ directory in your root file system.

The files are:

```
RFMD_D_E5.ini TQS_D_1.0.ini TQS_S_2.5.ini  
RFMD_S_3.5.ini TQS_D_1.7.ini TQS_S_2.6.ini
```

Comment: some other files may exist, so copy all the ini files



HOME

References

- [1] <https://gforge.ti.com/gf/download/frsrelease/555/4629/ti-compat-wireless-wl12xx-2011-05-17-r3-m1-rc2.tgz>
- [2] http://processors.wiki.ti.com/index.php/OMAP_Wireless_Connectivity_OpenSource_Kernel_Switches
- [3] <http://linuxwireless.org/en/users/Download>
- [4] <https://gforge.ti.com/gf/download/frsrelease/557/4631/ti-linux-firmware-wl12xx-r3-m1-rc2.tar.gz>

OMAP Wireless Connectivity mac80211 compat wireless implementation Architecture

Scope

This section will explain the Compat Wireless Generation process and architecture.

Introduction

compat-wireless backports both the bluetooth and 802.11 subsystems down to older kernels. To be able to synchronize backporting the latest and greatest the linux-next.git tree is used as its main source for kernel updates. General Linux kernel compatibility is addressed through a general kernel compatibility tree, compat.git. compat-wireless then has its own tree for specific wireless compatibility.

Building Instructions

You will then need to checkout three trees to start hacking on compat-wireless:

```
git://github.com/TI-OpenLink/wl12xx.git
git://github.com/TI-OpenLink/compat.git
git://github.com/TI-OpenLink/compat-wireless.git
```

Clone the gits:

```
git clone git://github.com/TI-OpenLink/wl12xx.git
git clone git://github.com/TI-OpenLink/compat.git
git clone git://github.com/TI-OpenLink/compat-wireless.git
```

To update a git, get inside the directory that includes the git and type:

```
git fetch origin
```

To point to specific commit_ID (for wl12xx.git only):

```
git reset --hard commit_ID
```

wl12xx.git

The wl12xx.git tree brings all subsystems being worked on for the next kernel release into one tree. This means wl12xx will have what people today are working on for the 3.x kernel release.

compat.git

The compat git tree is a general kernel compatibility layer which can be shared among different compatibility projects, or drivers. compat-wireless is just one of the kernel compatibility projects using compat.git. compat.git builds a general compatibility module, compat, and any additional modules to let you get new general kernel updates from future kernels on your old kernels.

compat-wireless.git

Anything that is not general kernel compatibility but instead specific to 802.11 or bluetooth goes into compat-wireless.git

Now that we have the git trees, we need to point to them by exporting the following variables:

```
export GIT_TREE=/home/user/Projects/MAC80211/wl12xx
export GIT_COMPAT_TREE=/home/user/Projects/MAC80211/compat
```

Where:

- The GIT_TREE variable points to the wl12xx git previously downloaded.
- The GIT_COMPAT_TREE points to the compat git previously downloaded.

Now we need to run scripts/admin-refresh.sh in order to apply the compat-wireless-2.6 patches and update the Compat accordingly.

```
cd /home/user/Projects/MAC80211/compat-wireless
```

```
scripts/admin-refresh.sh
```

In general, you can update your local sources based on these wl12xx.git and compat.git trees:


```
scripts/admin-clean.sh - Cleans the compat-wireless tree
scripts/admin-update.sh - Updates compat-wireless with your git tree
scripts/admin-refresh.sh - Does the above two
```

Now we need to select the wl12xx. Switch to directory compat-wireless-2.6 and invoke:

```
scripts/driver-select wl12xx
```

Then, in order to compile the compat-wireless package you need to invoke:

```
make KLIB_BUILD=<your linux kernel version> KLIB=<your linux kernel version>
```

Deploy to rootfs

In order to install the Compat on the EVM you need to pick up the following kernel objects:

```
compat.ko - located under (Compat-Wireless)/compat/
cfg80211.ko - located under (Compat-Wireless)/net/wireless/
mac80211.ko - located under (Compat-Wireless)/net/mac80211/
wl12xx.ko - located under (Compat-Wireless)/drivers/net/wireless/wl12xx
wl12xx_sdio.ko - located under (Compat-Wireless)/drivers/net/wireless/wl12xx
```

Those files should be copied to the rootfs in the following directories:

```
compat.ko - located under /lib/modules/2.6.37/updates/compat
cfg80211.ko - located under /lib/modules/2.6.37/updates/net/wireless
mac80211.ko - located under /lib/modules/2.6.37/updates/net/mac80211
wl12xx.ko - located under /lib/modules/2.6.37/updates/drivers/net/wireless/wl12xx
wl12xx_sdio.ko - located under /lib/modules/2.6.37/updates/drivers/net/wireless/wl12xx
```

Now you need to reboot the EVM in order to run those kernel objects.

OMAP Wireless Connectivity Battleship Game demo

Purpose

The Battleship Game demonstrates WiFi-Direct capability.

The WiFi-Direct technology is used in order to search for other players and to establish wireless connections in ad-hoc style.

How to Play

Battleship is a well-known 2 player game where the objective is to sink all of your enemy ships.

On the start of the game players set up the positions for each of their ships, and then they take turns guessing squares on their opponent's grid and seeing if those squares are occupied (a 'hit') or not (a 'miss').

The first player to destroy all of his opponents ships wins the game.

For more details please follow: <Battleship Game description> ^[1]

Playing the Game

Setting the MAC Address

To work properly, each EVM must have a unique MAC address. Refer to the OMAP_Wireless_Connectivity_Station_MAC_Change for instructions.

Configuring Device Name

First thing we want to do is to configure the device name for the Wi-Fi direct.

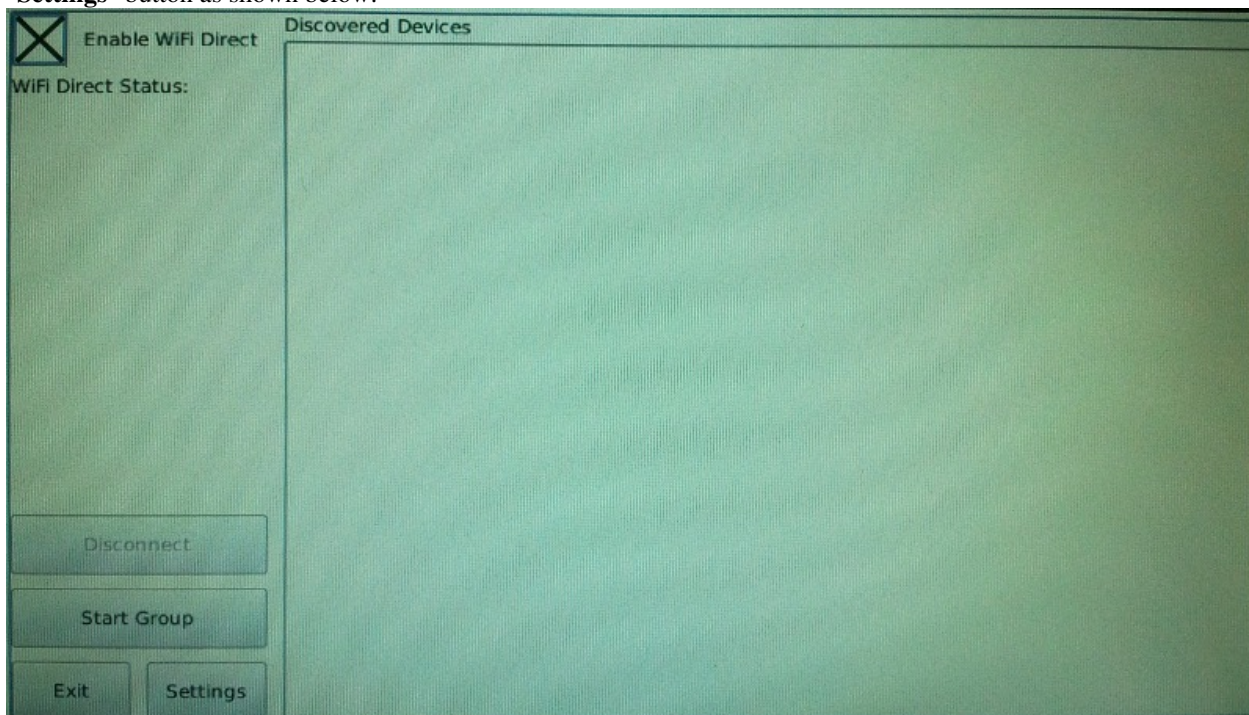
For this, you need to choose the "WiFi Direct" Icon as shown below in the second row at the right:



Figure 1 -

WiFi Direct Icon

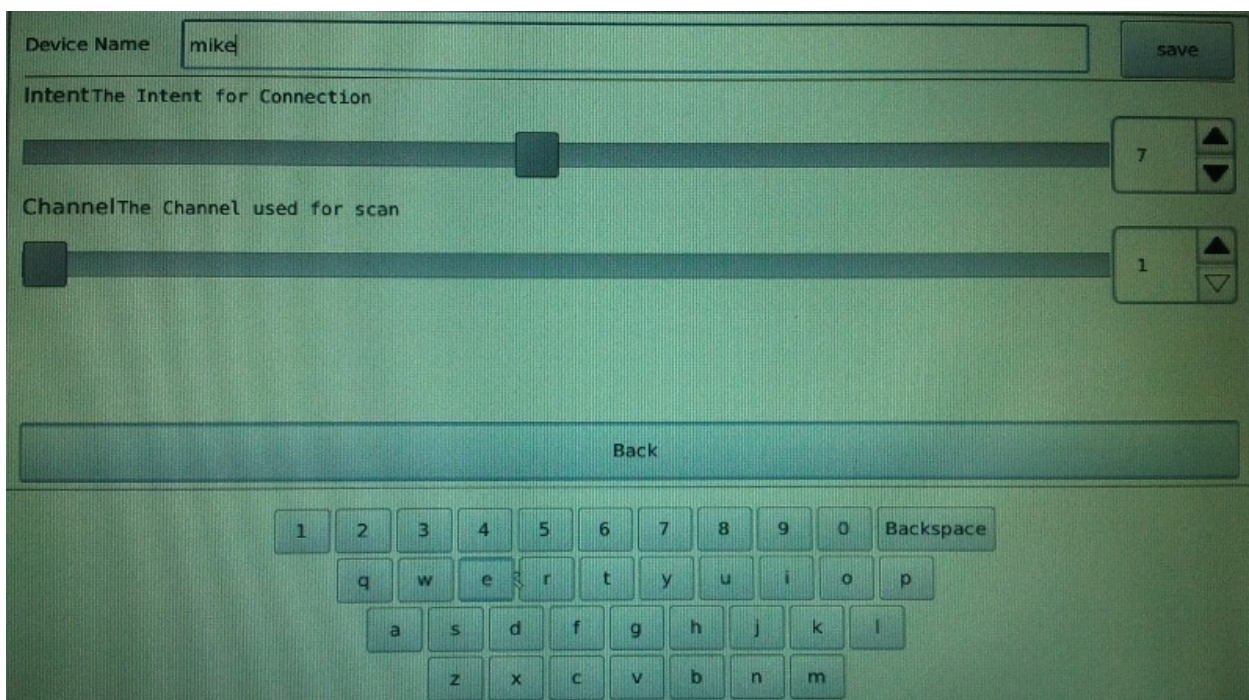
On the WiFi-Direct main screen, check the top-left check-box called "Enable WiFi Direct". Then, click on "Settings" button as shown below:



Figure

2 - WiFi-Direct Main Screen

Tap on the "Device Name" editor and a virtual screen will appear so you can type the desired device name:



Figure

3 - Setting Device Name

Once you choose the name, click on the "**Save**" button, followed by "**Back**" button, then, click "**Exit**" to quit the WiFi-Direct application.

Now, the device name is configured and we are ready to play the game.

NOTE that if you don't do this step, your device name will be set to default value which is the machine type, for example, "am37x-evm".

Starting Battleship Game

To start the Battleship game, tap on the icon called "Battleship" shown below:



Figure 4 -

Battleship Icon

Once the game starts, you should see the main screen:



Figure

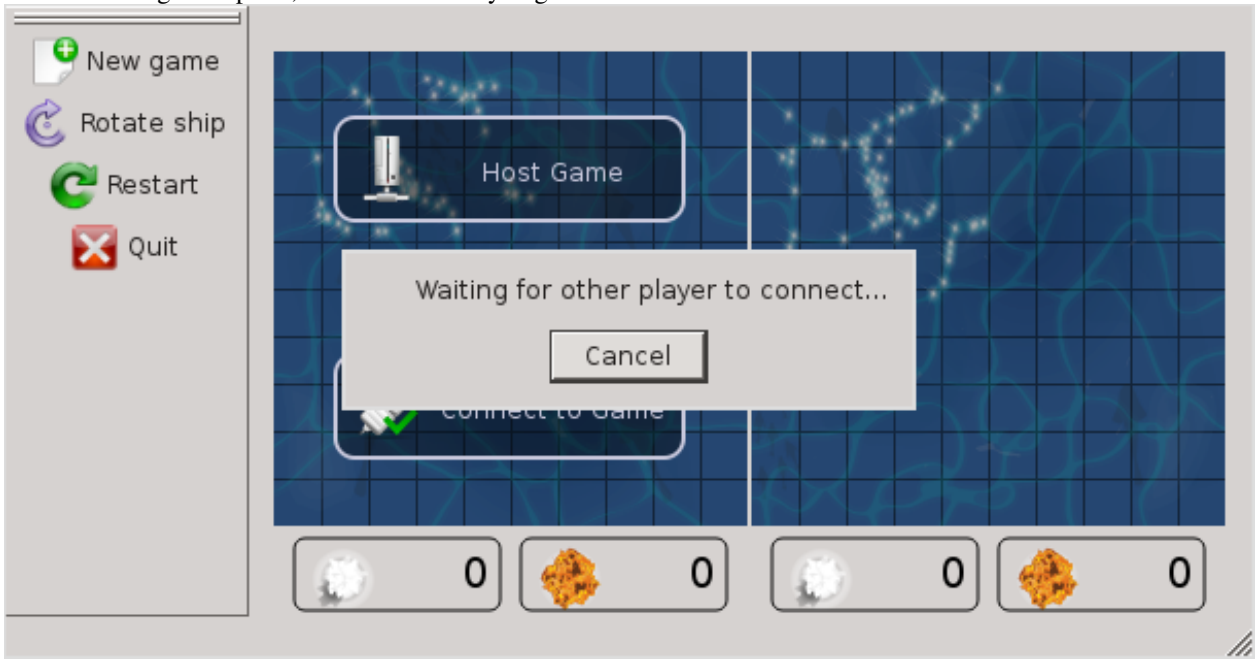
5 - Main Screen

Here, you have two choices:

1. Host the battleship game
2. Connect to a game (hosted by another EVM)

Hosting Game

If you choose the option "**Host Game**", the EVM will be the server and wait for other EVM to connect to it. When choosing this option, this is the screen you get:

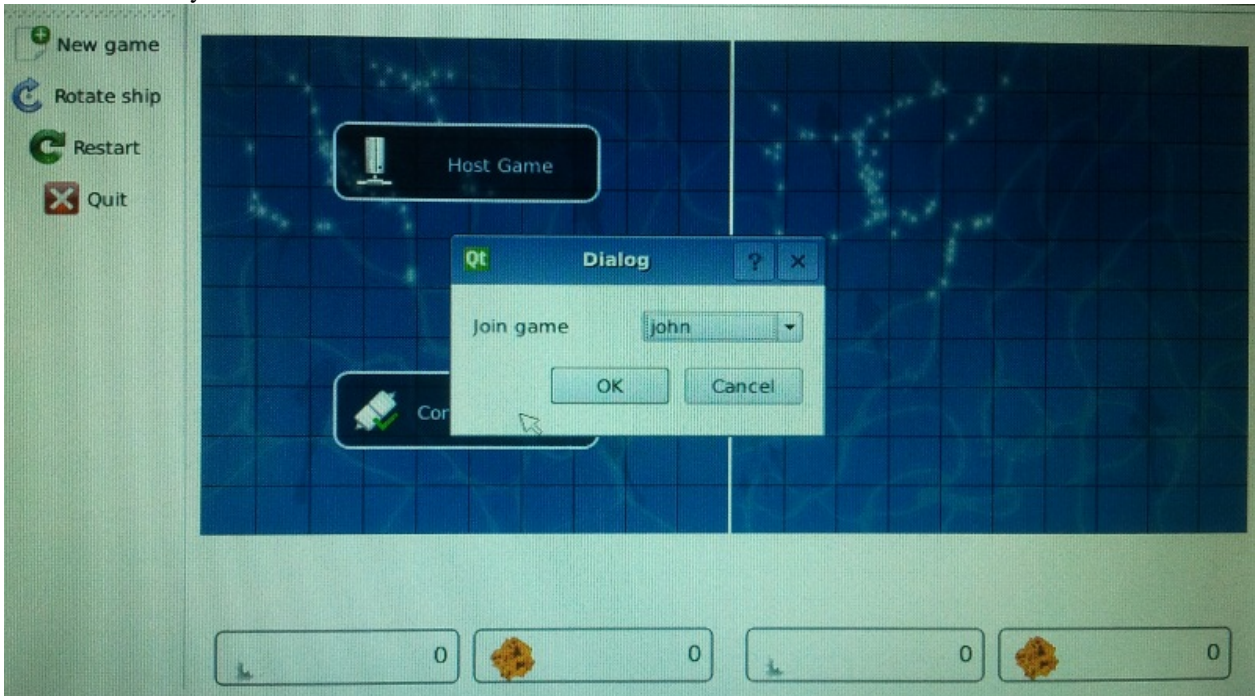


Figure

6 - Hosting Game

Connecting Game

If you choose the second option, which is called "**Connect to Game**" you will have a dialog window where you can choose the EVM you want to connect to:



Figure

7 - Connecting Game Dialog

The dialog box will show all available WiFi-Direct devices in range. Choose the device you want to connect to, and the game should start in a few seconds.

Battleship Game SD Card images

- AM37x
 - SD card image: Root Partition (am37x-dbus-rootfs_battleship_game_march_28_2012.tgz) ^[2]
 - SD card image: Boot Partition (am37x-dbus-boofs_battleship_game_march_28_2012.tgz) ^[3]
- AM18x
 - SD card image: Root Partition (am180x-root-battleship_game_march_28_2012.tgz) ^[4]
 - SD card image: Boot Partition (am180x-boot-battleship_game_march_28_2012.tgz) ^[5]
- AM335x
 - TBD

References

[1] [http://en.wikipedia.org/wiki/Battleship_\(game\)](http://en.wikipedia.org/wiki/Battleship_(game))

[2] https://gforge.ti.com/gf/download/frsrelease/830/5504/am37x-dbus-rootfs_battleship_game_march_28_2012.tgz

[3] https://gforge.ti.com/gf/download/frsrelease/831/5505/am37x-dbus-boofs_battleship_game_march_28_2012.tgz

[4] https://gforge.ti.com/gf/download/frsrelease/833/5507/am180x-root-battleship_game_march_28_2012.tgz

[5] https://gforge.ti.com/gf/download/frsrelease/832/5506/am180x-boot-battleship_game_march_28_2012.tgz

AMSDK u-boot User's Guide

Return to the [Sitara Linux Software Developer's Guide](#)



Overview

This document covers the general use of U-Boot v2012.10 and the AMSDK on following platforms:

- am335x EVM
- am335x EVM-SK
- BeagleBone

For am3517 EVM, am37x EVM and Beagleboard xM support please see the [AMSDK 5.05.00 documentation](#) ^[1].

For am180x information please see [this page](#).

The latest release has been validated in the following hardware configurations. If your hardware platform is not listed below you can either use the release provided in the SDK ^[13] for your device or try this u-boot release and provide feedback ^[14] on issues encountered so they can be addressed in a future release.

Board	Wired ethernet	USB gadget ethernet	NAND	SD/MMC	USB Host (mass storage)	SPI flash
AM335x EVM	yes	yes	yes	yes	yes	yes
AM335x EVM-SK	yes	yes	N/A	yes	yes	N/A
Beaglebone	yes	N/A	N/A	yes	yes	N/A

We assume that a GCC-based toolchain has already been installed and the serial port for the board has been configured. If this is not the case, please refer back to the Sitara Linux Software Developer's Guide. We also assume that a Linux Kernel has already been built (or has been provided) as well as an appropriate filesystem image. Installing and setting up DHCP or TFTP servers is also outside of the scope of this document, but snippets of information are provided to show how to use a specific feature, when needed.

Finally, please note that not all boards have all of the interfaces documented here. For example, the BeagleBoard xM and BeagleBone do not have NAND.

General Information

Building MLO and u-boot

We strongly recommend the use of separate object directories when building. This is done with `O=` parameter to `make`. We also recommend that you use an output directory name that is identical to the `make` target name. That way if you are working with multiple `make` targets it is very easy to know which folder contains the u-boot binaries that you are interested in.

Cleaning the Sources

If you did not use a separate object directory:

```
$ make CROSS_COMPILE=arm-arago-linux-gnueabi- ARCH=arm distclean
```

If you used 'O=am335x_evm' as your object directory:

```
$ rm -rf ./am335x_evm
```

Compiling MLO and u-boot

Building of both u-boot and SPL is done with a single `make` command. The `make` target depends on the board in question:

Board	make target
AM335x EVM, AM335x EVM-SK, Beaglebone	am335x_evm

```
$ make CROSS_COMPILE=arm-arago-linux-gnueabi- ARCH=arm O=make_target_from_table_above make_target_from_table_above
```


U-Boot Environment

Restoring defaults

It is possible to reset the set of U-Boot environment variables to their defaults and if desired, save them to where the environment is stored, if applicable. To do so, issue the following commands:

```
U-Boot # env default -f -a
U-Boot # saveenv
```

Networking Environment

When using a USB-Ethernet dongle a valid MAC address must be set in the environment. To create a valid address please read [this page](#)^[2]. Then issue the following command:

```
U-Boot # setenv usbethaddr value:from:link:above
```

Then start the USB subsystem:

```
U-Boot # usb start
```

The default behavior of U-Boot is to utilize all information that a DHCP server passes to us when the user issues the **dhcp** command. This will include the dhcp parameter *next-server* which indicates where to fetch files from via TFTP. There may be times however where the dhcp server on your network provides incorrect information and you are unable to modify the server. In this case the following steps can be helpful:

```
U-Boot # setenv autoload no
U-Boot # dhcp
U-Boot # setenv serverip correct.server.ip
U-Boot # tftp
```

Another alternative is to utilize the full syntax of the tftp command:

```
U-Boot # setenv autoload no
U-Boot # dhcp
U-Boot # tftp ${loadaddr} server.ip:fileName
```

Working with USB Device Firmware Upgrade

When working with USB Device Firmware Upgrade (DFU), regardless of the medium to be written to and of the board being used, there are some general things to keep in mind. First of all, you will need to get a copy of the **dfu-util** program installed on your host. If your distribution does not provide this package you will need to build it from source. Second, the examples that follow assume a single board is plugged into the host PC. If you have more than one device plugged in you will need to use the options that **dfu-util** provides for specifying a single device to work with. Finally, while one could build a U-Boot that supports both DFU writing to NAND and DFU writing to SD or eMMC, it would result in a more complicated environment to work from, so this has been avoided.

Using the network (Wired or USB)

This section documents how to configure the network and use it to load files and then boot the Linux Kernel using a root filesystem mounted over NFS. At this time, no special builds of U-Boot are required to perform these operations on the supported hardware.

Booting U-Boot from the network

In some cases we support loading SPL and U-Boot over the network because of ROM support. In some cases, a special build of U-Boot may be required. In addition, the DHCP server is needed to reply to the target with the file to fetch via tftp. In order to facilitate this, the **vendor-class-identifier** DHCP field is filled out by the ROM and the values are listed in the table below. Finally, you will need to use the **spl/u-boot-spl.bin** and **u-boot.img** files to boot.

Board	make target	Supported interfaces	ROM vendor-class-identifier value	SPL vendor-class-identifier value
AM335x EVM	am335x_evm	CPSW ethernet	DM814x ROM (PG1.0) or AM335x ROM (PG2.0 and later)	AM335x U-Boot SPL
AM335x EVM (PG2.0 and later)	am335x_evm_usbspl	SPL and U-Boot via USB RNDIS	N/A	AM335x U-Boot SPL
AM335x EVM (PG1.0)	am335x_evm_uart_usbspl	SPL via UART, U-Boot via USB RNDIS	AM335x ROM	AM335x U-Boot SPL

If using ISC dhcpd an example host entry would look like this:

```
host am335x_evm {
    hardware ethernet de:ad:be:ee:ee:ef;
    # Check for PG1.0, typically CPSW
    if substring (option vendor-class-identifier, 0, 10) = "DM814x ROM" {
        filename "u-boot-spl.bin";
    }
    # Check for PG2.0, CPSW or USB RNDIS
    } elsif substring (option vendor-class-identifier, 0, 10) = "AM335x ROM" {
        filename "u-boot-spl.bin";
    }
    } elsif substring (option vendor-class-identifier, 0, 17) = "AM335x U-Boot SPL" {
        filename "u-boot.img";
    }
    } else {
        filename "uImage-am335x-evm.bin";
    }
}
```

Note that in a factory type setting, the substring tests can be done inside of the subnet declaration to set the default filename value for the subnet, and overridden (if needed) in a host entry.

If you have removed NetworkManager from your system (which is not the default in most distributions) you need to configure your `/etc/network/interfaces` file thusly:

```
allow-hotplug usb0
iface usb0 inet static
    address 192.168.1.1
    netmask 255.255.255.0
    post-up service isc-dhcp-server reload
```

If you are using NetworkManager you need to create two files. First, as root create `/etc/NetworkManager/system-connections/AM335x USB RNDIS` (and use `\` to escape the space) with the following content:

```
[802-3-ethernet]
duplex=full
mac-address=AA:BB:CC:11:22:33

[connection]
id=AM335X USB RNDIS
uuid=INSERT THE CONTENTS OF 'uuidgen' HERE
type=802-3-ethernet

[ipv6]
method=ignore

[ipv4]
method=manual
addresses1=192.168.1.1;16;
```

Second as root, and ensuring execute permissions, create `/etc/NetworkManager/dispatcher.d/99am335x-dhcp-server`

```
#!/bin/sh

IF=$1
STATUS=$2

if [ "$IF" = "usb0" ] && [ "$STATUS" = "up" ]; then
    service isc-dhcp-server reload
fi
```

Multiple Interfaces

On some boards, for example when we have both a wired interface and USB RNDIS gadget ethernet, it can be desirable to change from the default U-Boot behavior of cycling over each interface it knows to telling U-Boot to use a single interface. For example, on start you may see lines like:

```
Net:    cpsw, usb_ether
```

So to ensure that we use **usb_ether** first issue the following command:

```
U-Boot # setenv ethact usb_ether
```

Network configuration via DHCP

To configure the network via DHCP, use the following commands:

```
U-Boot # setenv autoload no
U-Boot # dhcp
```

And ensure that a DHCP server is configured to serve addresses for the network you are connected to.

Manual network configuration

To configure the network manually, the **ipaddr**, **serverip**, **gatewayip** and **netmask**:

```
U-Boot # setenv ipaddr 192.168.1.2
U-Boot # setenv serverip 192.168.1.1
U-Boot # setenv gatewayip 192.168.1.1
U-Boot # setenv netmask 255.255.255.0
```

Booting Linux from the network

Within the default environment for each board that supports networking there is a boot command called **netboot** that will automatically load the kernel and boot. For the exact details of each use **printenv** on the **netboot** variable and then in turn **printenv** other sub-sections of the command. The most important variables here are **rootpath** and **nfsopts**.

Automatically programming flash from a network boot

Given the ability to load and execute software via network interfaces in the ROMs of some platform, one usage of this functionality is custom build of U-Boot that has a default boot command that will write a payload to flash. This is done by a combination of building a special purpose U-Boot (see the table below) and writing a U-Boot script (a series of commands in a text file that we pass to `mkimage`).

Board	make target	Supported interfaces
AM335x EVM	am335x_evm_restore_flash	CPSW ethernet
AM335x EVM (PG2.0 and later)	am335x_evm_restore_flash_ubspl	SPL and U-Boot via USB RNDIS
AM335x EVM (PG1.0)	am335x_evm_restore_flash_uart_ubspl	SPL via UART, U-Boot via USB RNDIS

An example script file, which we call **debrick.txt** that will download files via tftp and then write them to NAND looks like:

```
# erase NAND
nand erase.chip
# get MLO
tftp 0x81000000 MLO
# make 4 copies in the RAM
cp.b 0x81000000 0x81020000 20000
cp.b 0x81000000 0x81040000 20000
cp.b 0x81000000 0x81060000 20000
# get u-boot.img
tftp 0x81080000 u-boot.img
# write the whole block (4*MLO + u-boot.img) to NAND
nand write 0x81000000 0x0 0x260000
```

We then need to run that file through **mkimage**:

```
mkimage -A arm -O U-Boot -C none -T script -d debrick.txt debrick.scr
```

Then **debrick.scr**, along with **MLO** and **u-boot.img** go in to the top directory that your tftp server uses to serve files from. Along with having configured the dhcp server as discussed above to boot the target machine it will now boot and program the NAND flash.

Using NAND

This section documents how to write files to the NAND device and use it to load and then boot the Linux Kernel using a root filesystem also found on NAND. At this time, no special builds of U-Boot are required to perform these operations on the supported hardware. Finally, for simplicity we assume the files are being loaded from an SD card. Using the network interface (if applicable) is documented above.

Writing to NAND from U-Boot

Note:

- From the U-Boot build, the **MLO** and **u-boot.img** files are the ones to be written.
- We load all files from an SD card in this example but they can just as easily be loaded via network (documented above) or other interface that exists.
- The default layout of the NAND that each supported device supports is the same for MLO and U-Boot which is why all of the nand erase and write locations are the same for that section.
- This series of commands will write **MLO** to the default location and then the backup locations that the ROM will look in as well.

```
U-Boot # mmc rescan
U-Boot # nand erase 0x0 0x780000
U-Boot # fatload mmc 0 0x81000000 MLO
U-Boot # cp.b 0x81000000 0x81020000 20000
U-Boot # cp.b 0x81000000 0x81040000 20000
U-Boot # cp.b 0x81000000 0x81060000 20000
U-Boot # fatload mmc 0 0x81080000 u-boot.img
U-Boot # fatload mmc 0 0x81280000 uImage
U-Boot # nand write 0x81000000 0x0 0x780000
U-Boot # saveenv
```

Writing to NAND via DFU

Currently in boards that support using DFU, the default build supports writing to NAND, so no custom build is required. To see the list of available places to write to (in DFU terms, altsettings) use the **mtdparts** command to list the known MTD partitions and **printenv dfu_alt_settings** to see how they are mapped and exposed to **dfu-util**.

```
U-Boot# mtdparts

device nand0 <nand0>, # parts = 5

#: name                size                offset              mask_flags
0: SPL                  0x00020000         0x00000000         0
1: SPL.backup1         0x00020000         0x00020000         0
2: SPL.backup2         0x00020000         0x00040000         0
3: SPL.backup3         0x00020000         0x00060000         0
```

```

4: u-boot          0x001e0000      0x00080000      0
5: u-boot-env     0x00020000      0x00260000      0
6: kernel        0x00500000      0x00280000      0
7: rootfs        0x0f880000      0x00780000      0

active partition: nand0,0 - (SPL) 0x00080000 @ 0x00000000

U-Boot# printenv dfu_alt_info

dfu_alt_info=SPL part 0 1;SPL.backup1 part 0 2;SPL.backup2 part 0 3;SPL.backup3 part 0 4;u-boot part 0 5;kernel part 0 7;rootfs part 0 8

```

This means that you can tell dfu-util to write anything to any of:

- SPL
- SPL.backup1
- SPL.backup2
- SPL.backup3
- u-boot
- kernel
- rootfs

Before writing you must erase at least the area to be written to. Then to start DFU on the target on the first NAND device:

```

U-Boot # nand erase.chip
U-Boot # dfu nand 0

```

Then on the host PC to write **MLO** to the first SPL partition:

```
$ sudo dfu-util -D MLO -a SPL
```

Finally, if you do:

```
U-Boot # setenv nandsilent true
```

prior to starting dfu in U-Boot, a number of messages will be suppressed and a slightly faster write will occur.

Booting from NAND

Within the default environment for each board that supports nand there is a boot command called **nandboot** that will automatically load the kernel and boot. For the exact details of each use **printenv** on the **nandboot** variable and then in turn **printenv** other sub-sections of the command. The most important variables here are **nandroot** and **nandrootfstype**.

Using removable media

Using removable media such as SD/MMC cards or a USB flash drive (or SD card in a USB card reader) use very similar techniques. The biggest difference is that only SD/MMC can be used by the ROM at power-on. Once U-Boot is running however, a USB device can be used for the kernel and the root filesystem.

Updating an SD card from a host PC

This section assume that you have created an SD card following the instructions on Sitara Linux SDK create SD card script or have made a compatible layout by hand. In this case, you will need to copy the **MLO**, **u-boot.img** and **uImage** files to the *boot* partition. At this point, the card is now bootable in the SD card slot.

Updating an SD card or eMMC using DFU

Currently in boards that support using DFU, the default build do not support writing to SD card or eMMC, so a custom build is required. Please see the table below for how to build a binary that does DFU writing to MMC. To see the list of available places to write to (in DFU terms, altsettings) use the **mmc part** command to list the partitions on the MMC device and **printenv dfu_alt_settings** to see how they are mapped and exposed to **dfu-util**.

Board	Build target
AM335x EVM, EVM-SK and Beaglebone	am335x_evm_dfu_mmc

```
U-Boot# mmc part
```

```
Partition Map for MMC device 0 -- Partition Type: DOS
```

```
Partition      Start Sector      Num Sectors      Type
  1              63                144522           c Boot
  2             160650         1847475          83
  3             2024190         1815345          83
```

```
U-Boot# printenv dfu_alt_info
```

```
dfu_alt_info=boot part 0 1;rootfs part 0 2;MLO fat 0 1;u-boot.img fat 0 1;uEnv.txt fat 0 1"
```

This means that you can tell dfu-util to write anything to any of:

- boot
- rootfs

And that the **MLO**, **u-boot.img** and **uEnv.txt** files are to be written to a FAT filesystem.

To start DFU on the target on the first MMC device:

```
U-Boot # dfu mmc 0
```

Then on the host PC to write **MLO** to the boot partition:

```
$ sudo dfu-util -D MLO -a boot
```

Finally, if you do:

```
U-Boot # setenv mmcsilent true
```

prior to starting dfu in U-Boot, a number of messages will be suppressed and a slightly faster write will occur.

Booting Linux from SD card

Within the default environment for each board that supports SD/MMC there is a boot command called **mmcboot** that will set the boot arguments correctly and start the kernel. In this case however, you must first run **loaduimagefat** or **loaduimage** to first load the kernel into memory. For the exact details of each use **printenv** on the **mmcboot**, **loaduimagefat** and **loaduimage** variables and then in turn **printenv** other sub-sections of the command. The most important variables here are **mmcrout** and **mmcroofstype**.

Booting Linux from USB storage

To load the Linux Kernel and rootfs from USB rather than SD/MMC card, if we assume that the USB device is partitioned the same way as an SD/MMC card is, we can utilize the **mmcboot** command to boot. To do this, perform the following steps:

```
U-Boot # usb start
U-Boot # setenv mmcroot /dev/sda2 ro
U-Boot # fatload usb 0 ${kloadaddr} ${bootfile}
U-Boot # run mmcboot
```

Using SPI

This section documents how to write files to the SPI device and use it to load and then boot the Linux Kernel using a root filesystem also found on SPI. At this time, no special builds of U-Boot are required to perform these operations on the supported hardware. The table below however, lists builds that will also use the SPI flash for the environment instead of the default, which typically is NAND. Finally, for simplicity we assume the files are being loaded from an SD card. Using the network interface (if applicable) is documented above.

Writing to SPI from U-Boot

Note:

- From the U-Boot build, the **MLO.byteswap** and **u-boot.img** files are the ones to be written.
- We load all files from an SD card in this example but they can just as easily be loaded via network (documented above) or other interface that exists.

Board	Build target
AM335x EVM	am335x_evm_spiboot

```
U-Boot # mmc rescan
U-Boot # sf probe 0
U-Boot # sf erase 0 +80000
U-Boot # fatload mmc 0 ${loadaddr} MLO.byteswap
U-Boot # sf write ${loadaddr} 0 ${filesize}
U-Boot # fatload mmc 0 ${loadaddr} u-boot.img
U-Boot # sf write ${loadaddr} 0x20000 ${filesize}
U-Boot # sf erase 80000 +${spiimgsize}
U-Boot # fatload mmc 0 ${loadaddr} uImage
U-Boot # sf write ${loadaddr} ${spisrcaddr} ${filesize}
```


Booting from SPI

Within the default environment for each board that supports SPI there is a boot command called **spiboot** that will automatically load the kernel and boot. For the exact details of each use **printenv** on the **spiboot** variable and then in turn **printenv** other sub-sections of the command. The most important variables here are **spiroot** and **spirootfstype**.

Archived Versions

- [5.03.03](#) ^[3]
- [5.03.00](#) ^[4]
- [5.05.00](#) ^[1]

Additional Information

- AM335x
 - [AM335x Flash Programming Guide](#)
 - [AM335x u-boot User's Guide](#)

References

- [1] http://processors.wiki.ti.com/index.php?title=AMSDK_u-boot_User%27s_Guide&oldid=109080
[2] <http://www.denx.de/wiki/view/DULG/WhereCanIGetAValidMACAddress>
[3] http://processors.wiki.ti.com/index.php?title=AMSDK_u-boot_User%27s_Guide&oldid=88139
[4] http://processors.wiki.ti.com/index.php?title=AMSDK_u-boot_User%27s_Guide&oldid=83941

AMSDK Linux User's Guide

Return to the [Sitara Linux Software Developer's Guide](#)



Overview

This article will cover the basic steps for building the Linux kernel and driver modules for TI Sitara devices.

Preparing to Build

In order to build the Linux kernel you will need a cross compiler installed on your system which can generate object code for the ARM core in your Sitara device. In the case of the AMSDK this compiler can be found inside of the SDK in the `<sdk install dir>/linux-devkit/bin` directory. If you have not already done so you should add this compiler to your path by doing:

```
export PATH="<sdk install dir>/linux-devkit/bin:$PATH"
```

Where `<sdk install dir>` should be replaced with the directory where the SDK was installed.

It is important that when using the GCC toolchain provided with the SDK or stand alone from TI that you do **NOT** source the `environment-setup` file included with the toolchain when building the kernel. Doing so will cause the compilation of host side components within the kernel tree to fail.

The following commands are intended to be run from the root of the kernel tree unless otherwise specified. The root of the kernel tree is the top-level directory and can be identified by looking for the "MAINTAINERS" file.

Cleaning the Kernel Sources

Prior to compiling the Linux kernel it is often a good idea to make sure that the kernel sources are clean and that there are no remnants left over from a previous build.

NOTE: The next step will delete any saved .config file in the kernel tree as well as the generated object files. If you have done a previous configuration and do not wish to lose your configuration file you should save a copy of the configuration file before proceeding.

The command to clean the kernel is:

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- mrproper
```

For user's who are building the kernel from within the AMSDK you can also use the Makefile at the root of the SDK installation at <sdk install dir>/Makefile to clean the kernel by doing:

```
cd <sdk install dir>
make linux_clean
```

Configuring the Kernel

Before compiling the Linux kernel it needs to be configured to select what components will become part of the kernel image, which components will be build as dynamic modules, and which components will be left out all together. This is done using the Linux kernel configuration system.

Using Default Configurations

It is often easiest to start with a base default configuration and then customize it for you use case if needed. In the Linux kernel a command of the form:

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- <config>
```

will look in the arch/arm/configs directory for the configuration file (<config> in the line above) use that file to set default configuration options. The table below shows the default configuration files for various platforms.

NOTE: The configuration file used to build the pre-built binaries found in the SDK can be found in the arch/arm/configs directory as tisdk_<device>_config and can be used in place of the config file given below to reproduce the SDK Linux kernel configuration settings. The differences between these files generally revolve around enabling/disabling additional modules for expected SDK use cases.

Device	SDK config	PSP config
AM335x/Beaglebone	tisdk_am335x-evm_defconfig	am335x_evm_defconfig
AM37x	tisdk_am37x-evm_defconfig	omap3_evm_defconfig
AM3517	tisdk_am3517-evm_defconfig	am3517_evm_defconfig
Beagleboard	tisdk_beagleboard_defconfig	omap3_beagle_defconfig
AM180x	tisdk_am180x-evm_defconfig	da850_omap138_defconfig

For example, to build the default PSP configuration for the AM335x the command would be:

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- am335x_evm_defconfig
```

To build the SDK configuration for the AM335x the command would be:

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- tisdsk_am335x-evm_defconfig
```

After the configuration step has run the full configuration file is saved to the root of the kernel tree as `.config`. Any further configuration changes are based on this file until it is cleanup up by doing a kernel clean as mentioned above.

Customizing the Configuration

When you want to customize the kernel configuration the easiest way is to use the built in kernel configuration systems. Two of the most popular configuration systems are:

menuconfig: an ncurses based configuration utility xconfig: a Qt based graphical configuration utility.

NOTE: on some systems in order to use xconfig you may need to install the `libqt3-mt-dev` package. For example on Ubuntu 10.04 this can be done using the command `sudo apt-get install libqt3-mt-dev`

To invoke the kernel configuration you simply use a command like:

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- <config type>
```

i.e. for menuconfig the command would look like

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- menuconfig
```

whereas for xconfig the command would look like

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- xconfig
```

Once the configuration window is open you can then select which kernel components should be included in the build. Exiting the configuration will save your selections to a file in the root of the kernel tree called `.config`.

Compiling the Kernel

Once the kernel has been configured it must be compiled to generate the bootable kernel image as well as any dynamic kernel modules that were selected. For u-boot the kernel should be built with a u-boot header that the u-boot program can read. This is easily accomplished by using the `uImage` build target in the kernel. In order to build the `uImage` target the command is:

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- uImage
```

This will result in a kernel image file being created in the `arch/arm/boot/` directory called `uImage`. This file can be used by u-boot to boot your Sitara device.

If you selected any components of the kernel to be build as dynamic modules you must issue an additional command to compile those modules. The command is:

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- modules
```

This will result in `.ko` (kernel object) files being placed in the kernel tree. These `.ko` files are the dynamic kernel modules. The next section will cover how to install these modules.

NOTE For user's who are building the kernel from within the AMSDK you can also use the Makefile at the root of the SDK installation at `<sdk install dir>/Makefile` to configure the kernel with the default SDK configuration and compile the `uImage` and kernel modules by doing:

```
cd <sdk install dir>
make linux
```

Installing the Kernel

Once the Linux kernel and modules have been compiled they must be installed. In the case of the kernel image this can be installed by copying the uImage file to the location where it is going to be read from. For example when using TFTP boot this may be the /tftpboot directory, whereas when booting from SD card this may be the first partition of the SD card.

To install the kernel modules you use another make command similar to the others, but with an additional parameter which give the base location where the modules should be installed. This command will create a directory tree from that location like lib/modules/<kernel version> which will contain the dynamic modules corresponding to this version of the kernel. The base location should usually be the root of your target file system. The general format of the command is:

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- INSTALL_MOD_PATH=<path to root of file system> modules_install
```

For example if you are installing the modules to an NFS share located at /home/user/targetNFS you would do:

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- INSTALL_MOD_PATH=/home/user/targetNFS modules_install
```

NOTE: For user's who are building the kernel from within the AMSDK you can also use the Makefile at the root of the SDK installation at <sdk install dir>/Makefile to install the kernel modules into the location pointed to by DESTDIR in your Rules.make file by doing:

```
cd <sdk install dir>
make linux_install
```

Out-of-tree Kernel Modules

NOTE: Some drivers like the SGX drivers are delivered as modules outside of the kernel tree. If you rebuild the kernel and install the modules using the "modules_install" target or the "make linux_install" instructions above you will also need to rebuild the out of tree modules and install them as well. The modules_install command used by both methods will remove any existing drivers before installing the new ones. This means those drivers are no longer available until they have been rebuilt against the kernel and re-installed.

For information on installing the kernel into NAND on the EVM (if supported) please see the [Installing the Linux Kernel](#) section of the [AMSDK u-boot User's Guide](#)

Additional Information

The links below provide additional information about the PSP kernel releases.

- For AM335x/Beaglebone
 - [AM335x PSP 04.06.00.03 Release Notes](#)
 - [AM335x PSP 04.06.00.03 Features and Performance Guide](#)
- For AM37x/AM3517/Beagleboard
 - [AM35x-AM37-PSP 04.02.00.07 Feature Performance Guide](#)
 - [AM35x-AM37-PSP 04.02.00.07 Release Notes](#)
- For AM180x
 - [AM180x-PSP 03.21.00.04 Feature Performance Guide](#)
 - [AM180x-PSP 03.21.00.04 Release Notes](#)

Archived Versions

- [05.03.03.00](#) ^[1]
- [05.03.00.00](#) ^[2]

References

[1] http://processors.wiki.ti.com/index.php?title=AMSDK_Linux_User%27s_Guide&oldid=90942

[2] http://processors.wiki.ti.com/index.php?title=AMSDK_Linux_User%27s_Guide&oldid=83825

Code Composer Studio v5 Users Guide



Return to the [Sitara Linux Software Developer's Guide](#)

Overview

Code Composer Studio v5.3 is currently provided with the Sitara Software Development Kit. It uses the Eclipse backend and includes the Remote System Explorer plug-in that provides tools to provide access to the remote target board.












Locating the CCSv5 Installer

Using the SD Card Provided with the EVM

When the SD card provided in the box with the EVM is inserted into an SD card reader attached to a Linux system three partitions will be mounted. The third partition, labeled START_HERE, will contain the CCS installer along with the Sitara Linux SDK installer. The CCS installer is located inside of the CCS directory and there is a helper script called `ccs_install.sh` available to help call the installer.

Downloading from the Web

The CCS installer is available for download as a compressed tarball (tar.gz) file on the same page as the Sitara Linux SDK download. The installer can be located by browsing to <http://www.ti.com/tool/linuxezsdk-sitara> ^[13] and selecting the device being used. On the individual SDK download page you can find the CCS installer under the **Optional Addons** section. i.e.

AM335x SDK Product Downloads		
Title	Description	Size
AM335x SDK Essentials		
 ti-sdk-am335x-evm-05.07.00.00-Linux-x86-Install	AM335x EVM SDK	1295184K
AM335x SDK Optional Addons		
 CCS-5.3.0.00090_Sitara-ARM.tar.gz	Code Composer Studio for Sitara ARM	1485528K
 README.ccs	Code Composer Studio for Sitara ARM README	4K
AM335x SDK Individual Components		
Sitara Linux SDK Release Notes	Link to Release Notes for Sitara Linux SDK	
 am335x-evm-qsg.pdf	AM335x EVM Quick Start Guide	2316K
 am335x-evm-sk-qsg.pdf	AM335x EVM-SK Quick Start Guide	2156K
 beaglebone-qsg.pdf	BeagleBone Quick Start Guide	176K
 sitara-linuxsdk-sdg-05.05.01.00.pdf	Software Developers Guide	K
Wiki version of Software Developers Guide	Link to the online Software Developers Guide which has the latest content	
 Software Manifest	Software Manifest of Components Inside the SDK	612K
 am335x-evm-sdk-src-05.07.00.00.tar.gz	AM335x SDK PSP Source Code	363740K
 am335x-evm-sdk-bin-05.07.00.00.tar.gz	AM335x SDK prebuilt PSP binaries and root filesystem	607112K
Download Pinmuxtool	Sitara Pin Mux Configuration Utility	
AM335x SDK Checksums		
 md5sum.txt	MD5 Checksums	4K

Clicking this link will prompt you to fill out an export restriction form. After filling out the form you will be given a download button to download the file and you will receive an e-mail with the download link. Download the tarball and save it to your Linux host development system.

Starting the CCSv5 Installer

Using the Sitara Linux SDK Installer

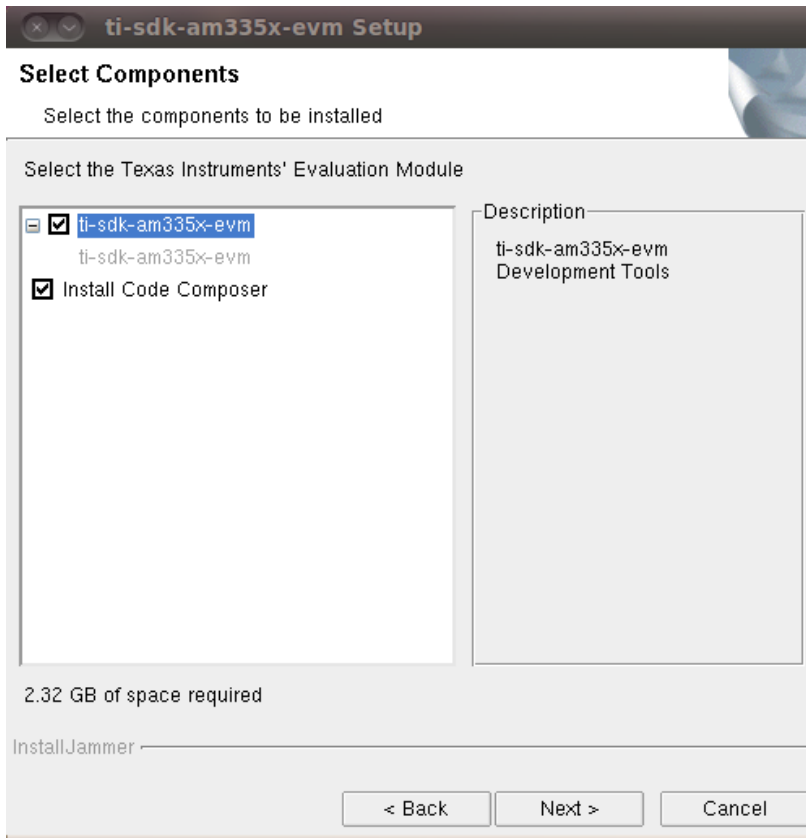
The Sitara Linux SDK installer has the capability of launching the CCSv5 installer during the SDK installation. In order for the installer to launch the CCSv5 installer the *CCS* directory and *ccs_install.sh* script must be located in the same directory as the Sitara Linux SDK installer. When installing from the SD card found in the EVM box this directory and script are already located in the same directory as the Sitara Linux SDK installer. However, if you downloaded the CCS installer tarball from the ti.com website as mentioned above, then you will need to:

1. Place the CCS tarball (the .tar.gz file) in the same directory as the Sitara Linux SDK installer
2. Extract the CCS tarball using a command like:

```
tar xzf CCS_<version>_Sitara_ARM.tar.gz
```

Where <version> is the version string of the CCS installer

After the CCS installer files are located in the same directory as the Sitara Linux SDK installer you can execute the SDK installer to begin SDK installation. During the SDK installation you will see a screen similar to the one below. The option to **Install Code Composer** is enabled by default.



NOTE: If the **Install Code Composer** option is selected and the CCS install files are not located in the same directory as the Sitara Linux SDK installer you will be given a message that the installer could not be located and the SDK installation will continue as normal. To install CCS later you can follow the steps in the next section to bypass the Sitara Linux SDK installer.

From Linux Command Line

If you want to install CCSv5 apart from the Sitara Linux SDK installer, or if you decided not to install it as part of the SDK install and want to install it now, you can install CCS using the following commands:

1. Open a Linux terminal and change directory to the location where the CCS files are located. This may be the START_HERE partition of the SD card, or the location where you downloaded the tarball file from the ti.com website.

2. If the CCS files are still in a compressed tarball extract them using the command

```
tar xzf CCS_<version>_Sitara_ARM.tar.gz
```

Where <version> is the version string of the CCS installer

3. Invoke the CCS installer using the **ccs_install.sh** script located in the START_HERE directory (copy the script to your current directory).

```
./ccs_install.sh $PWD
```

NOTE: You can also invoke the CCS installer using in the CCS directory using the commands:

```
cd CCS
```

```
./ccs_setup*.bin --setupfile ccs_installini.xml
```

CCSv5 Installation Steps

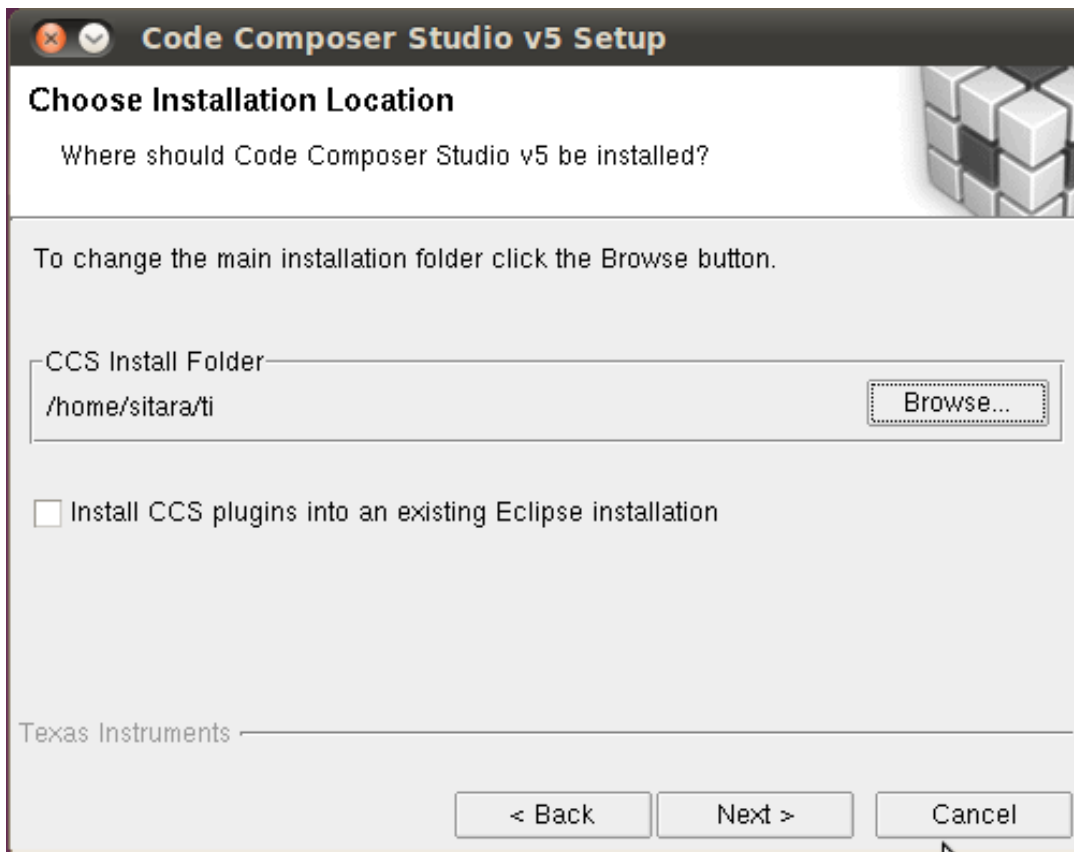
NOTE: The "Limited 90-day period" language in the CCS installer license agreement applies only for the case of using high-speed JTAG emulators (does not apply to use of the XDS100v2 JTAG emulator or an on-board emulator). If a debug configuration is used that requires a high-speed JTAG emulator, you will be prompted to register your software for a fee. All use of CCSv5 (excluding use of high-speed JTAG emulators) is free and has no 90-day time limit.

When the CCSv5.3 installer runs you can greatly reduced the install time and installed disk space usage by taking the defaults as they appear in this CCS installer. The screen captures below show the default installation options and the recommended settings when installing CCSv5.

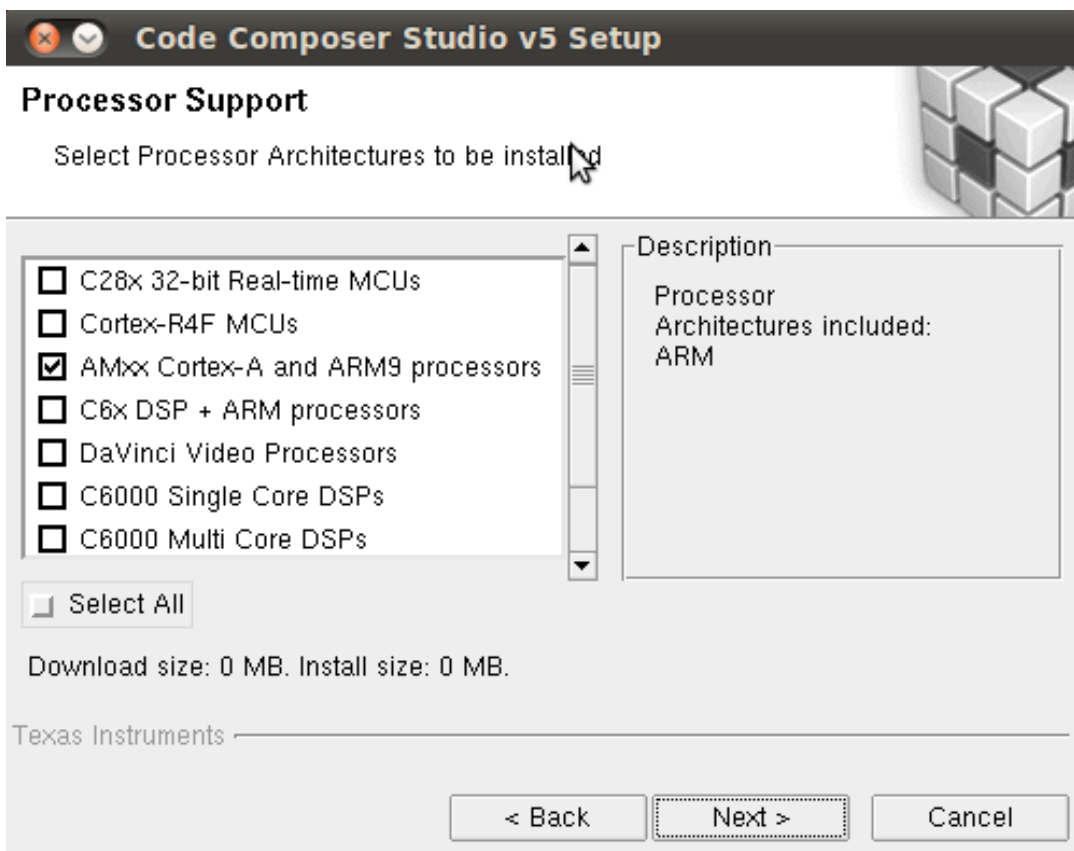
1. The *License Agreement* screen will prompt you to accept the terms of the license agreement. Please read these terms and if you agree select **I accept the terms of the license agreement**. If not then please exit the installation.



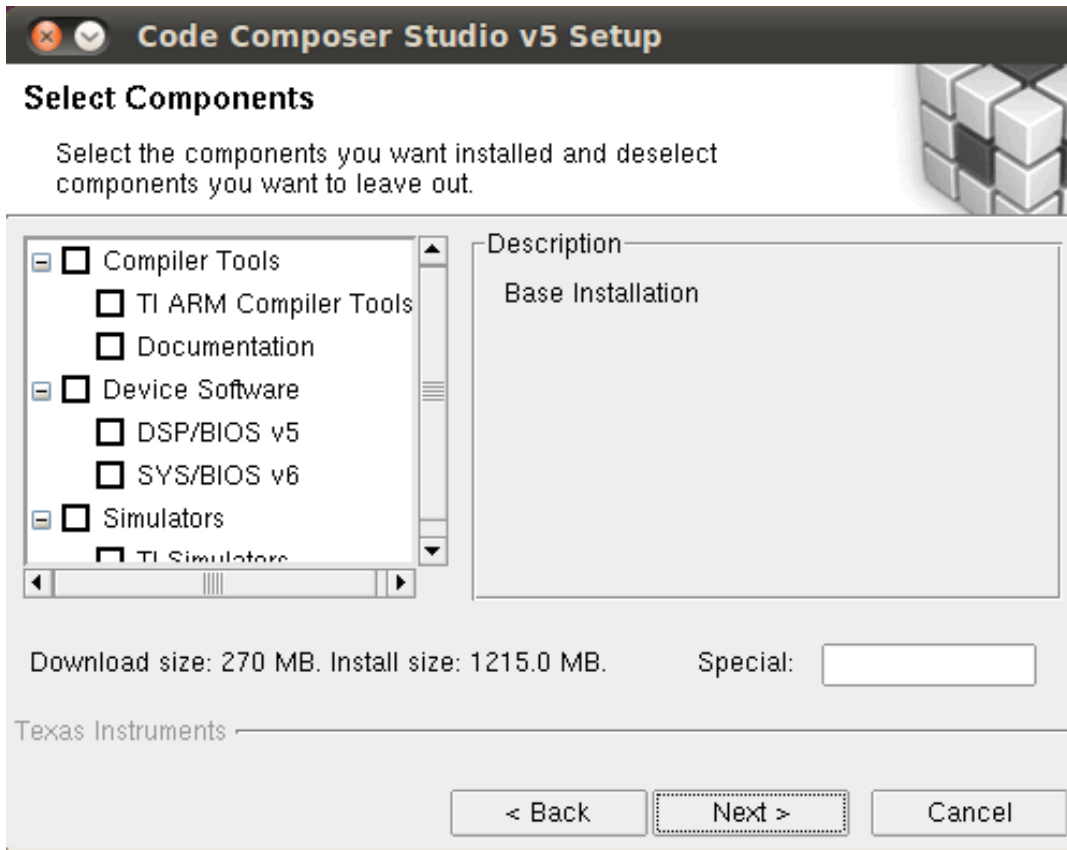
2. At the *Choose Installation Location* screen do **NOT** check **Add TI plug-ins into an existing Eclipse install**



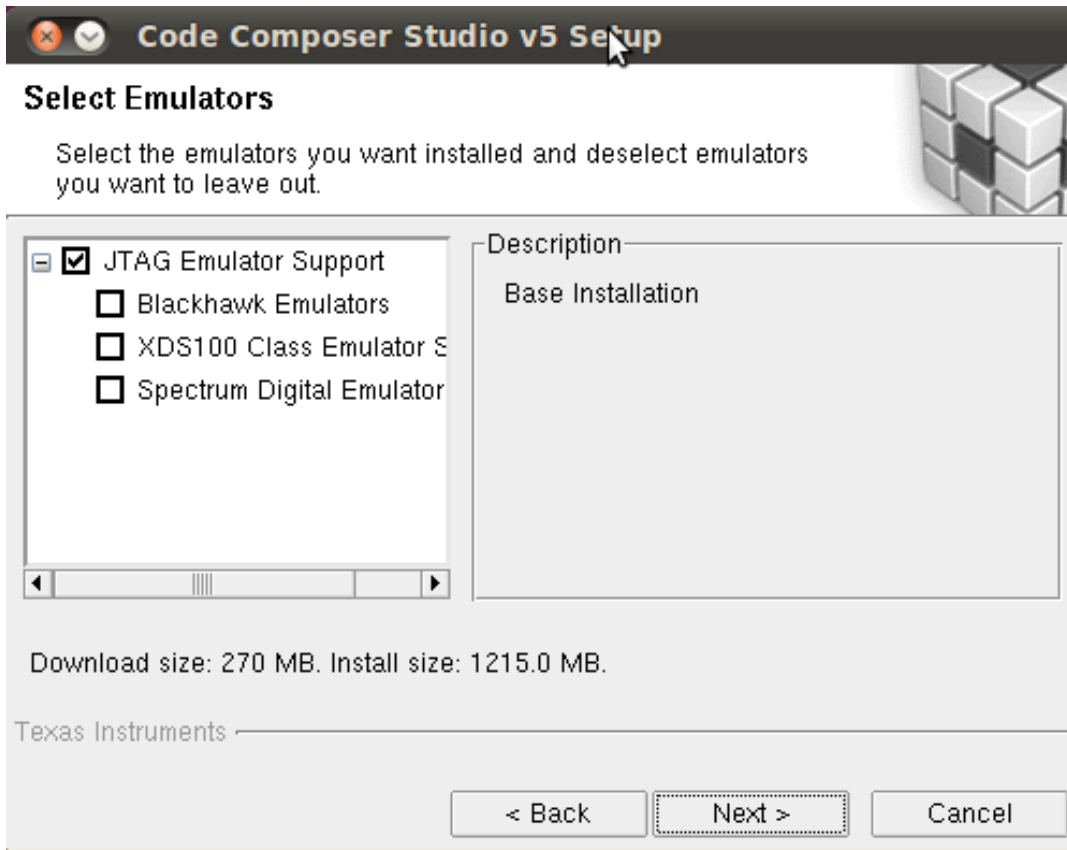
3. At the *Processor Support* screen make sure to select the **AMxx Cortex-A and ARM9 processors** option



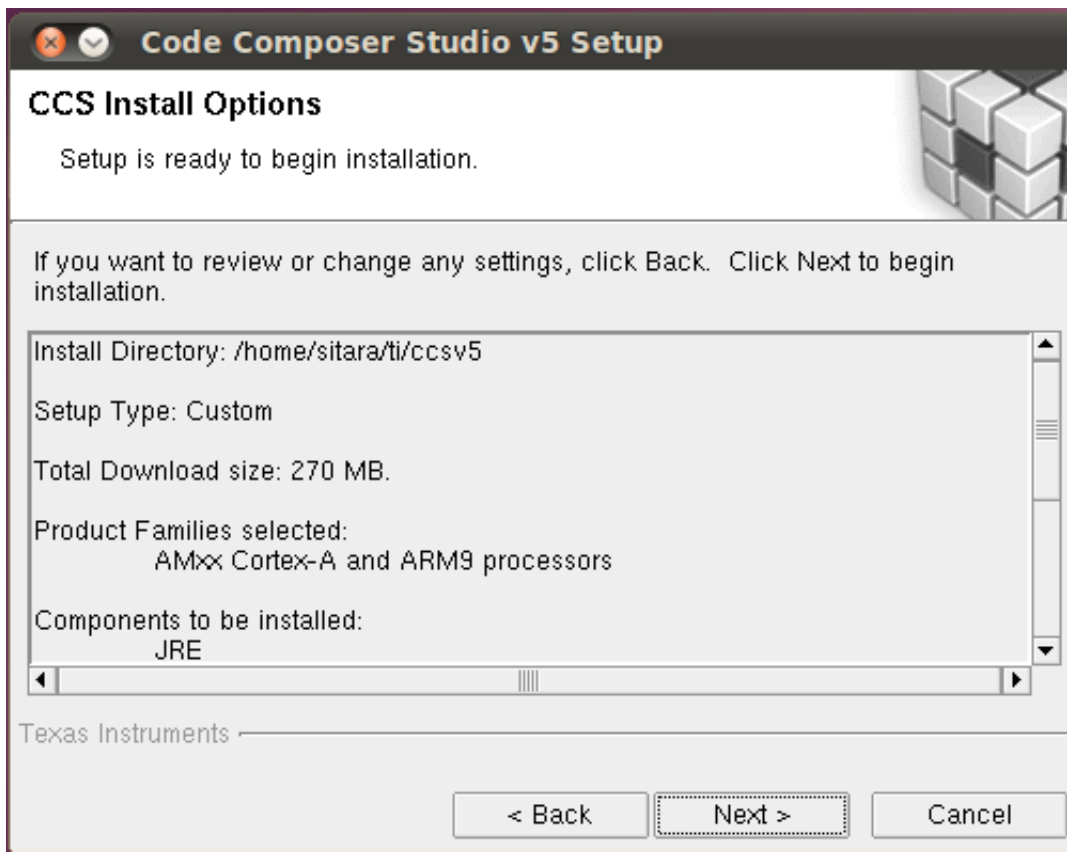
4. At the *Select Components* screen do **NOT** select **TI ARM Compiler Tools**. These tools are the TI compiler tools for ARM devices, whereas for Linux the Sitara Linux SDK uses the open source GCC compiler.



- 5. At the *Select Emulators* screen make sure that **JTAG Emulator Support** is enabled but you do not need to select individual emulators unless you require support for that model of JTAG. To install those drivers later see the *Installing Emulator Support* section below.



- 6. At the *CCS Install Options* screen verify that the options look correct and then select **Next** to begin installation.



7. After the installation has completed click **Finish**

Installing Emulator Support

If during the CCSv5 installation, you selected to install drivers for the Blackhawk or Spectrum Digital JTAG emulators a script must be run with administrator privileges

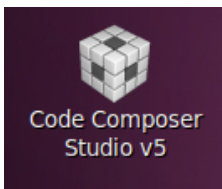
to allow the Linux Host PC to recognize the JTAG emulator. The script must be run as "sudo" with the following command:

```
sudo <CCSv5_INSTALL_PATH>/ccsv5/install_scripts/install_drivers.sh
```

where <CCSv5_INSTALL_PATH> is the path that was chosen when the CCSv5 installer was run.

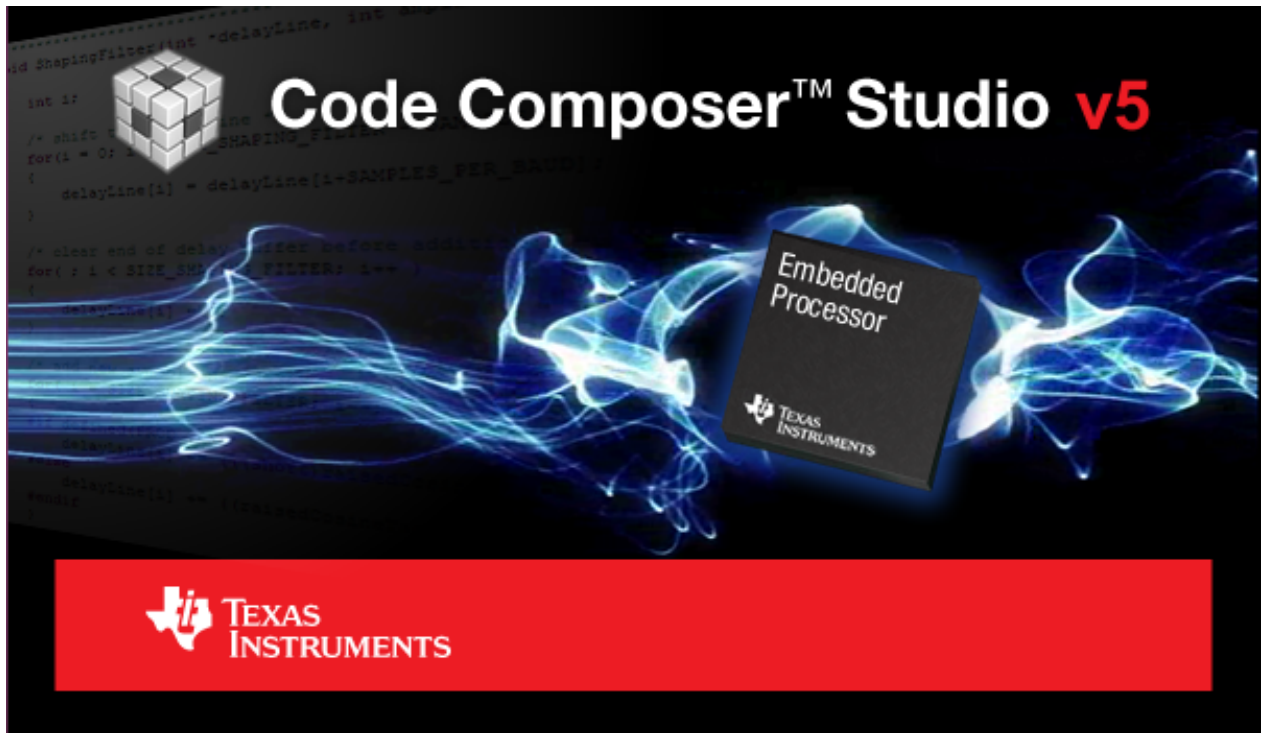
Launching CCS

After the CCS installer has finished executing you should have an icon on your desktop call **Code Composer Studio v5** like:

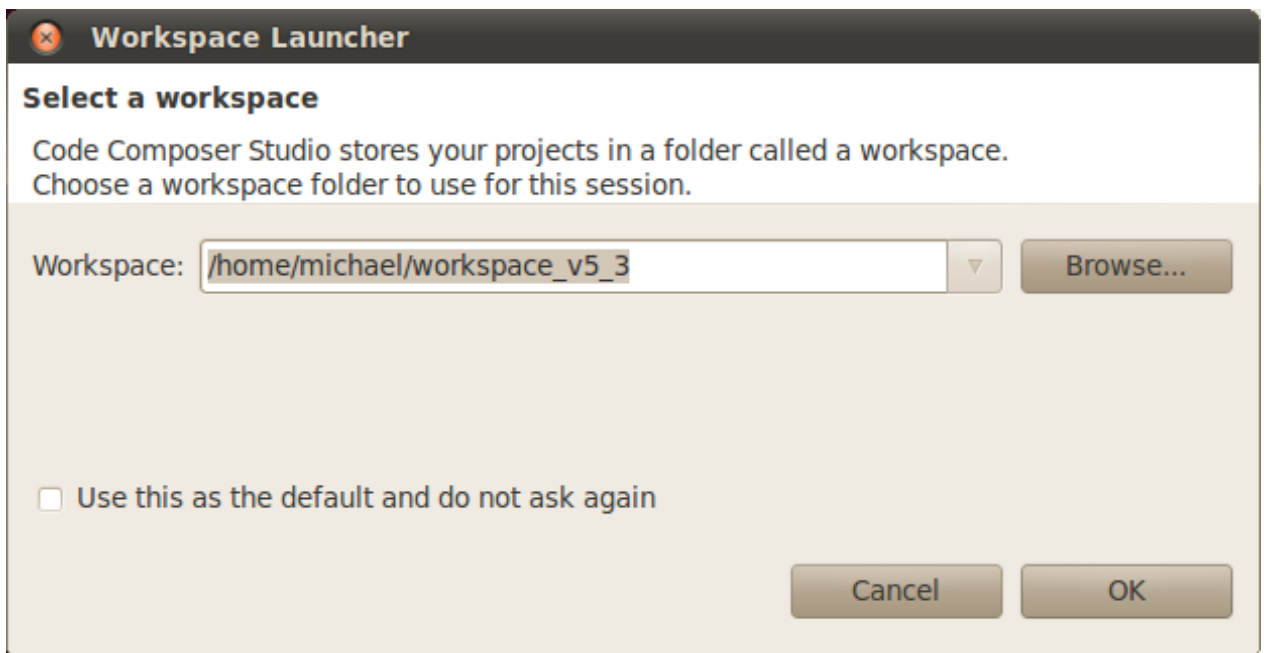


To launch CCS you should:

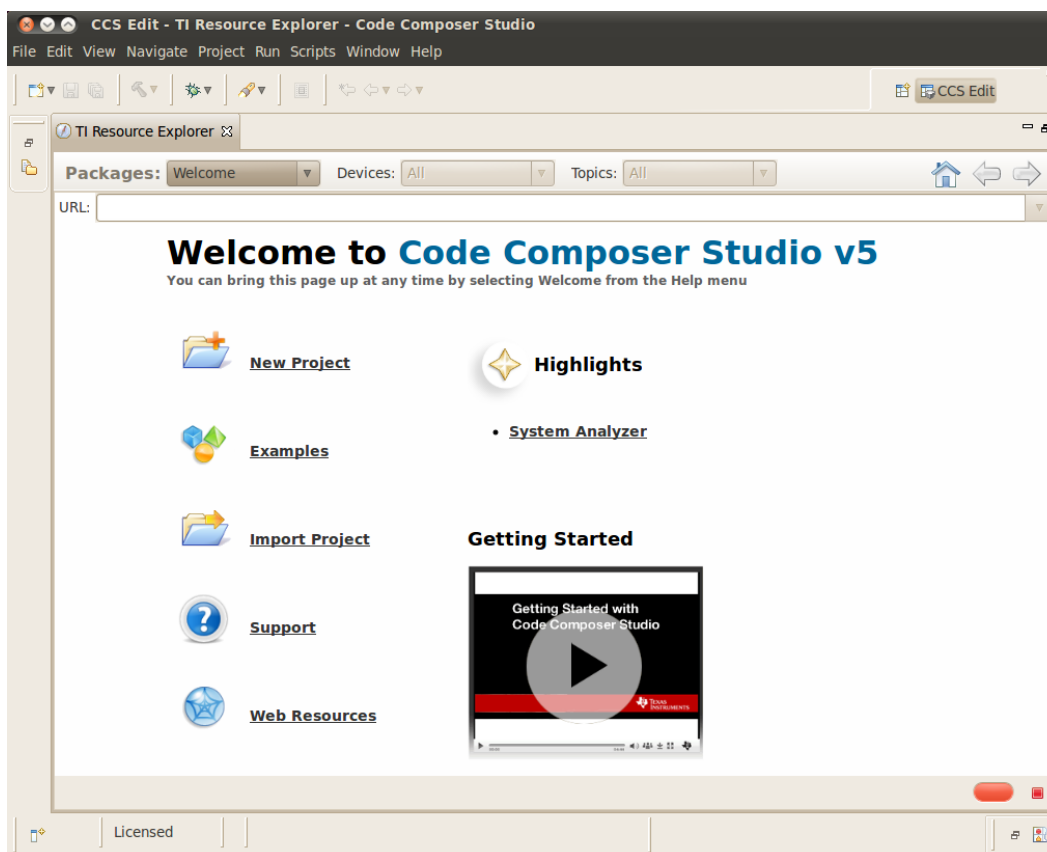
1. **Double-Click** the CCS icon on the desktop. You will see the CCSv5 splash screen appear while CCS loads



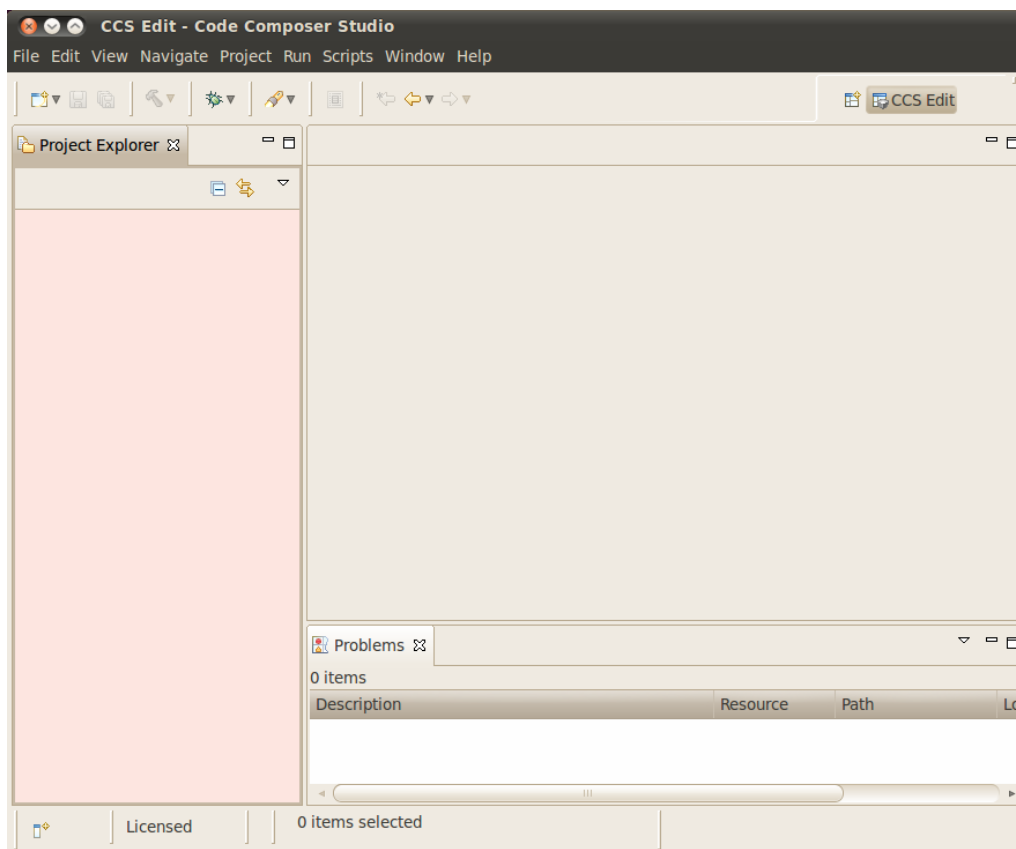
2. The next window will be the **Workspace Launcher** window which will ask you where you want to locate your CCSv5 workspace. You can take the default here or choose a custom directory.



3. CCS will load the workspace and then launch to the default **TI Resource Explorer** screen



4. Close the **TI Resource Explorer** screen. This screen is useful when making TI CCS projects which use TI tools. The Sitara Linux SDK uses open source tools with the standard Eclipse features and therefore does not use the TI Resource Explorer. You will be left in the Project Explorer default view.

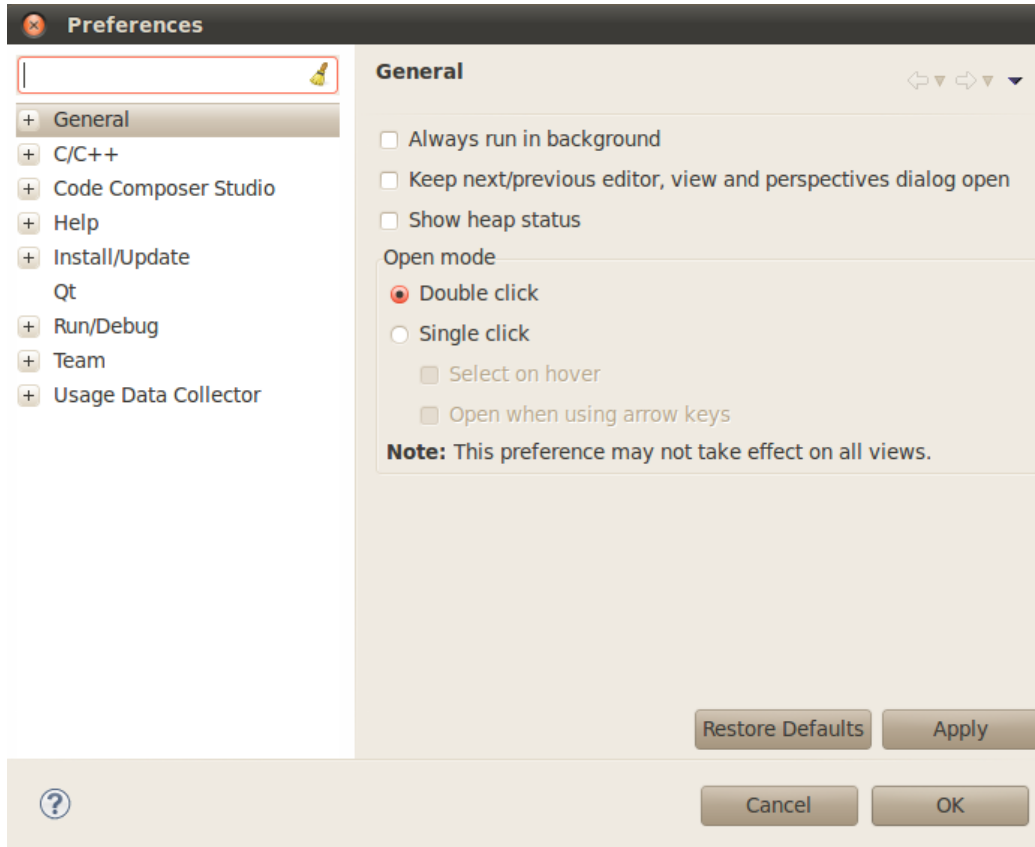


Enabling CCS Capabilities

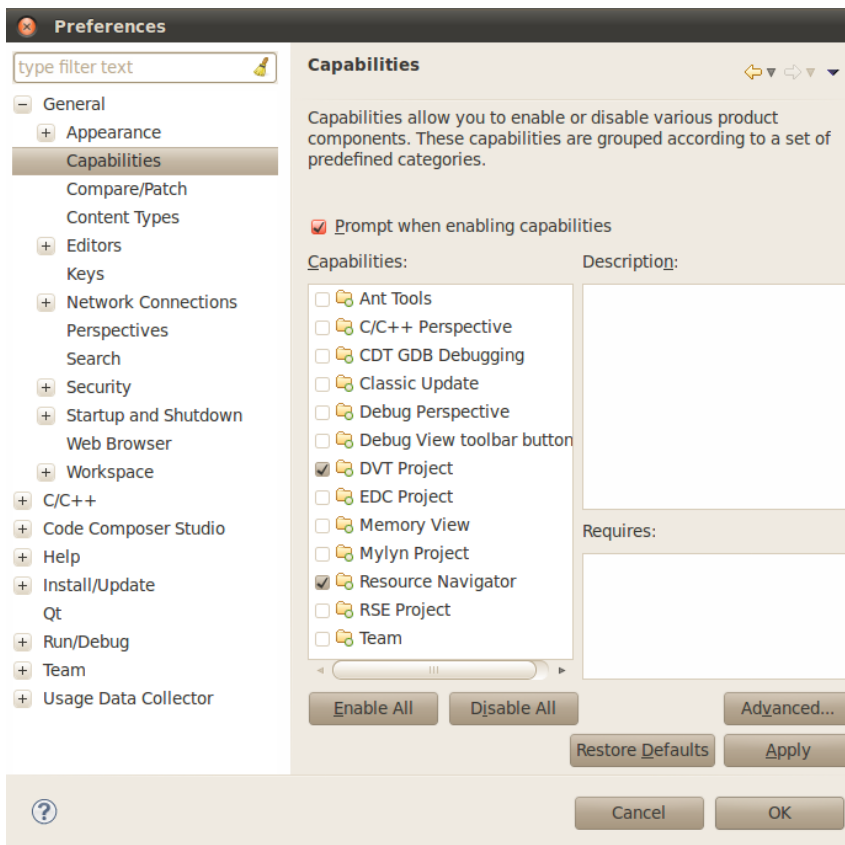
Each time CCSv5 is started using a new workspace, additional capabilities need to be enabled so that perspectives for those capabilities will be selectable in the **Window -> Open Perspectives** list.

After opening CCSv5 with a new workspace:

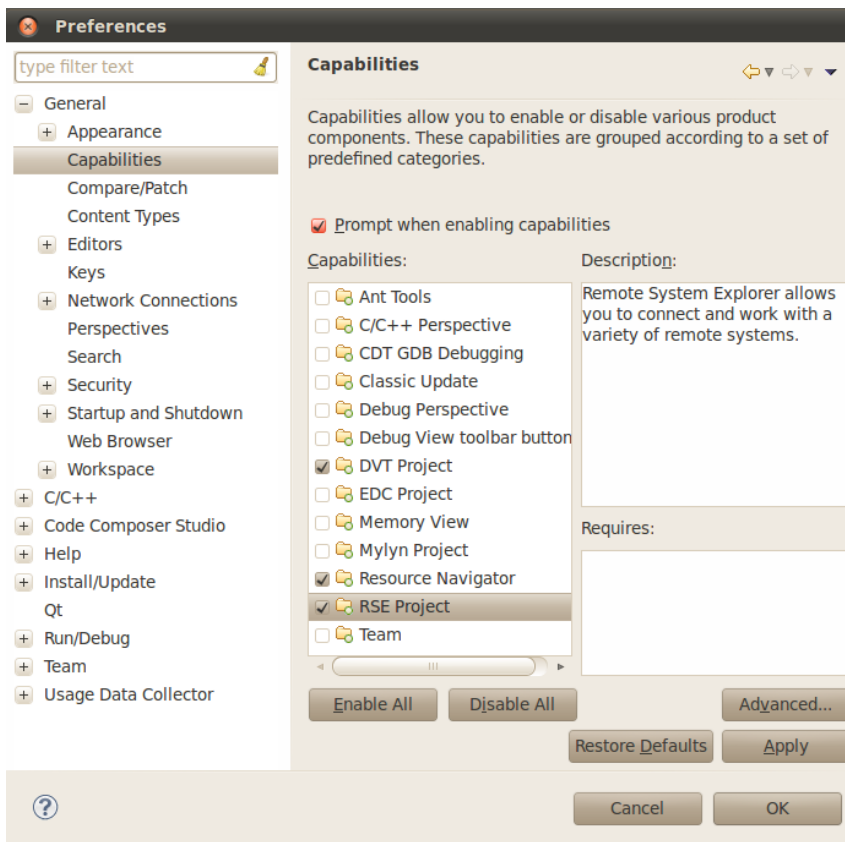
1. Open the **Window -> Preferences** menu



2. Go to the **General -> Capabilities** menu



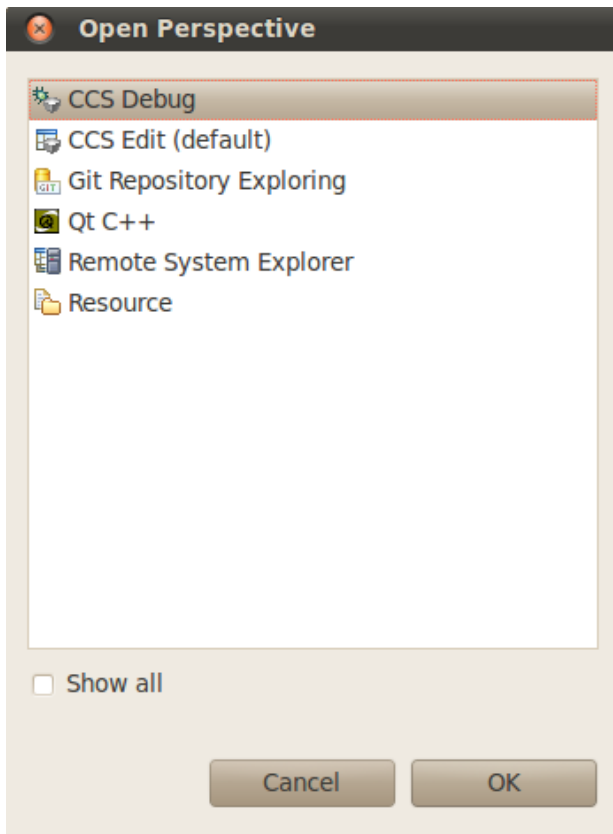
3. Select the **RSE Project** Capability



4. Click **Apply** and then **OK**

This enables the perspectives in the **Window -> Open Perspective -> Other** menu as shown below and is needed to make the Remote System Explorer plug-ins selectable.

Note: The Qt C++ Perspective is not compatible with this version of Eclipse. Instead Qt projects are to be built using the Makefiles inside of the project as will be detailed in the later sections of this guide.

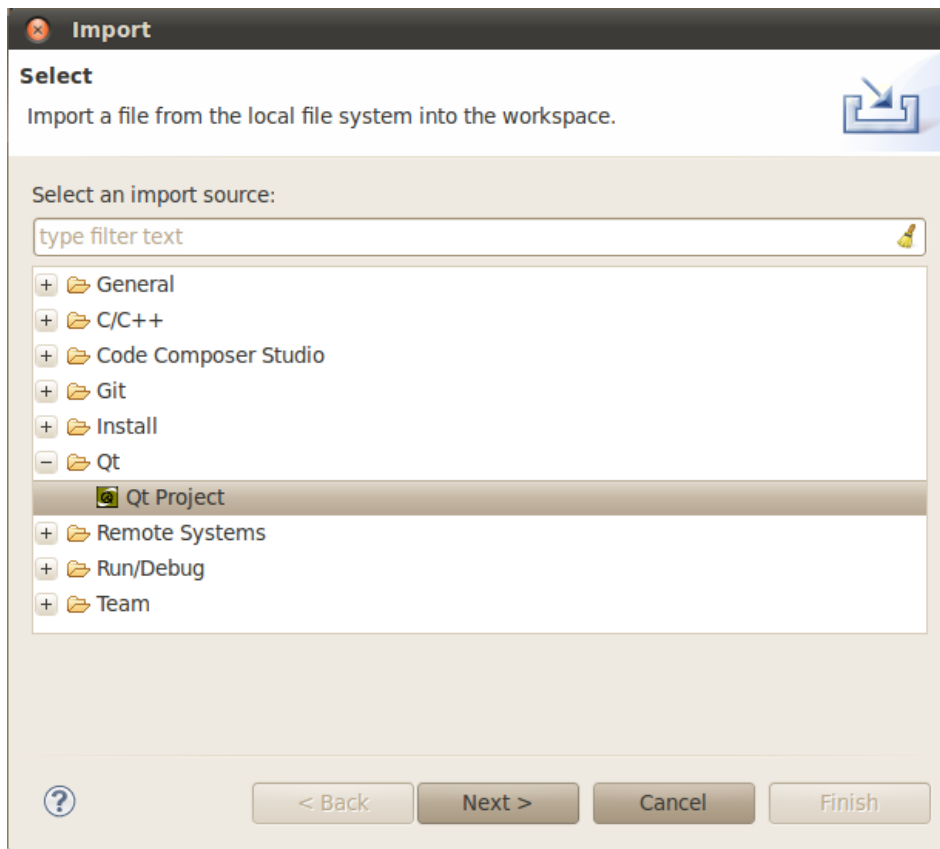


Importing Qt Projects

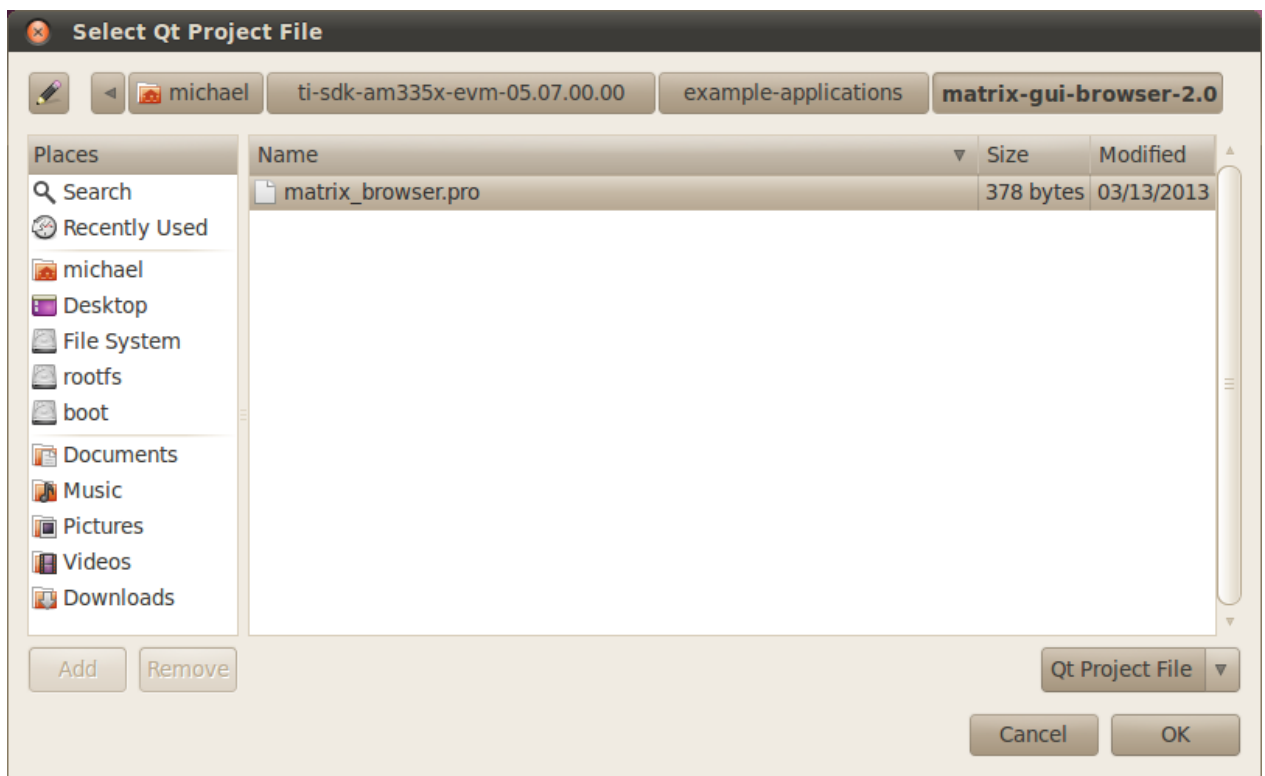
Since the Qt plugin does not work with the latest version of Qt the example projects have been modified to use a Makefile to handle the build step. You can use projects like the **matrix-gui-browser** project as a reference for how to configure a Qt project to build using a Makefile. The following steps detail the changes that should be made for an existing Qt project to use a Makefile to build.

Importing matrix browser project

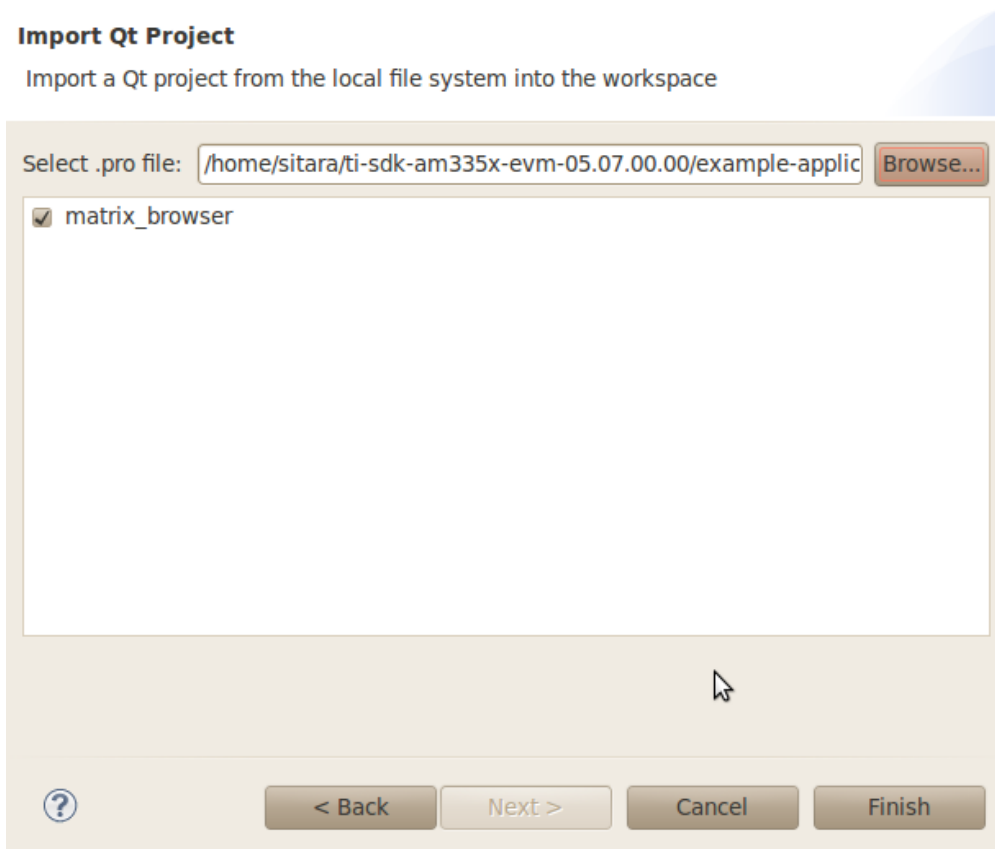
1. First import the Qt project using the **File -> Import...** menu
2. Select the **Qt -> Qt Project** option



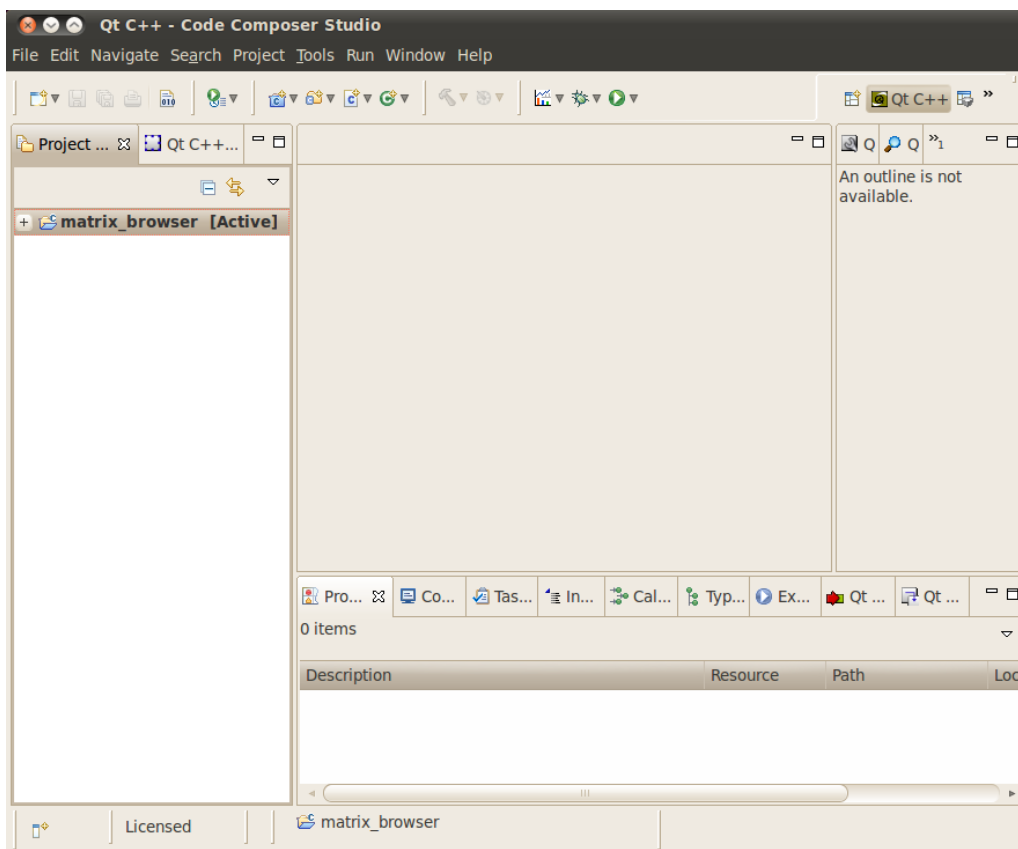
3. Press the **Next >** button
4. On the **Import Qt Project** screen click the **Browse...** button and locate the *matrix_browser.pro* file within the `<SDK INSTALL DIR>/example-applications/matrix-gui-browser-x.x` directory



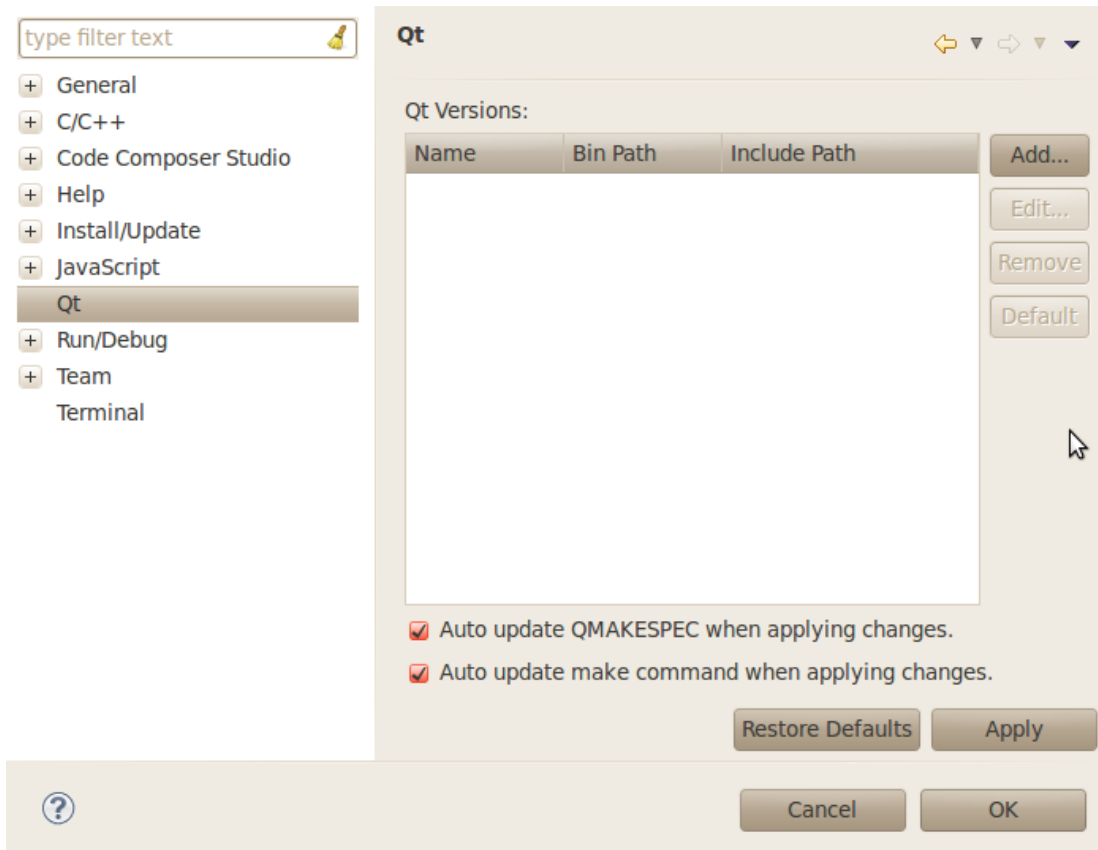
5. Click **OK**
6. You should now see the *matrix_browser* project listed as being selected for import into CCS



7. Click **Finish** to import the matrix-gui-browser project
8. You should now see the matrix_browser project in the **Project** view



9. Now you will need to add the Qt version found in the SDK to CCS. This can be done using **Window -> Preferences** and selecting the *Qt* menu item



10. Select the **Add...** button
11. In the **Add new Qt version** dialog fill in:

Version Name: AMSDK Qt

Bin Path: <SDK INSTALL DIR>/linux-devkit/bin

Include Path: <SDK INSTALL DIR>/linux-devkit/arm-arago-linux-gnueabi/usr/include/qtopia

Add new Qt version

Specify the Name and Bin + Include Pathes of the Qt version.

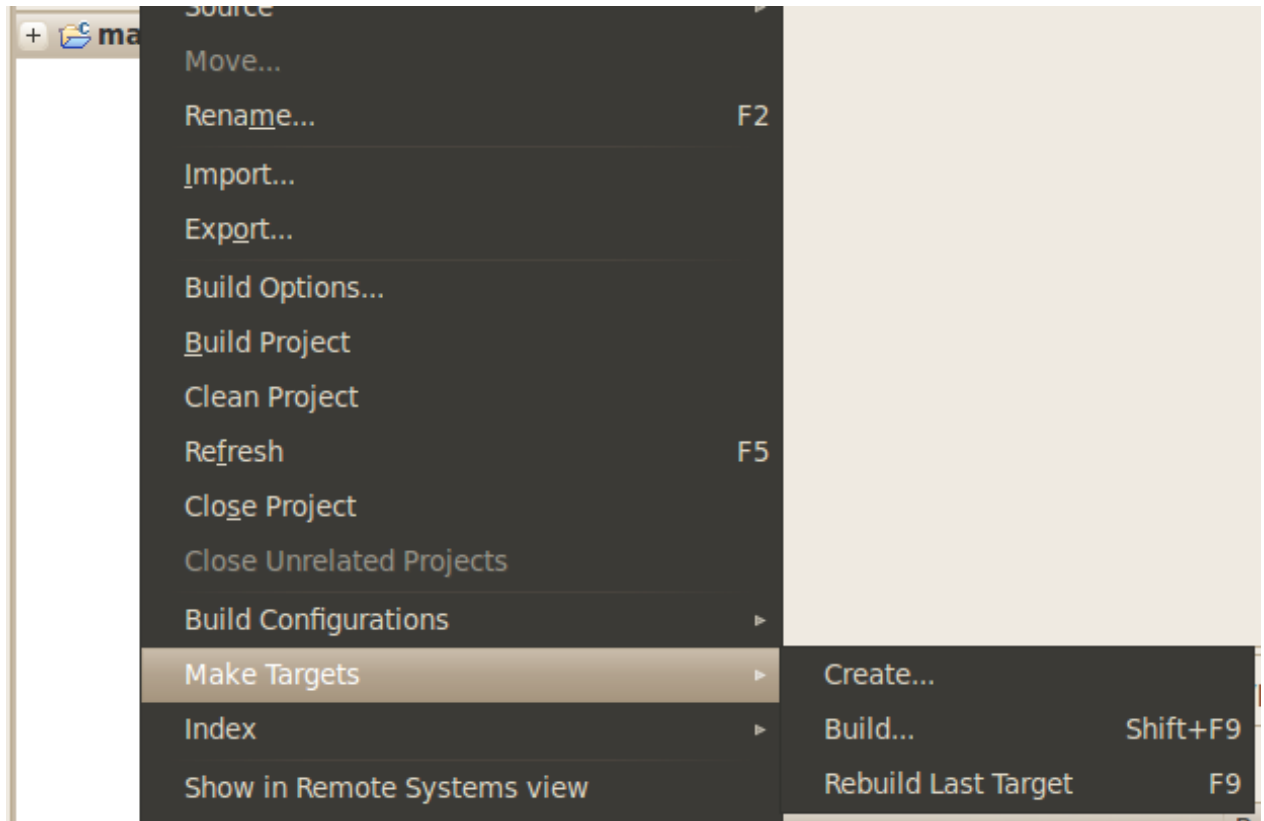


12. Click **Finish**
13. Click **Apply**
14. When prompted that **Qt versions have changed** select **Yes**
15. Click **OK**

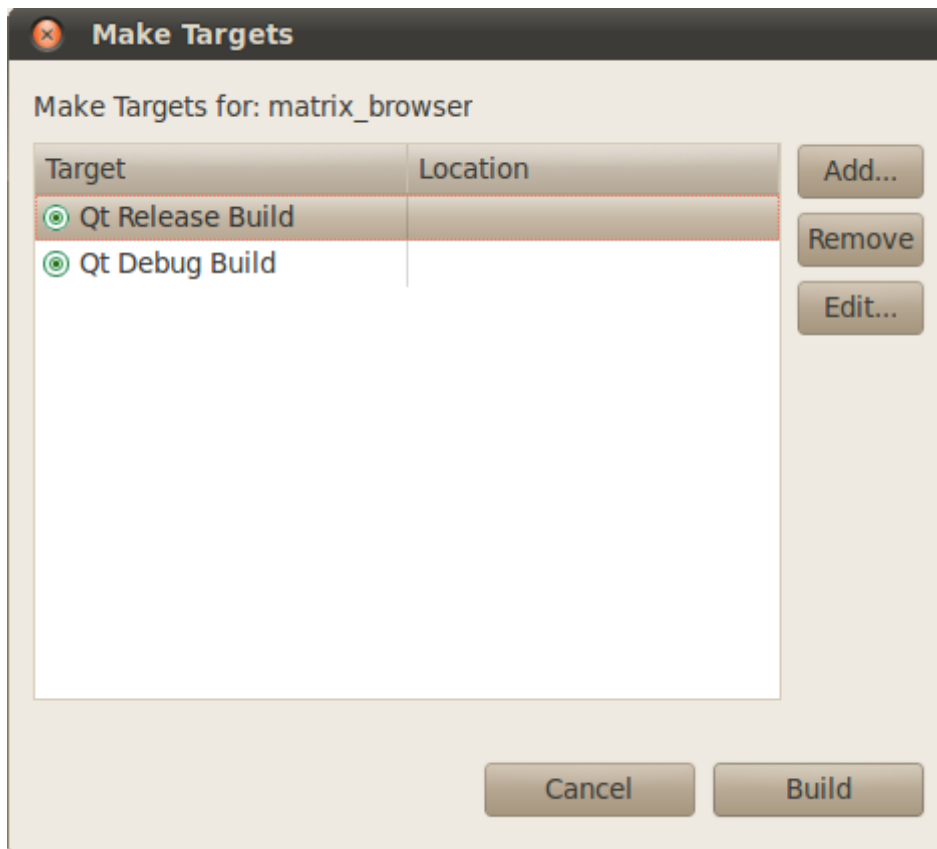
Changing the Make Target

By default the debug target is compiled when you selected to rebuild the Qt projects due to the Qt version change above. You can build the release version by doing:

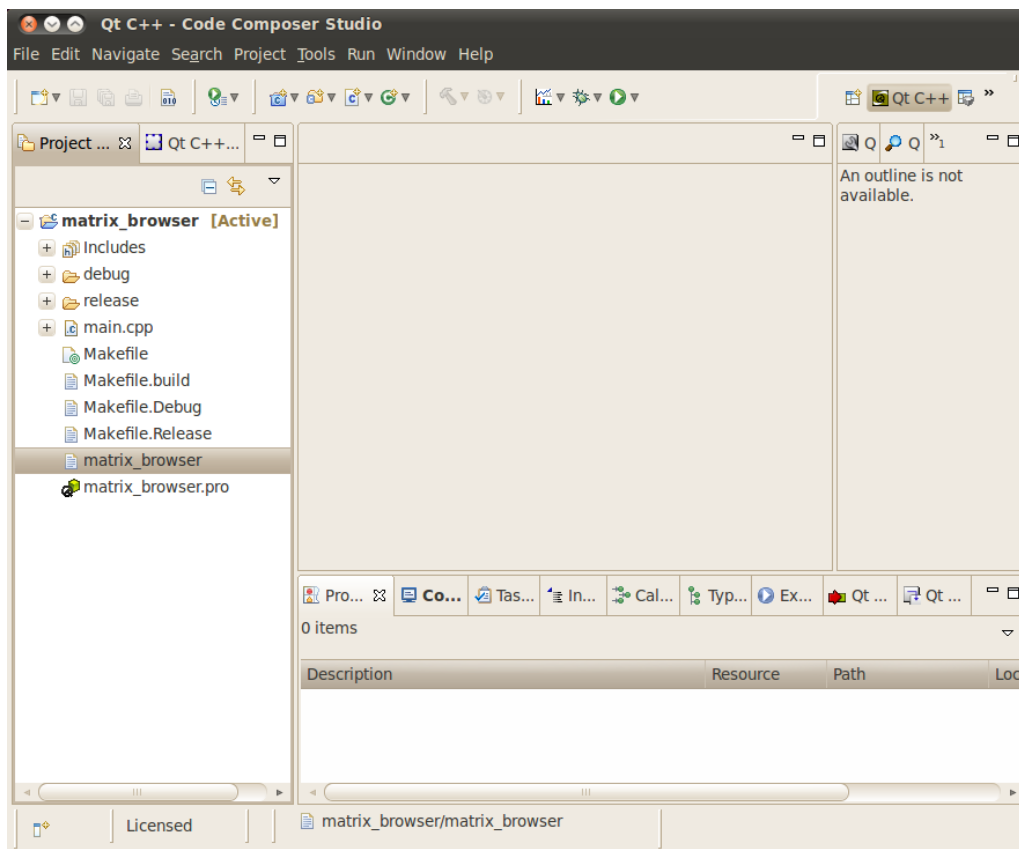
1. Right Click matrix_browser project
2. Select **Make Targets -> Build...**



3. Highlight **Qt Release Build** and click the **Build** button



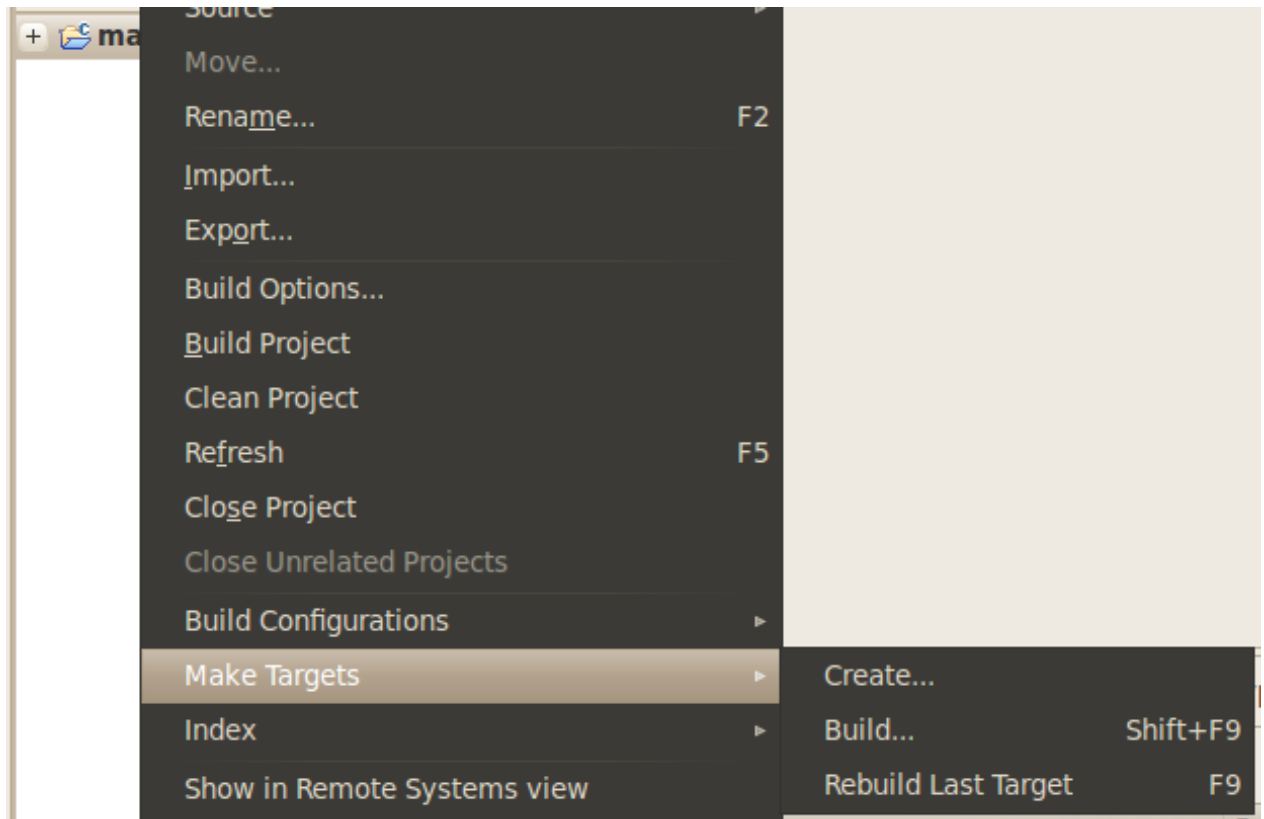
4. You will find the matrix_browser executable built in the matrix_browser project.



Creating a New Make Target

You may want to create additional make targets for steps like the installation step. In this example we will make an install target that installs the release version of the matrix_browser executable.

1. Right click **matrix_browser** project and select **Make Targets -> Create...**



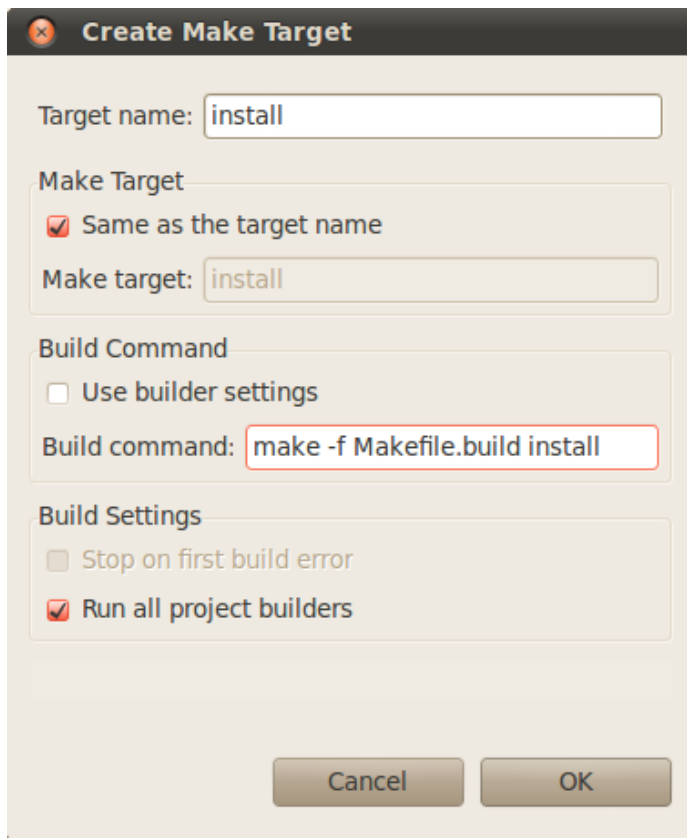
2. In the dialog box set:

Target name: install

Check **Same as the target name** for the **Make Target**

un-check **Use builder settings**

Change **Build command:** to *make -f Makefile.build install*



3. Click **OK**

You can now build the **install** target using the steps in the [Changing the Make Target](#) section above.

Using a Makefile

In order for the above steps to work a `Makefile.build` Makefile was created in the `matrix-gui-browser` directory. This `Makefile.build` has some key points that are worth mentioning.

- The `Rules.make` file is included from the top-level of the Sitara Linux SDK. This is to provide access to variables like `DESTDIR` for installing the built executable.

```
-include ../../Rules.make
```

- There is a variable called `ENV_SETUP` that points to the **environment-setup** script in the `linux-devkit` directory. The `qmake` target, which is used by the `release` and `debug` targets will first source the **environment-setup** script to get access to `qmake2` and configure the build to use the Qt version inside of the Sitara Linux SDK

```
qmake : matrix_browser.pro //qmake target depends on
matrix_browser.pro
    @ . ${ENV_SETUP}; \ //source the environment-setup script using
    a shell
    qmake2 CONFIG+=debug_and_release
QMAKE_CXXFLAGS_DEBUG+=-D${PLATFORM_DEFINE}
QMAKE_CXXFLAGS_RELEASE+=-D${PLATFORM_DEFINE} matrix_browser.pro //call
qmake2 to make the project Makefiles
```

Installing to the Target File System

Depending on your file system type you can use the methods below to install the `matrix_browser` executable. If the file system is NFS you should have first run the [SDK Setup Script ^[1]]

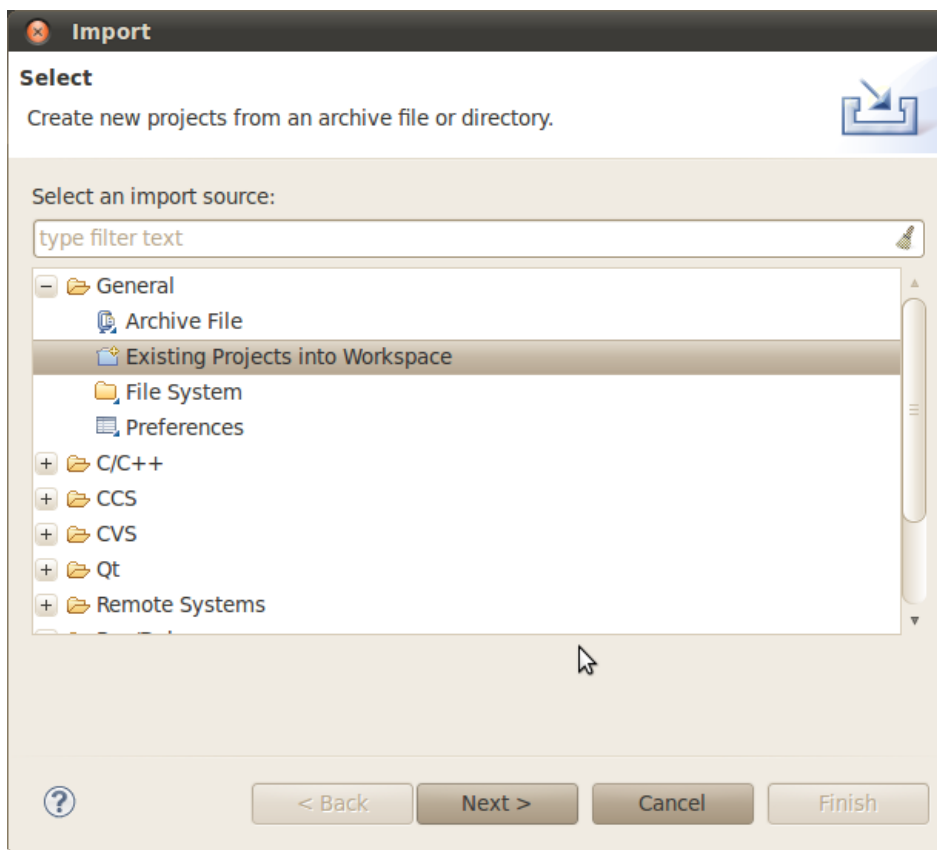
1. Create a Make Target using the steps in the **Creating a New Make Target** section above
2. Browsing to the `<SDK INSTALL DIR>/example-applications/matrix-gui-browser-x.x/` directory in a terminal and typing `make -f Makefile.build install`
3. For all file system types you can also transfer the file using the drag-and-drop method of Remote System Explorer. See the **Remote System Explorer** section below for more details

Importing C/C++ Projects

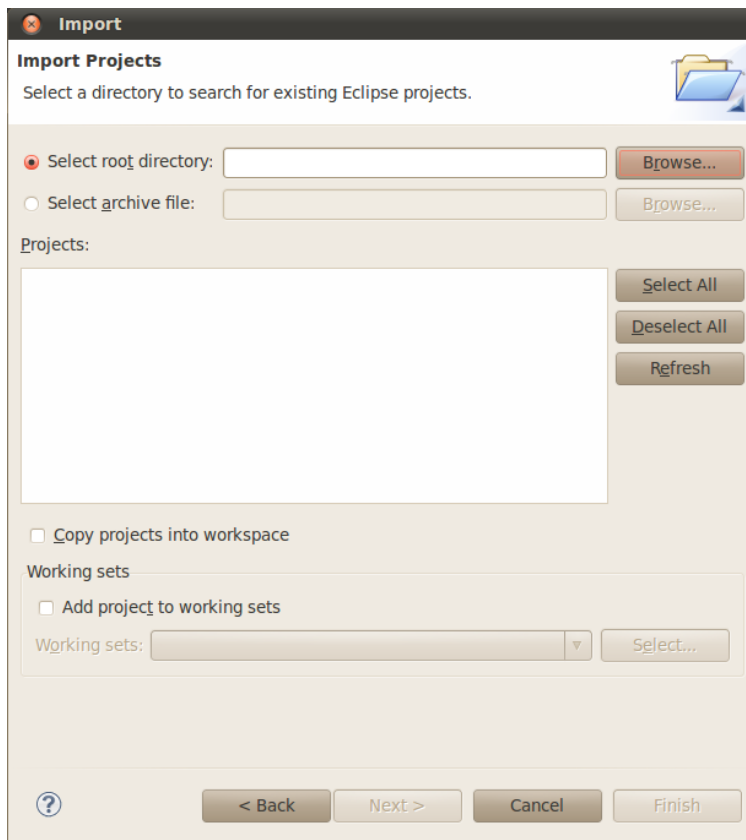
Importing the Projects

The following instructions will help you to import the example C/C++ application projects into CCSv5. For instructions on importing Qt application see the **Importing Qt Projects** section above.

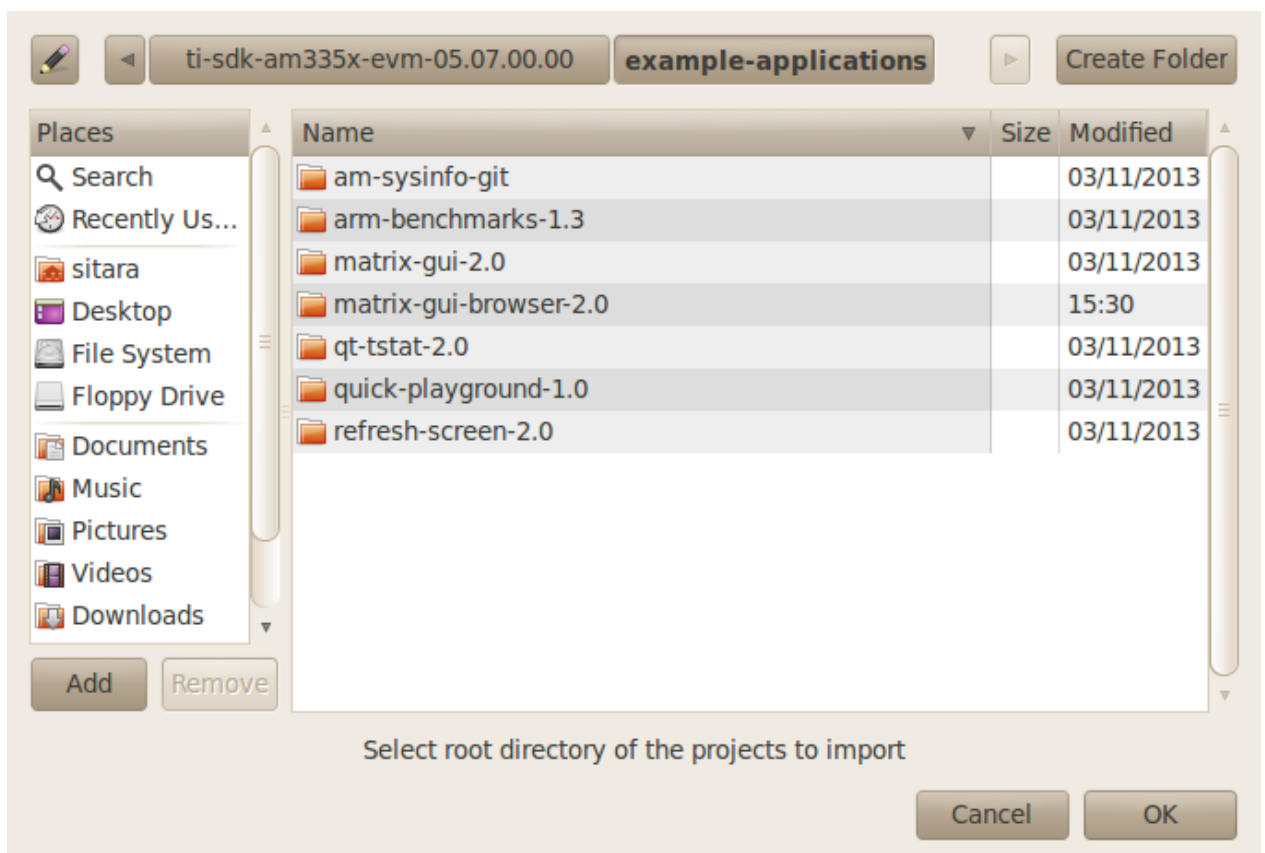
1. From the main CCSv5 window, select **File -> Import...** menu item to open the import dialog
2. Select the **General -> Existing Projects into Workspace** option



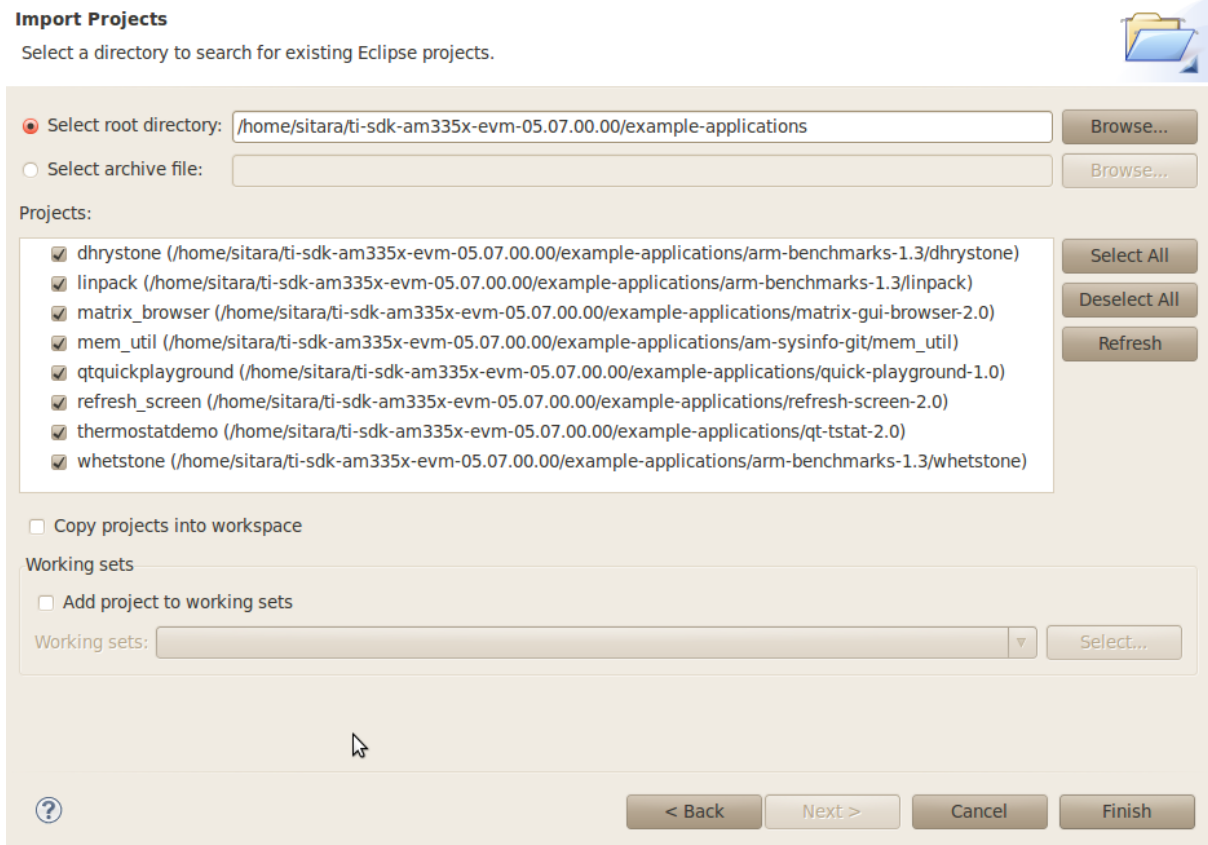
3. Click **Next**
4. On the *Import Projects* page click **Browse**



5. In the file browser window that is opened navigate to the `<SDK INSTALL DIR>/example-applications` directory and click **OK**



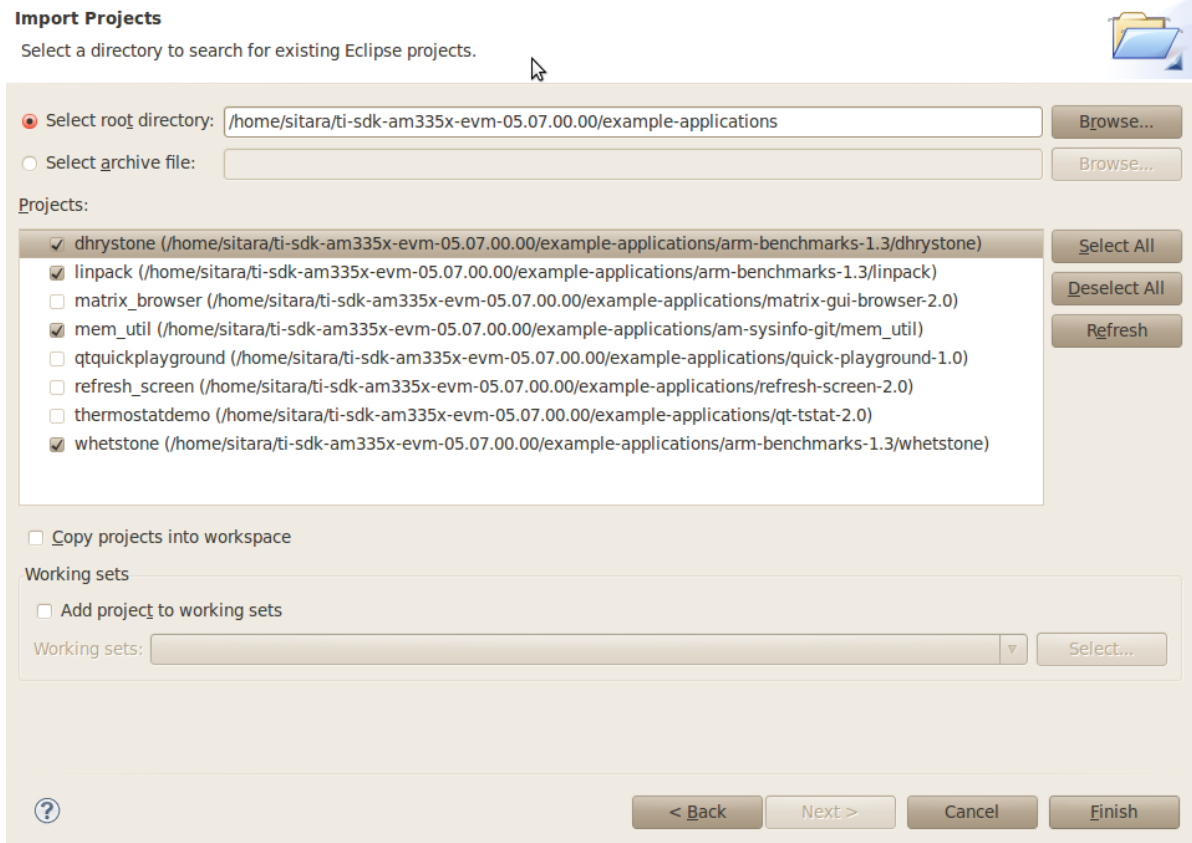
6. The *Projects:* list will now be populated with the projects found.



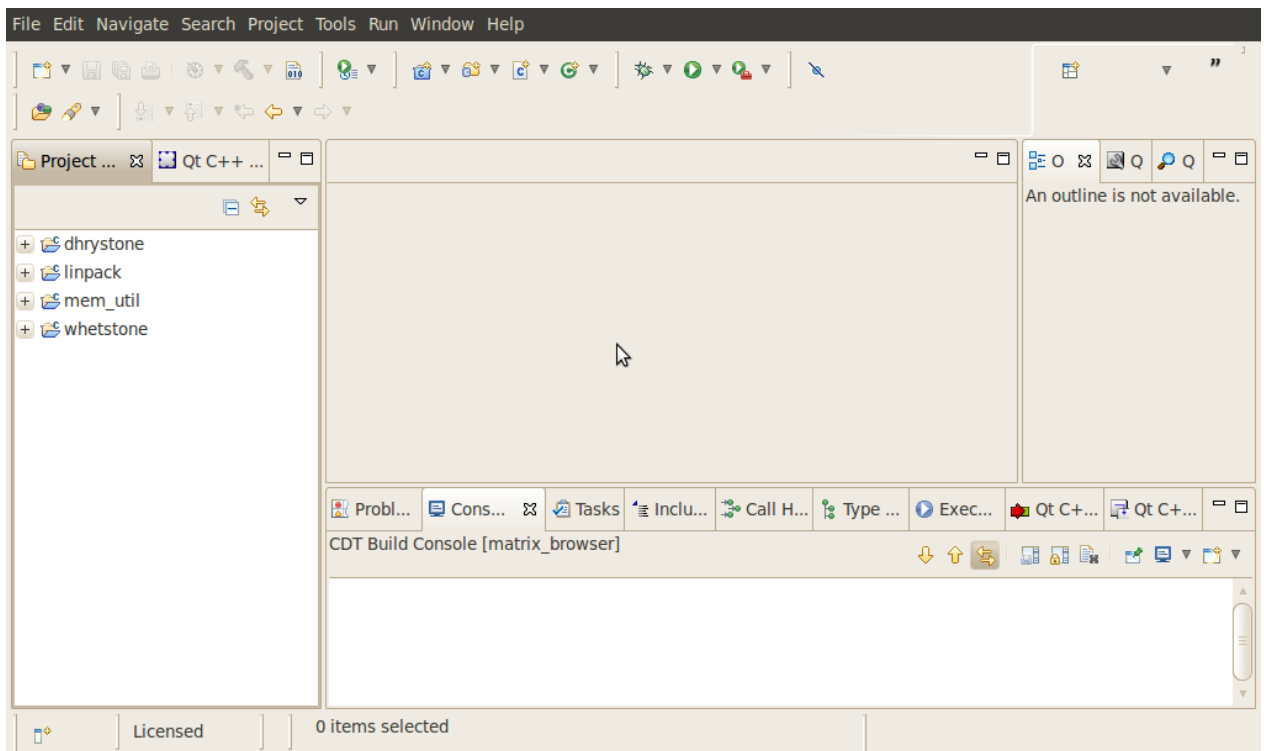
- **NOTE:** The `matrix_browser`, `qtquickplayground`, `refresh_screen`, and `thermostatdemo` projects should be un-checked since they are Qt projects and should be imported using the steps in the Importing Qt Projects section above.

Your Sitara SDK installation probably contains other Qt projects which need to be unchecked when importing C/C++ projects. You can tell which are the Qt projects. Qt projects have a `.PRO` project file.

Select the projects you want to import. The following screen capture shows importing all of the example projects for an ARM-Cortex device, excluding the Qt projects.



1. Click **Finish** to import all of the selected projects.
2. You can now see all of the projects listed in the *Project Explorer* tab.



Building C/C++ Projects

1. Right-Click on the project in the *Project Explorer*
2. Select the build configuration you want to use
 - For Release builds: **Build Configurations -> Set Active -> Release**
 - For Debug builds: **Build Configurations -> Set Active -> Debug**
3. Select **Project -> Build Project** to build the highlighted project
 - **NOTE:** You can use **Project -> Build All** to build all of the projects in the *Project Explorer*

Installing C/C++ Projects

There are several methods for copying the executable files to the target file system:

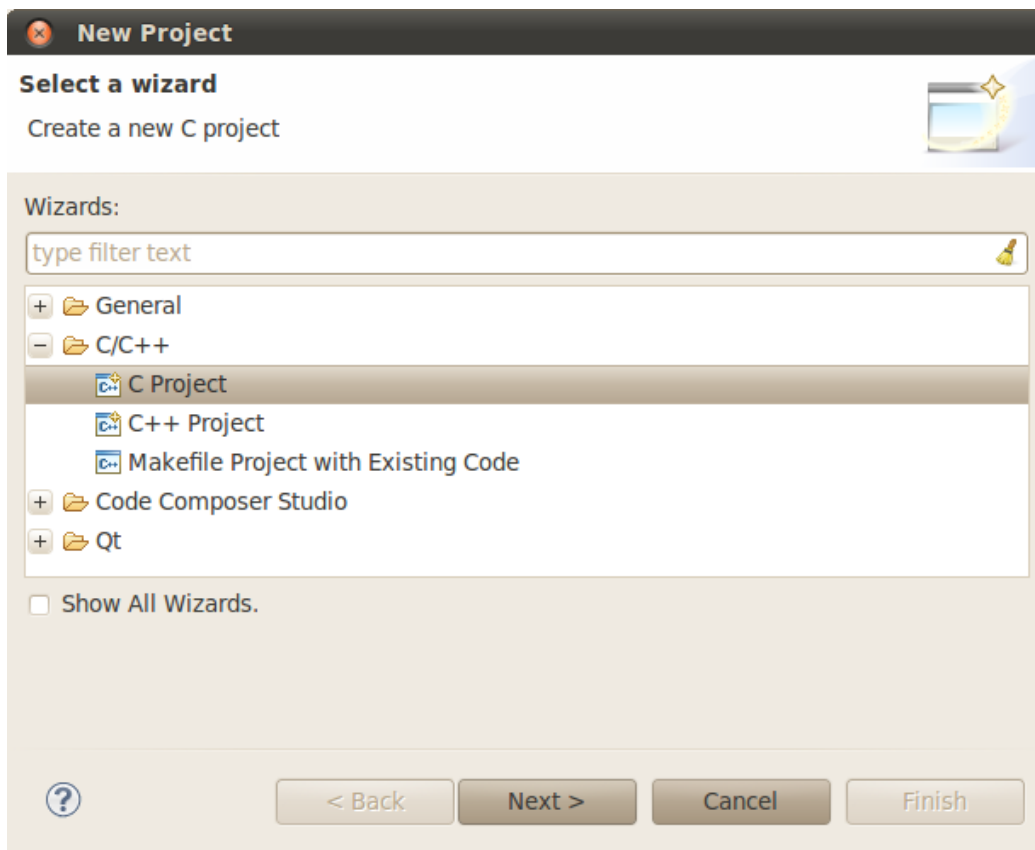
1. Use the top-level Makefile in the SDK install directory. See the **Top-Level Makefile** section for details of using the top-level Makefile to install files to a target file system.
 - **NOTE:** The top-level Makefile uses the install commands in the component Makefiles and can be used as a reference for how to invoke the install commands.
2. For all file system types you can also transfer the file using the drag-and-drop method of Remote System Explorer. See the **Remote System Explorer** section below for more details

Creating a New Project

This section will cover how to create a new cross-compile project to build a simple *Hello World* application for the target.

Configuring the Project

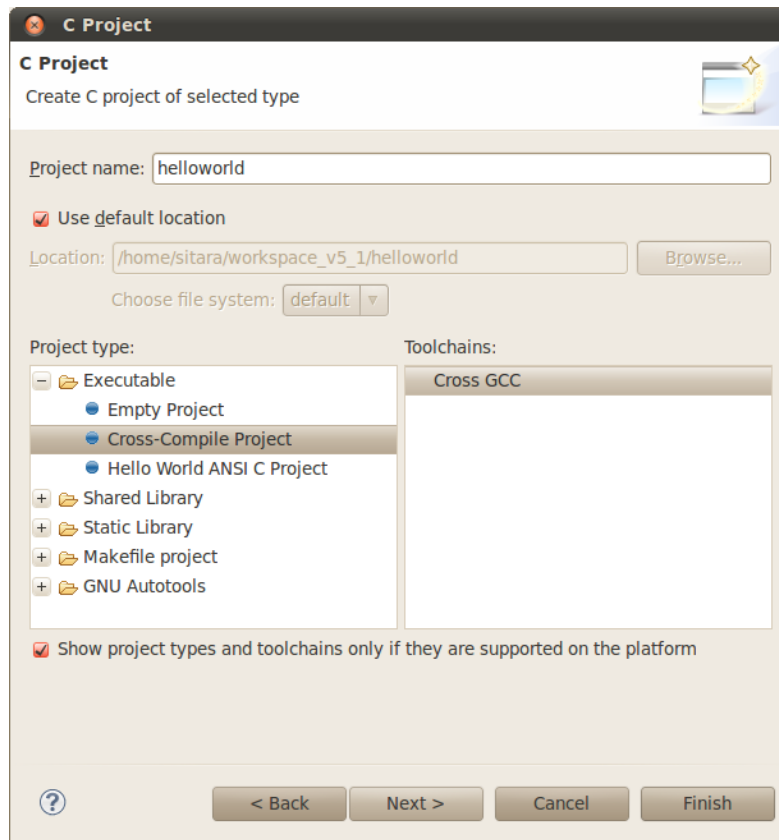
1. From the main CCSv5 window, select **File -> New -> Project...** menu item
2. in the **Select a wizard** window select the **C/C++ -> C Project** wizard



3. Click **Next**
4. In the **C Project** dialog set the following values:

Project Name: **helloworld**

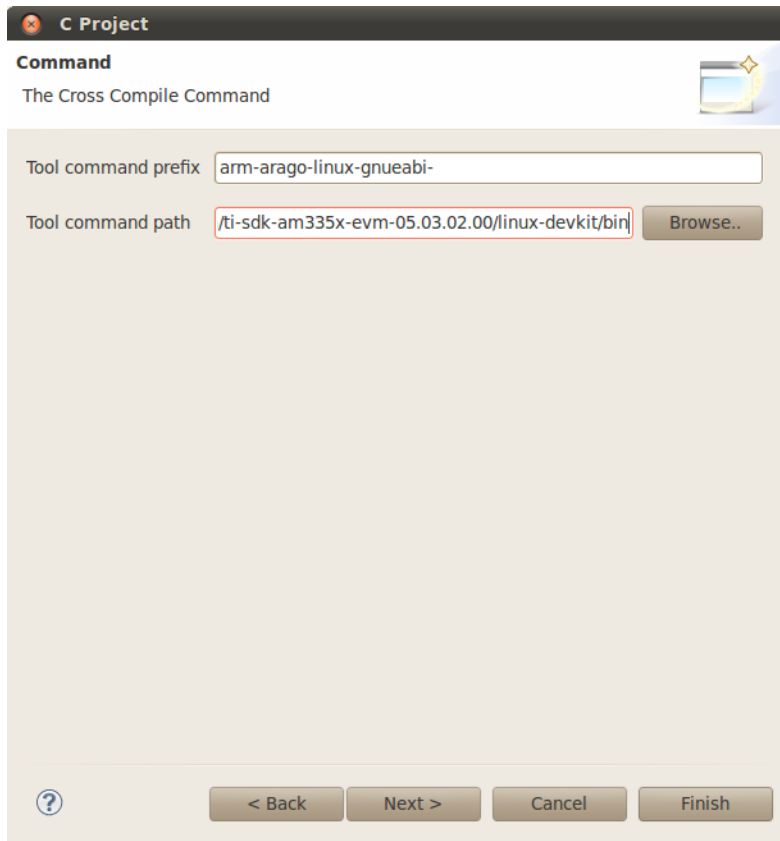
Project type: **Cross-Compile Project**



5. Click **Next**
6. In the **Command** dialog set the following values:

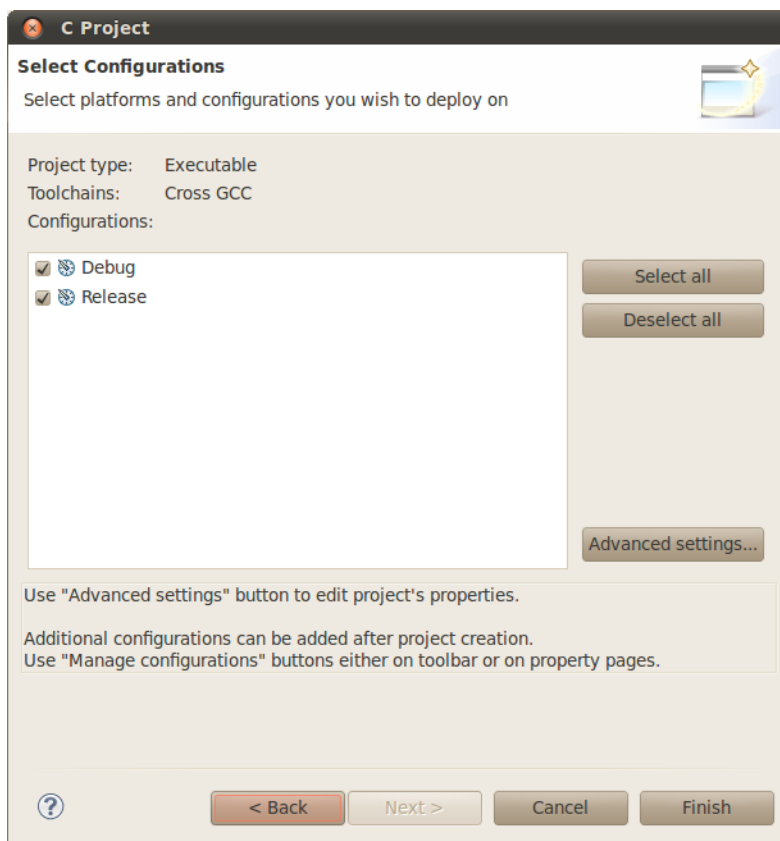
Tool command prefix: **arm-arago-linux-gnueabi-**. Note the the prefix ends with a "-". This is the prefix of the cross-compiler tools as will be seen when setting the *Tool command path*

Tool command path: **<SDK INSTALL DIR>/linux-devkit/bin**. Use the *Browse..* button to browse to the Sitra Linux SDK installation directory and then to the **linux-devkit/bin** directory. You should see a list of tools such as *gcc* with the prefix you entered above.



7. Click **Next**

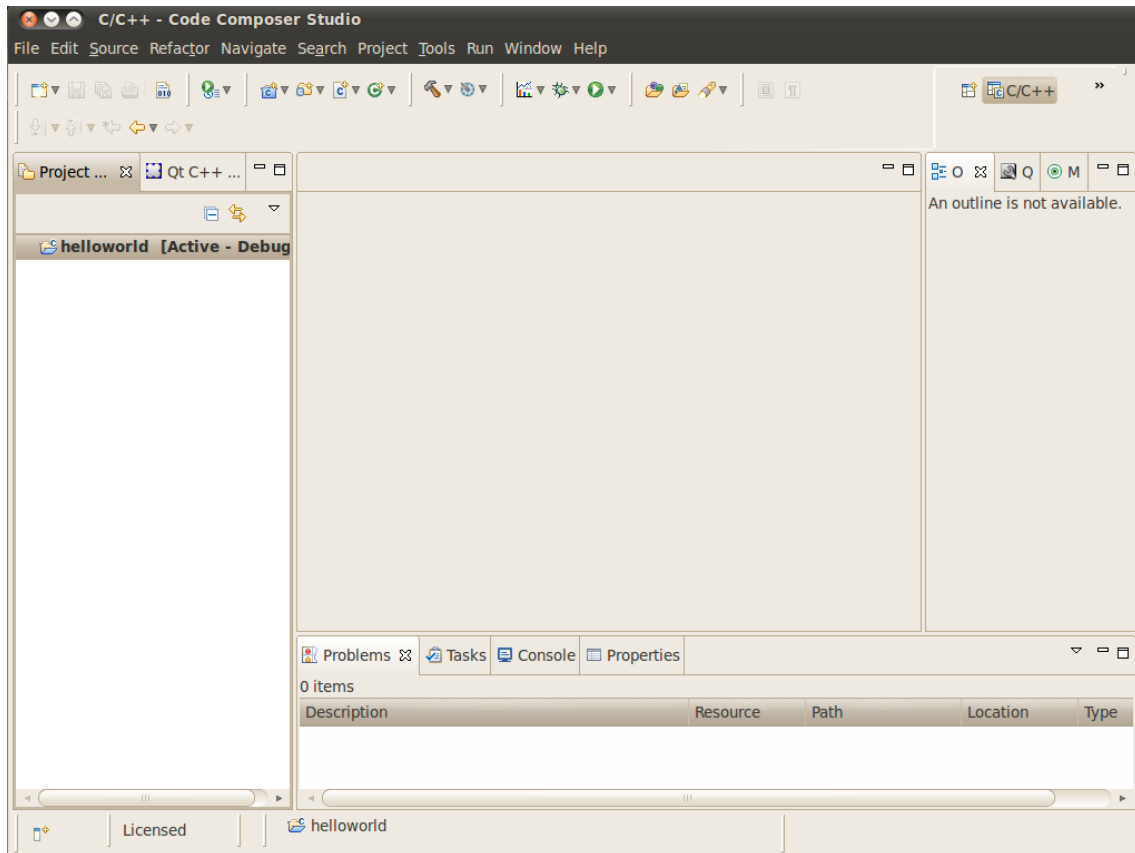
8. In the **Select Configurations** dialog you can take the default *Debug* and *Release* configurations or add/remove more if you want.



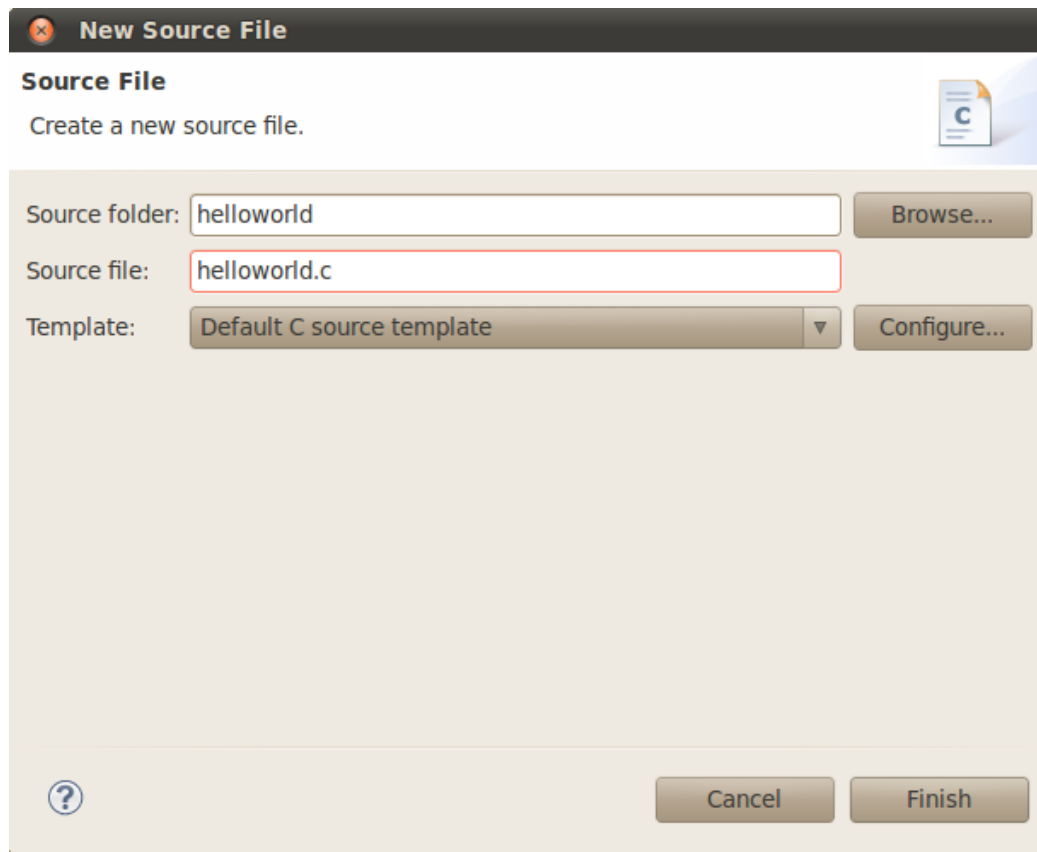
9. Click **Finish**

Adding Sources to the Project

1. After completing the steps above you should now have a **helloworld** project in your *CCS Project Explorer* window, but the project has no sources.



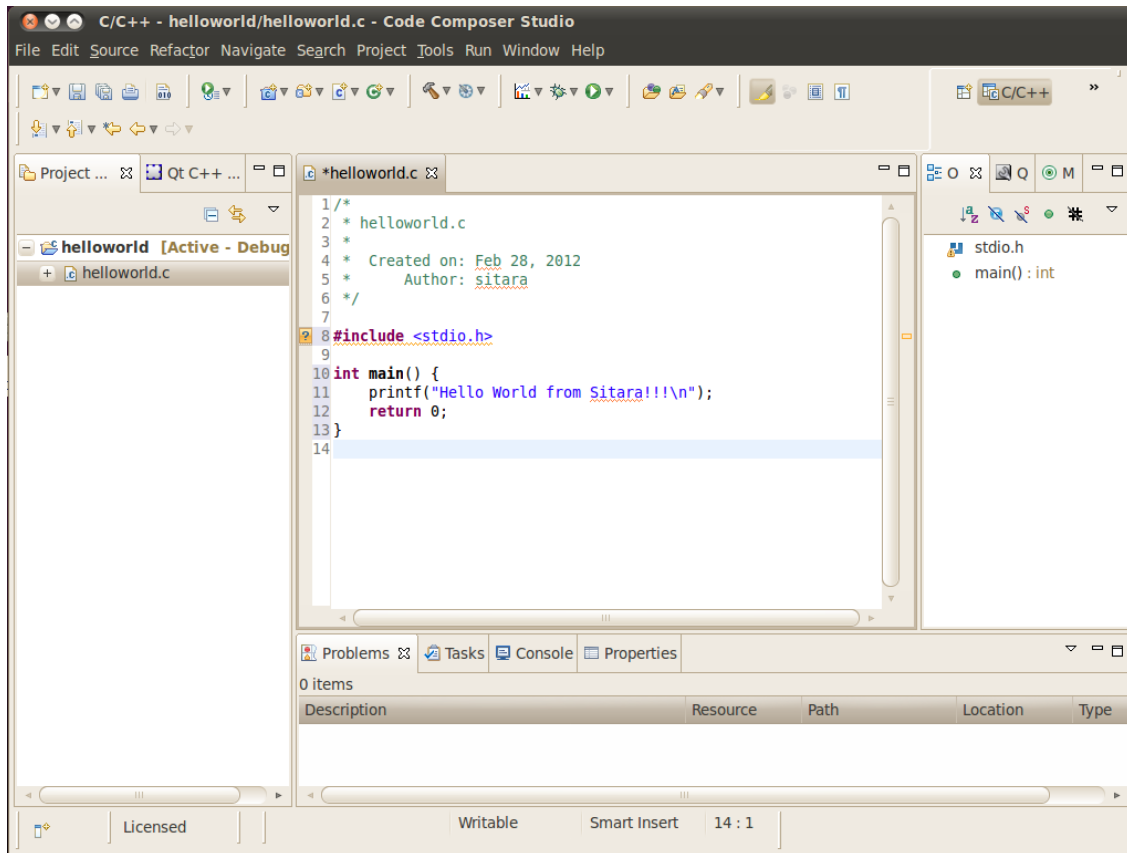
2. From the main CCSv5 window select **File -> New -> Source File** menu item
3. In the **Source File** dialog set the *Source file:* setting to **helloworld.c**



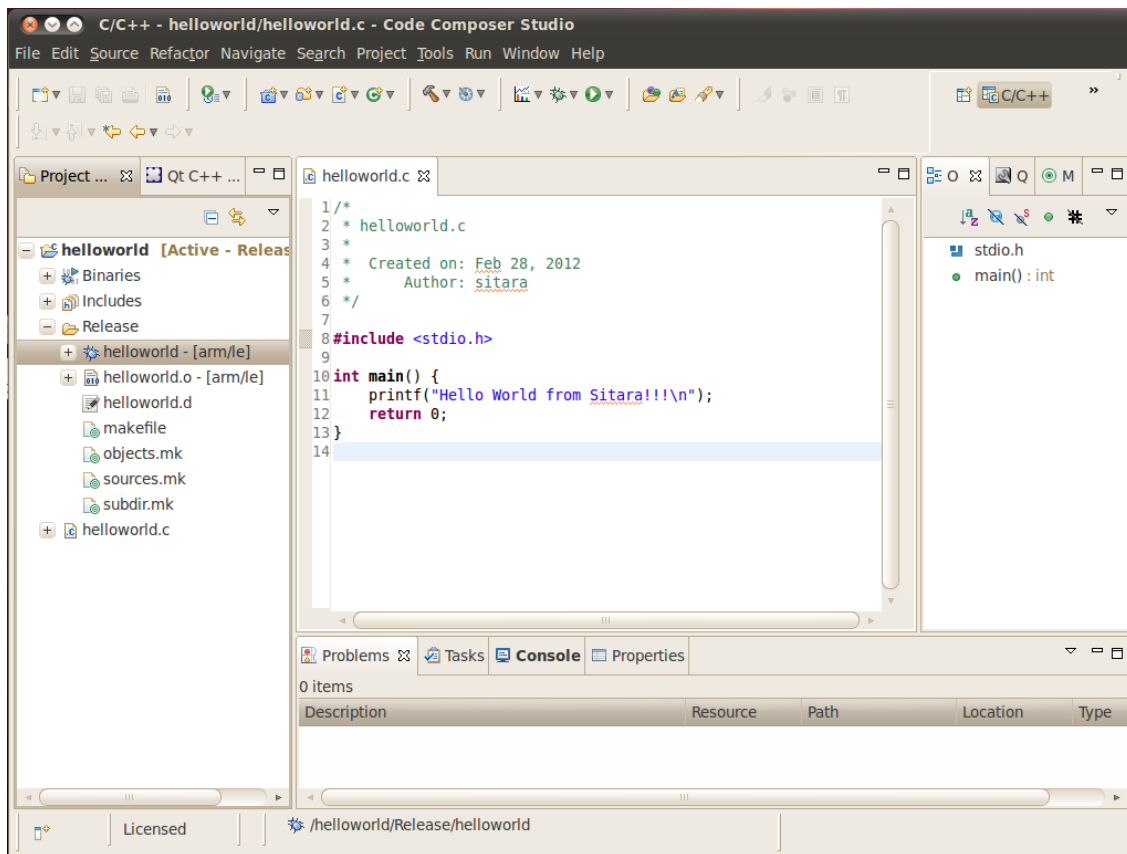
4. Click **Finish**

Cross-Compiling the Sources

1. After completing the steps above you will have a template **helloworld.c** file. Add your code to this file like the image below:



2. Change the build configuration to **Release** by selecting **Project -> Build Configurations -> Set Active -> Release**
3. Compile the **helloworld** project by selecting **Project -> Build Project**
4. The resulting executable can be found in the *Release* directory



5. You can now install the executable to the target file system using **Remote System Explorer**, NFS, or any other method you want.

Remote System Explorer

CCSv5 as installed with this SDK includes the Remote System Explorer (RSE) plug-in. RSE provides drag-and-drop access to the target file system as well as remote shell and remote terminal views within CCS. Refer to [How to Setup and Use Remote System Explorer](#) to establish a connection to your target EVM and start using RSE.

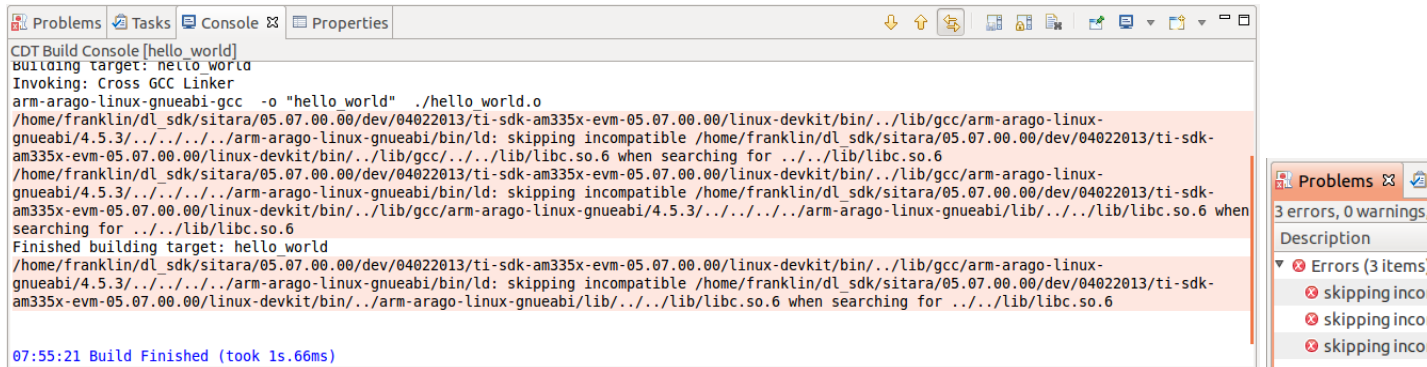
Using GDB Server in CCSv5 for Linux Debugging

In order to debug Linux code using Code Composer Studio v5, you first need to configure the gdbserver on both the host and target (EVM) side.

Please refer to [Running GDB Server on CCSv5](#) for more information.

Known Issues

When using the toolchain included in the 05.07 release Code Composer Studio will display the below errors:



These skipping incompatible messages are harmless toolchain warning messages. This is a known issue planned to be fixed in the next release after the 05.07 SDK. Code Composer Studio will report these messages as errors but will still successfully compile the application.

Archived versions

- [Sitara SDK 05.03 CCSv5 User Guide \(archived\)](#) ^[2]
- [Sitara SDK 05.02 CCSv5 User Guide \(archived\)](#) ^[3]

References

- [1] http://processors.wiki.ti.com/index.php/Sitara_Linux_SDK_Setup_Script
- [2] http://processors.wiki.ti.com/index.php?title=Code_Composer_Studio_v5_Users_Guide&oldid=84244
- [3] http://processors.wiki.ti.com/index.php?title=Code_Composer_Studio_v5_Users_Guide&oldid=68253

Linux Debug in CCSv5

Background

Linux Debug Overview

CCSv5 supports run mode debug (a.k.a. remote GDB debug, agent-based debug, application debug) and stop mode debug (a.k.a. JTAG debug, low-level debug). For Linux aware debug support (an extension of the stop mode debug), please read the section Linux Aware Debug below.

- In run mode debug, the user can debug one or more Linux processes. On the host side, CCSv5 launches a cross platform GDB debugger to control the target side agent (a GDB server process). The GDB server launches or attaches to the process to be debugged and accepts instructions from the host side over a serial or TCP/IP connection. The Linux kernel remains active during the debug session. The user can only examine the state of the processes being debugged.
- In the stop mode debug, CCSv5 halts the target using a JTAG emulator. The Linux kernel and all processes are suspended completely. The user can examine the state of the target and the execution state of the current process.

IMPORTANT! This page refers to CCS version 5.1.0 and newer. For CCSv5.0.x check this page.

Run Mode Debug

Dependencies

The following dependencies apply to Run Mode Debug:

- CCS versions: CCSv5.1 or greater
- Devices: any core that is capable of running Linux: Cortex-A, ARM9, C66x.
- Host requirement: a cross platform GDB debugger (typically part of a GCC package like CodeSourcery or Arago)
- Target requirement: a GDB server that is compatible with the GDB debugger located on the host (typically part of a SDK package like EZSDK, DVSDK, etc.)
- A GCC project (see How to create GCC projects in CCSv5).

The run mode debug requires two connections to the target system:

1. One connection to the target console is used to execute Linux commands.
 - If using a serial port (common in all TI's EVMs and low-cost boards like Beagleboard and Pandaboard), this connection can be done using a simple terminal program like Hyperterminal, Putty, TeraTerm or even a CCSv5 terminal plug-in.
 - If using Ethernet, this connection must be done using one of the programs above and configuring it for telnet or SSH. Keep in mind that the linux running on the target board requires a telnet or SSH server running on it.
2. The other connection is used by the gdb debugger to communicate with the gdb server running on the target.
 - This connection can be done either via Ethernet or serial port. Keep in mind the speed of a serial connection can be a lot slower and timeouts may occur.

Procedure

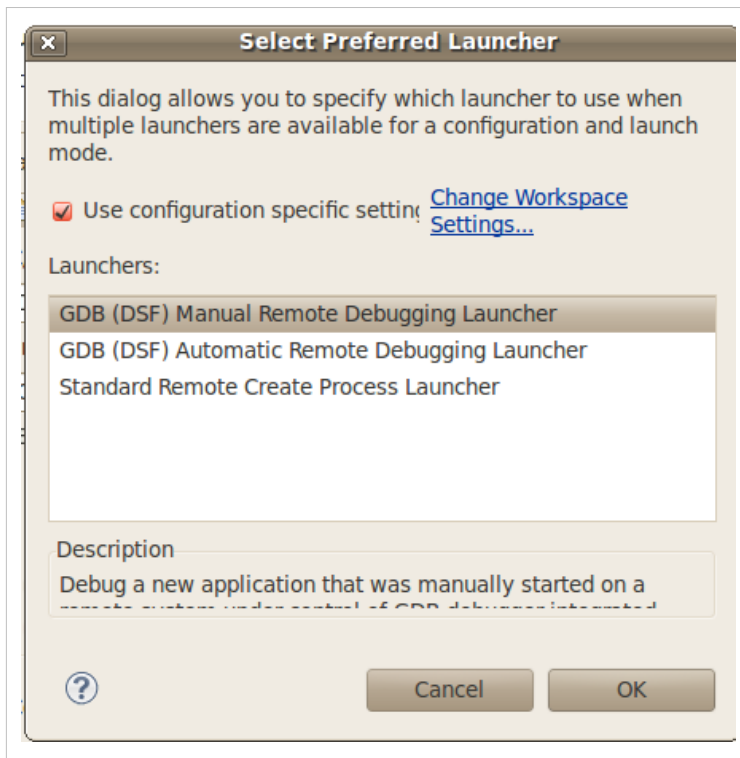
IMPORTANT! In certain versions CCSv5 does not enable "CDT GDB Debugging" configurations. You need to enable them from the Capabilities tab in the Preference dialog (select *Window --> Preferences --> General --> Capabilities*).

1. Bring up the Debug Configurations dialog by selecting menu *Run --> Debug Configurations*
 2. Select *C/C++ Remote Application*
-

3. Click on the button *New launch configuration* (Top left of the pane)
4. Set the fields *Project* and *C/C++ Application:* respectively to the existing project in the workspace and the binary executable file

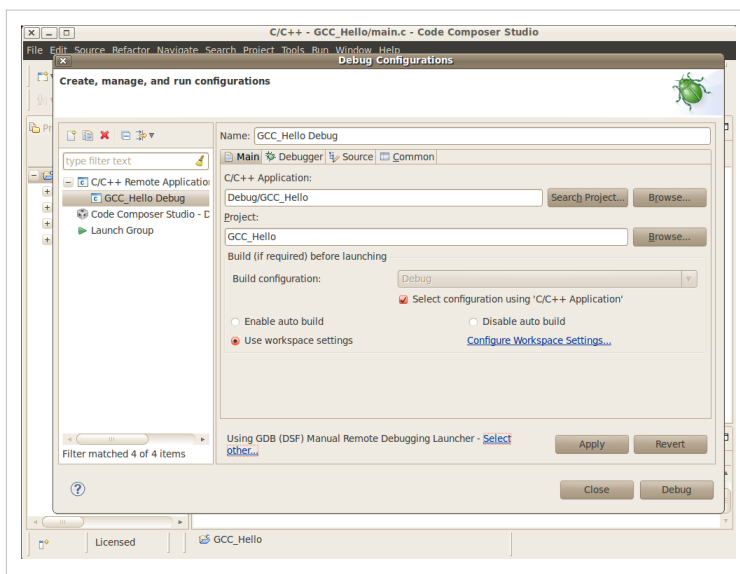
Note: If the project is already in focus (Active or highlighted) in the *Project Explorer* view, these fields will be already populated.

5. In tab *Main*, click on the link *Select Other* at the bottom where it says **Using GDB (ASF) Automatic Remote Debugging Launcher**. Check *Use configuration specific settings* and select **GDB (DSF) Manual Remote Debugging Launcher**. Click **OK**.



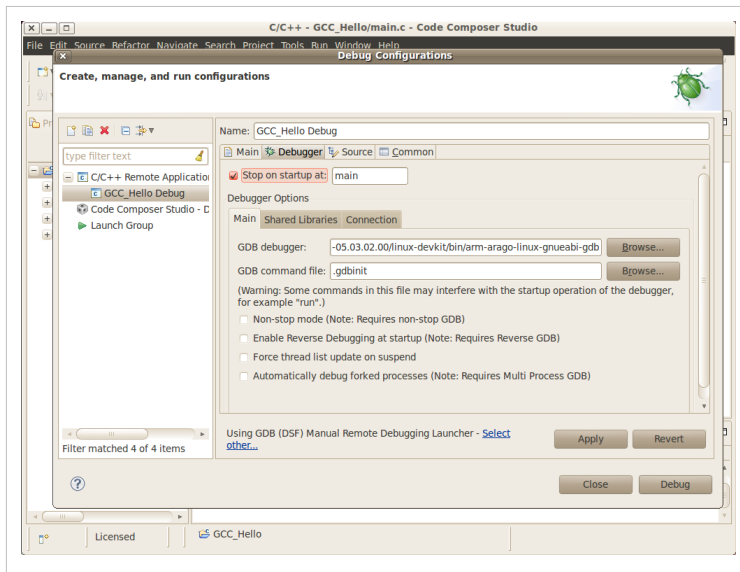
Note: It is possible to set up CCSv5 to automatically connect and launch the debugger in the target by leaving the settings above untouched. Check section 8 of the Eclipse CDT FAQ ^[1].

Note: Other options like *Enable auto build*, *arguments* and others can be modified at this time.



6. Select the **Debugger** tab and specify the GDB debugger. In this case the GDB debugger from Arago is being used, but it is possible to use also CodeSourcery or other toolchain

In this example it was used the path
 /home/user/ti-sdk-am335x-evm-05.03.02.00/linux-devkit/bin/arm-arago-linux-gnueabi-gdb

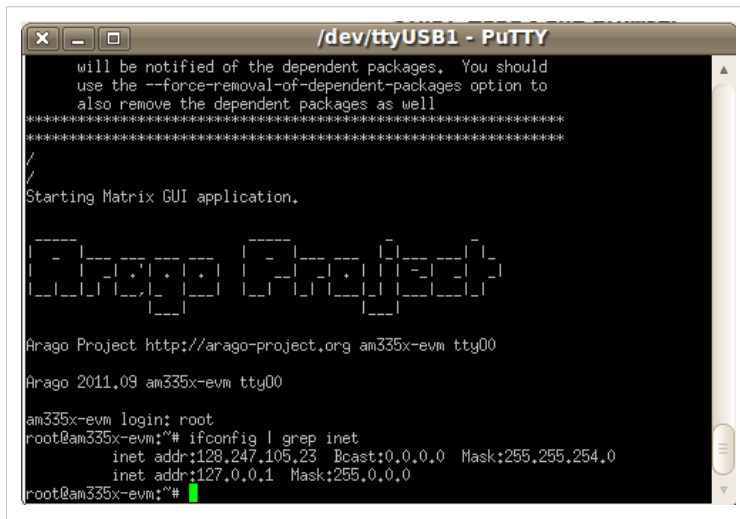


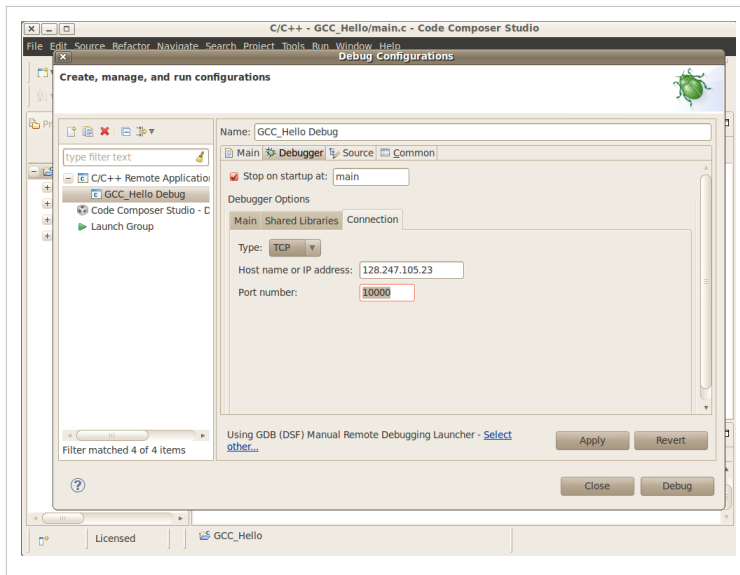
7. On the Debugger **Connection** tab, specify the IP address and port of the GDB server running on the target.

Note: the port number is arbitrary and is specified when the gdbserver is launched - unless you have a strong reason to change it, the value of 10000 is just fine.

Note: the IP address of the target can be determined from the target linux console.

IMPORTANT! Some SDKs do not have gdbserver installed by default in the supplied filesystem. Check the SDK documentation for details on how to install it.

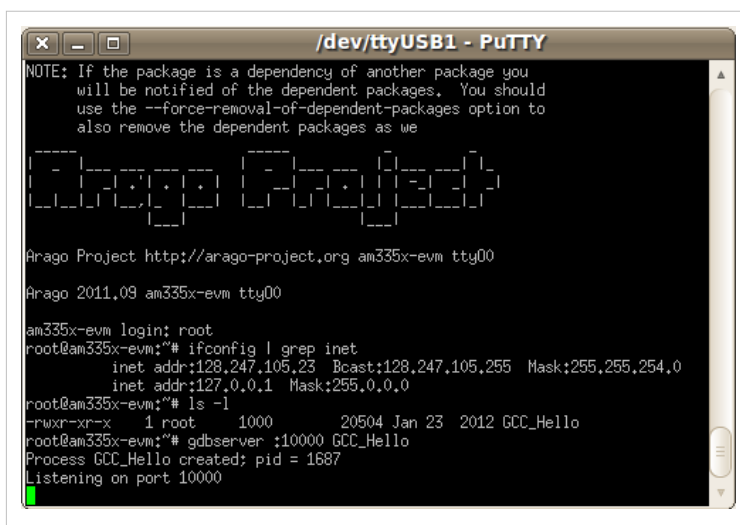




- On the target console, start the GDB server specifying the application file and the port number.

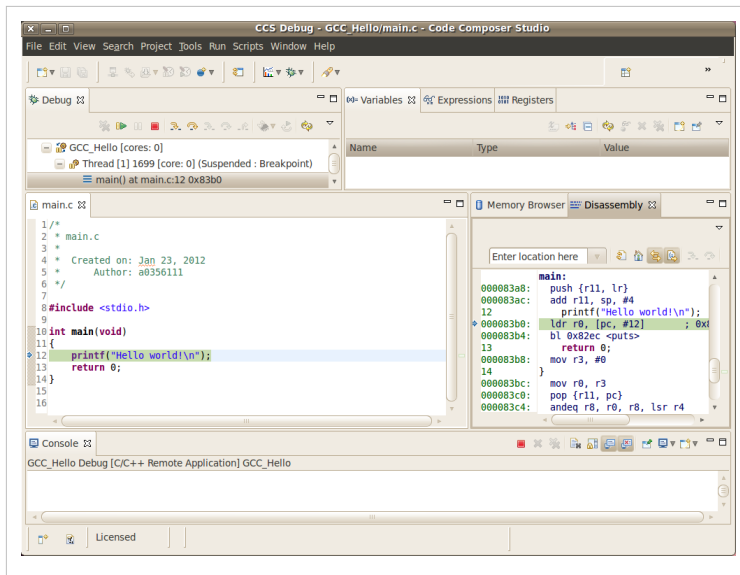
Note: make sure the port number matches the one specified in the Debugger Connection tab (10000 by default).

Note: the application under debug must be located on the target filesystem. This can be done in multiple ways: either copying it to the shared NFS directory, to the SD card being used to boot linux, etc.



- Launch the debug configuration by clicking the Debug button.

- CCSv5 will launch the GDB debugger to connect to the GDB server.
- After the connection is established, you can step, set breakpoints and view the memory, registers and variables of the application process running on the target.



10. You may need to set the shared library (object) search path in a cross compile debug environment.

- Under *Debug Configuration* -> *Debugger* tab -> *Shared Libraries* tab enter the path to the target filesystem lib directory
- You may need a copy of the target filesystem on the local debug host

Stop Mode Debug

Dependencies

The following dependencies apply to Stop Mode Debug:

- CCS version 5.1.0 or greater. This facilitates working on either a Windows host, or a Linux host.
 - In addition to the procedure below, a short video clip is located here ^[2].
- Devices: any core that is capable of running Linux: Cortex-A, ARM9, C66x.
- Host system requirements:
- Target system requirements: a Linux distribution running on the target. Kernel releases 2.6.x and 3.1.x were tested.

The stop mode debug requires a JTAG connection to the target system. It supports either a standalone JTAG emulator (XDS100, XDS510, XDS560) or an embedded emulator on the development board (OMAPL137EVM, Beaglebone, etc.)

An additional connection to the target console is helpful to monitor the Linux boot procedure and the integrity during the debug session.

Procedure

Although it is possible to connect to the device using the JTAG emulator without any reference to the source code, this makes the debugging process very difficult as the information in the debugger will consist in pure assembly code. In order to perform low-level debugging with complete visibility of the Linux kernel source code, a few steps are necessary:

1. Compile the kernel with the appropriate debug symbols (EABI executable file `vmlinux`).
2. Create a project in the CCS workspace that contains all Linux kernel source code.
3. Create a debug configuration that loads the debug symbols to the debugger and references the source code in the Linux kernel tree.

Compiling the Linux kernel with debug information

The Linux kernel must be built with debugging information, otherwise no source code correlation can be made by the debugger.

In order to add or verify if the debug symbols are properly added to the configuration, the step *make menuconfig* must be performed before the kernel is built, and the options below must be enabled:

- Enable *Kernel hacking* --> *Compile the kernel with debug info*

Also, if the kernel is in experimental mode, you should enable the option below:

- *Kernel hacking* ---> *Enable stack unwinding support*

To check if the kernel is in this mode, check if the option below is enabled.

- *General Setup* ---> *Prompt for development and/or incomplete code/drivers*

Note: for kernel 3.1.0 and above, there is an additional option that must be set:

- *Kernel Hacking* ---> *Enable JTAG clock for debugger connectivity*

Note: for kernel 3.2.0, the option *Enable stack unwinding support* shown above is only available if the kernel is built with ARM EABI support. To enable it, go to:

- *Kernel Features* ---> *Use the ARM EABI to compile the kernel*

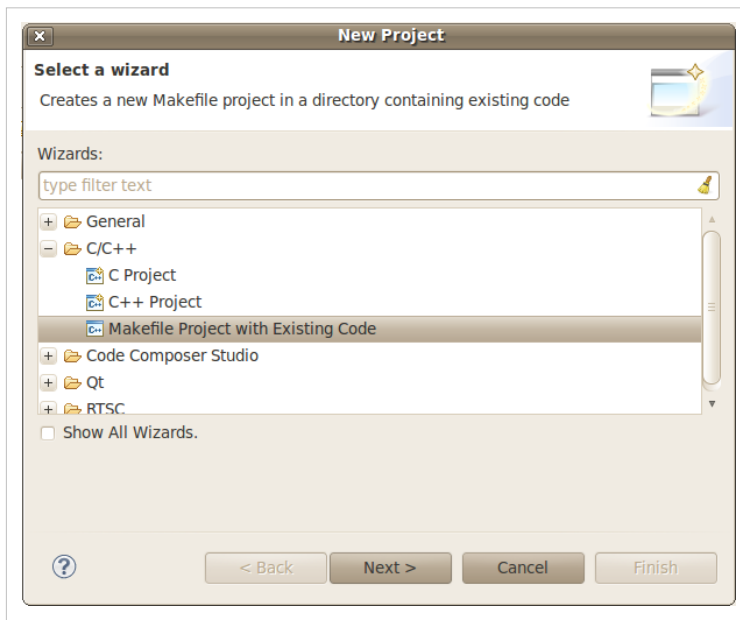
Note: for kernel 3.2.0, the option *Compile the kernel with debug info* shown above is only available if the option *Kernel Debugging* is enabled. To do it, go to:

- *Kernel hacking* ---> *Kernel Debugging*

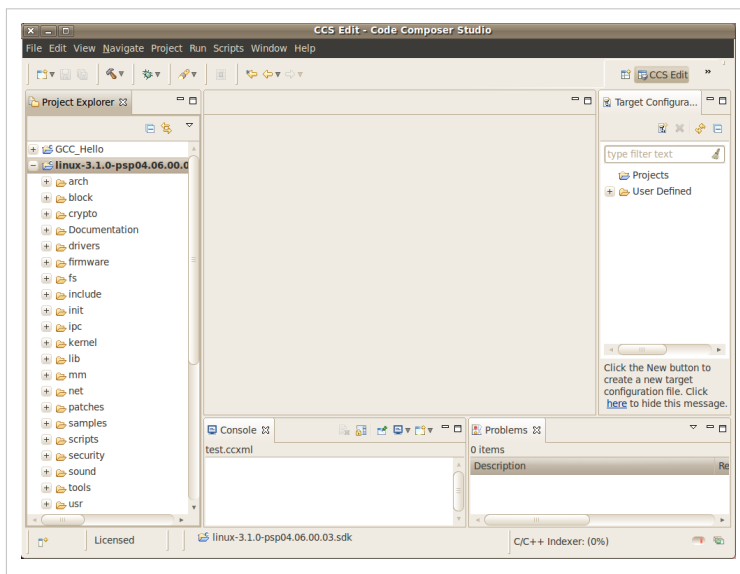
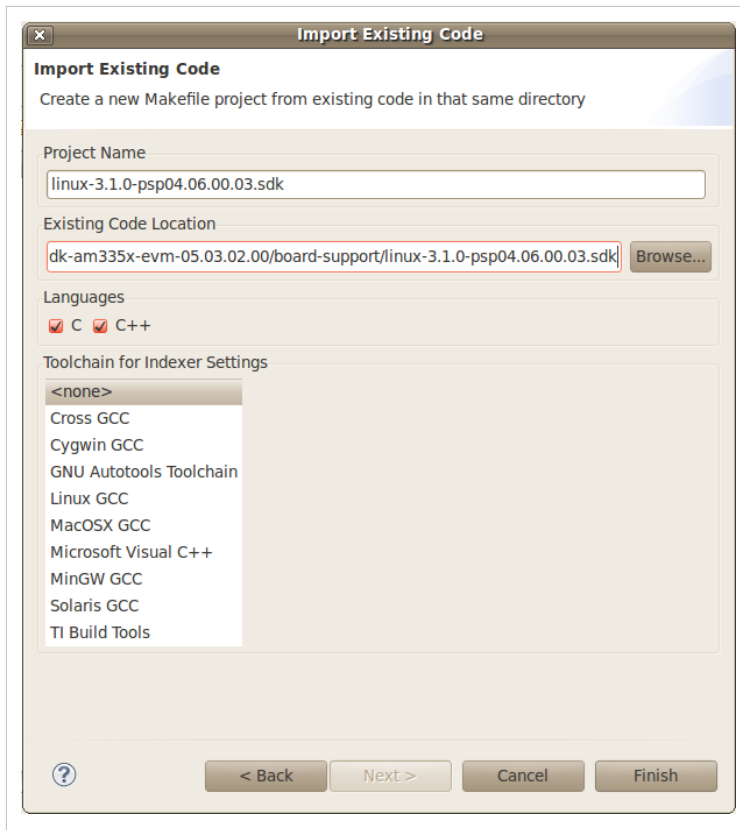
Note: the building process depends on the Linux distribution being used, therefore it is recommended to read the SDK documentation regarding this step.

Creating a source code project for the kernel

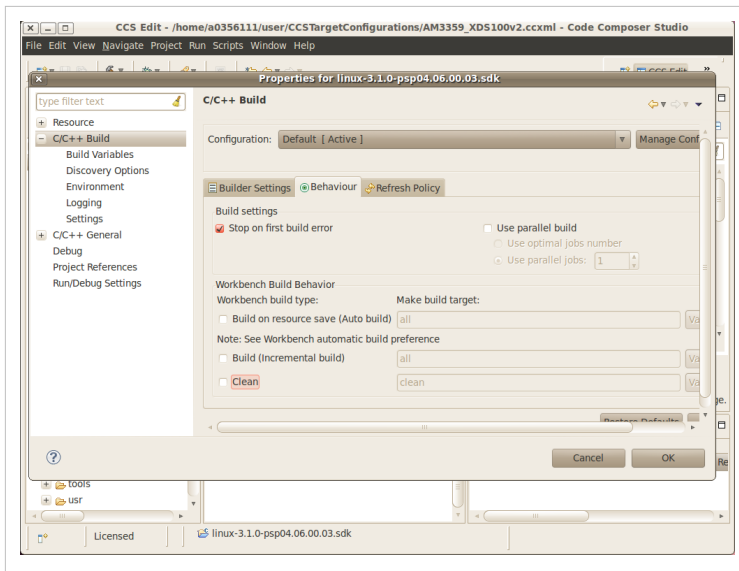
1. Create a new C/C++ project by selecting *File* --> *New* --> *Project* and select *Makefile Project with Existing Code*. Click *Next*.



2. In the section *Existing Code Location*, click on *Browse...* and point to the root directory of the Linux kernel source tree. Leave the toolchain as *<none>* and click *Finish*.



3. To prevent CCS from building the Linux kernel automatically before launching the debugger, this option must be disabled. Highlight the Linux kernel project in the *Project Explorer* view, right click and select *Build Options...*, then select *C/C++ Build* in the left tree and the tab *Behaviour*. Uncheck all the build rules boxes and click *OK*.



Note: it is possible the C-syntax error checker built into Eclipse is also activated, which may throw errors while launching the debugger. It can be configured by right-clicking on the project --> *Build Options...* --> click on *Show Advanced Settings* --> *C/C++ General* --> *Code Analysis*. It can also be completely disabled by going to the submenu *Launching* and then unchecking the box *Run as you type (selected checkers)*.

Associating the Kernel Project with the Target

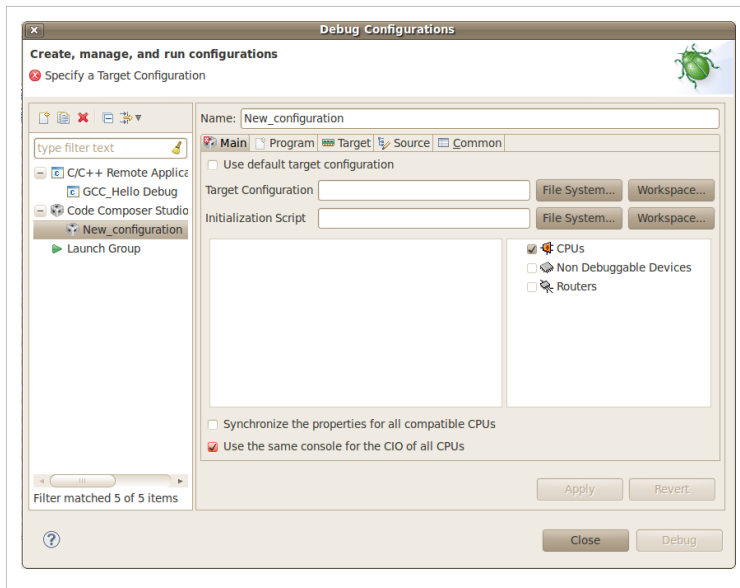
At this point, a target configuration file (.ccxml) that corresponds to your emulator and board must be ready.

In this example a Beaglebone (AM3359) was used, together with the Sitara support package available at the CCS download page.

Note: check the Getting Started Guide to learn how to create one.

Important! When debugging a target running any High-level OS (Linux, WinCE, Android, etc.) or its support/initialization routines (u-boot, WinCE bootloader, etc.) you should not rely on GEL files in the target configuration (.ccxml) for device and peripheral initializations that will disrupt your environment. Details on how to add/remove GEL files are shown in the section *Advanced target configurations* --> *Adding GEL files to a target configuration* of the CCSv5 Getting Started Guide.

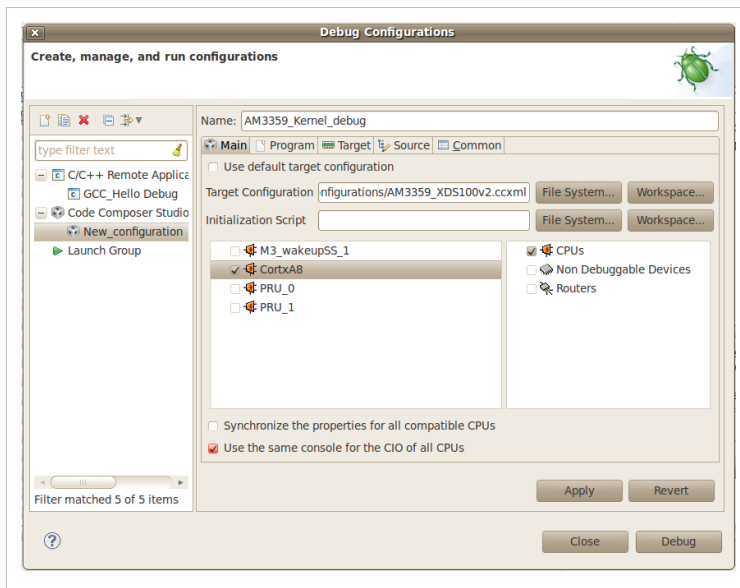
1. Select menu *Run* --> *Debug Configurations*
2. Select *Code Composer Studio - Device Debugging* and click on the button *New Launch configuration* at the top left.



3. Click on the button *File System...* near the box *Target Configuration* to select the target configuration file (.ccxml) for your hardware.

Optional: give a meaningful name for the Debug Configuration at the box *Name*:

Optional: depending on the target configuration, at this point a list of cores will be shown and can be disabled to improve the debugger performance.



4. Select the tab *Program* to assign the Linux kernel source code to the Debug configuration.

5. On the drop-down menu *Device* select the core where the Linux is running.

In this example the core **Texas Instruments XDS100v2 USB Emulator_0/CortexA8** was selected

6. Click on the button *Workspace...* near the box *Project* to select the Linux kernel project

In this example it was used the project `linux-3.1.0-psp04.06.00.03.sdk`

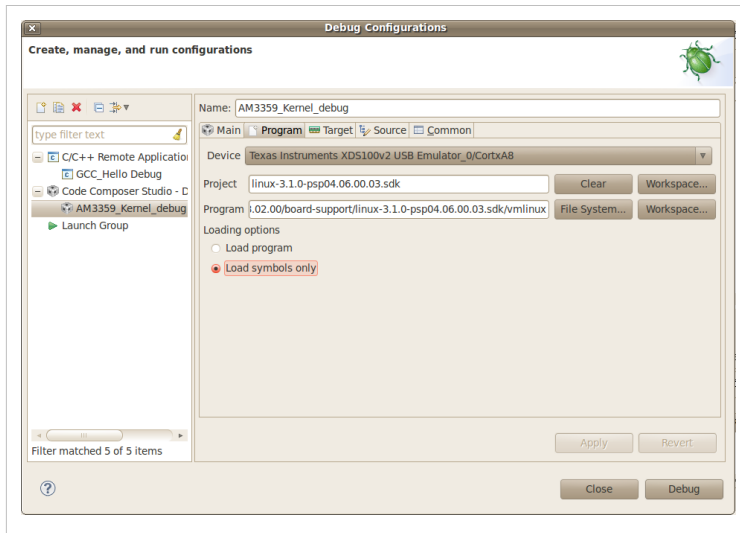
7. Click on the button *File System...* near the box *Program* to select the EABI executable `vmlinux` that contains the debug symbols

Note: If the Linux kernel was rebuilt, the location of this file is usually in the main directory of the Linux kernel source tree.

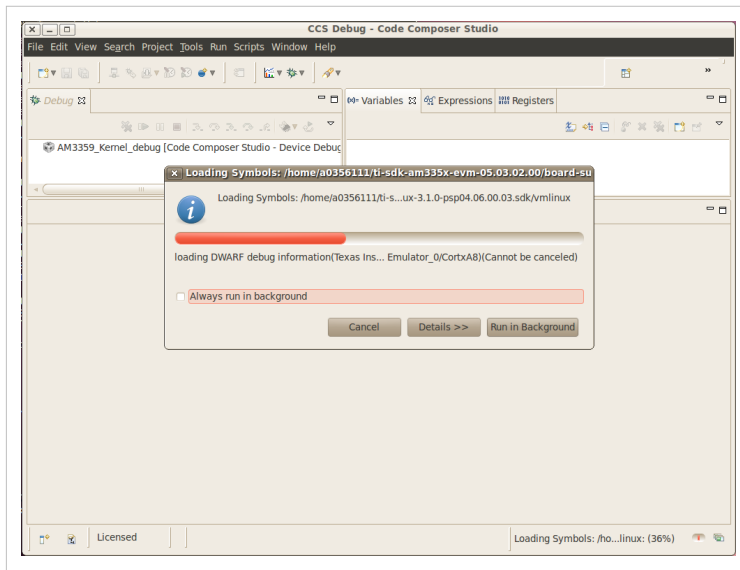
In this example the location selected is the same as the Linux kernel project:
`</tt>/home/user/ti-sdk-am335x-evm-05.03.02.00/board-support/linux-3.1.0-psp04.06.00.03.sdk</tt>`

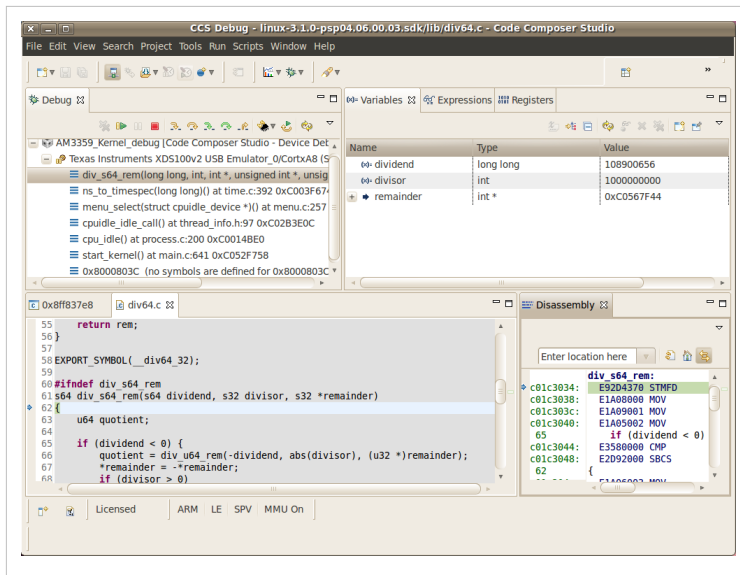
Important! It is common that a file `vmlinux` is also provided in the boot partition of the SD card shipped with the development board (where the file `uImage` is also located). However, check its size; if it is relatively small when compared to `uImage` (3, 4 times larger) it is possible it does not carry debug information. A typical size for the `vmlinux` file usually starts at 30~40MB.

- At last, check the box *Load symbols only*. Click *Apply*.



- Now the debug session is ready to be launched. At this point, the emulator must be connected, the target board powered up and Linux running (typically in the command prompt). Click on the *Debug* button.





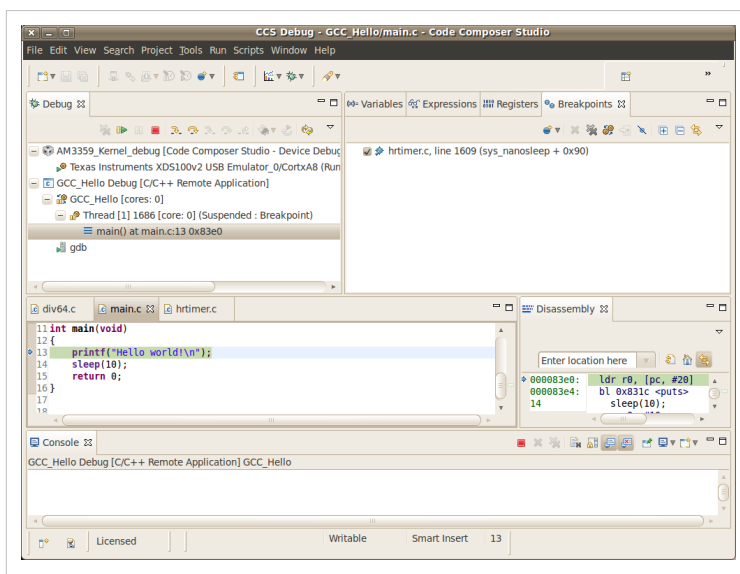
Mixed Mode Debug

The stop mode debug can be used concurrently with the run mode debug.

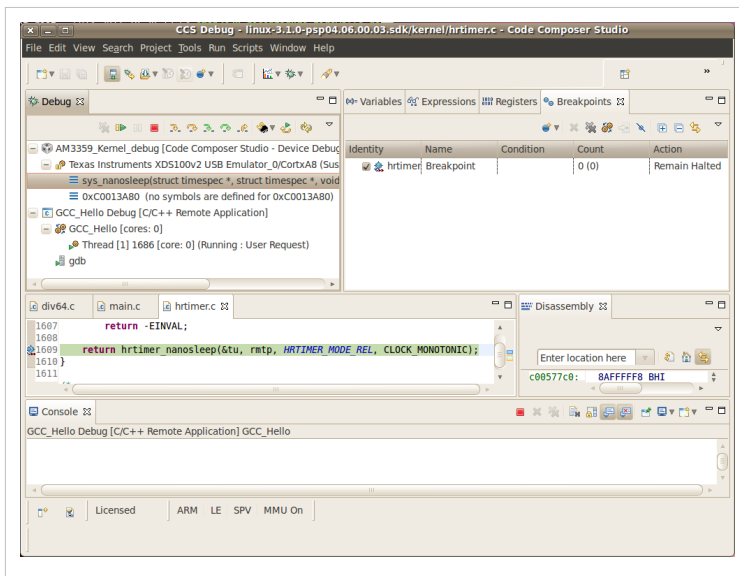
The user can set breakpoints in the user process using the run mode debug and breakpoints in the kernel using the stop mode debug.

To demonstrate this, a call to the function `sleep()` is added to the Linux application used earlier in the Run mode debug and a breakpoint is added to the function `sys_nanosleep()` (file `<kernel/hrtimer.c>`). This will provoke a halt on the breakpoint set in the Stop Mode debug caused by a function call from the Linux application in the Run mode.

1. Search for the function call `hrtimer_nanosleep()` on the file `<kernel/hrtimer.c>` that belongs to the Linux kernel project.
2. With the Stop mode debug session still running, halt the target. Right-click on the line of the call, select *Breakpoint (Code Composer Studio)* then *Hardware Breakpoint*. Resume the target execution.
3. Start a Run mode debug session with the application that has the `sleep()` function call. After launching, the *Debug* view should show two debug sessions as in the screen below:



- Put the target to run. When the application calls `sleep()` the Stop mode debug session should halt at the breakpoint, as shown in the screen below:



Important! Keep in mind that halting the Linux kernel while GDB/GDBserver are running may cause communication timeouts, clock skews or other glitches inherent from the fact that the host system and other peripherals are still running.

Linux Aware Debug

This feature was not ported to CCSv5.1 due to compatibility break with the standard Eclipse (required significant changes that would penalize other debug features), lack of popularity and overall performance (speed and memory usage to refresh and store all processes at every breakpoint).

To date there is not estimate to implement an "add-on" tool to CCSv5.1. Please check back regularly for updates.

Limitations and Known Issues

- When performing Run Mode debug, by default Eclipse looks in the host PC root directory for runtime shared libraries, thus failing to load these when debugging the application in the target hardware. The error messages are something like:

warning: .dynamic section for "/usr/lib/libstdc++.so.6" is not at the expected address (wrong library or version mismatch?)

warning: .dynamic section for "/lib/libm.so.6" is not at the expected address (wrong library or version mismatch?)

warning: .dynamic section for "/lib/libgcc_s.so.1" is not at the expected address (wrong library or version mismatch?)

warning: .dynamic section for "/lib/libc.so.6" is not at the expected address (wrong library or version mismatch?)

[New Thread 1701]

To correct this you must create a *gdb command file* (.gdbinit) in a text editor (gedit, vim, etc.) and add the following line to it:

set solib-absolute-prefix /home/user/targetfs (or wherever the target filesystem is located)

Close any GDB debugging sessions. Open the *Debug Configurations* as shown in the Run Time debugging and then browse to this file in the *Debugger* tab --> box *GDB command file*.

Source: <http://www.eclipse.org/forums/index.php/m/503935/>

References

[1] <http://wiki.eclipse.org/index.php/CDT/User/FAQ>

[2] http://software-dl.ti.com/sdo/sdo_apps_public_sw/CCSv5/Demos/Linux_kernel_debugging/Linux_kernel_debugging.html

How to setup Remote System Explorer plug-in

Return to the [Sitara Linux Software Developer's Guide](#)

Overview

Remote System Explorer (RSE) is an Eclipse plug-in that provides:

- Drag-and-drop access to the remote file system
- Remote shell execution
- Remote terminal
- Remote process monitor

Prerequisites

Before you configure RSE you should make sure the following prerequisites are met:

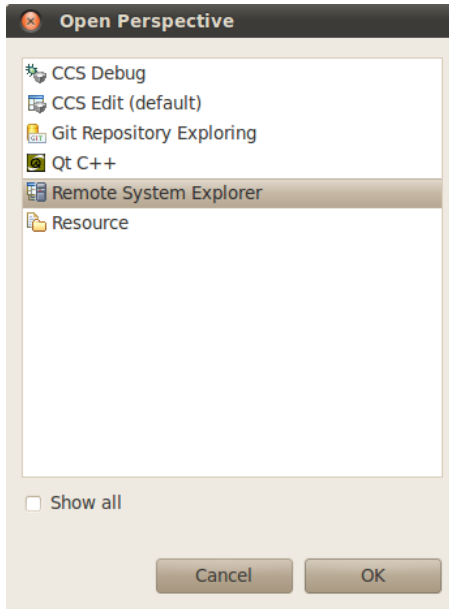
1. The RSE capability is **enabled**
2. The EVM you plan to connect to is powered on and you know the IP address of the EVM.
 - You can obtain the IP address of the EVM using matrix and selecting **Settings -> Network Settings** or by connecting over the serial console and using the **ifconfig** command.

The first time this plug-in is used, a connection to the target EVM needs to be established and configured. Follow the procedures below to setup and configure RSE for use with the Sitara Linux SDK.

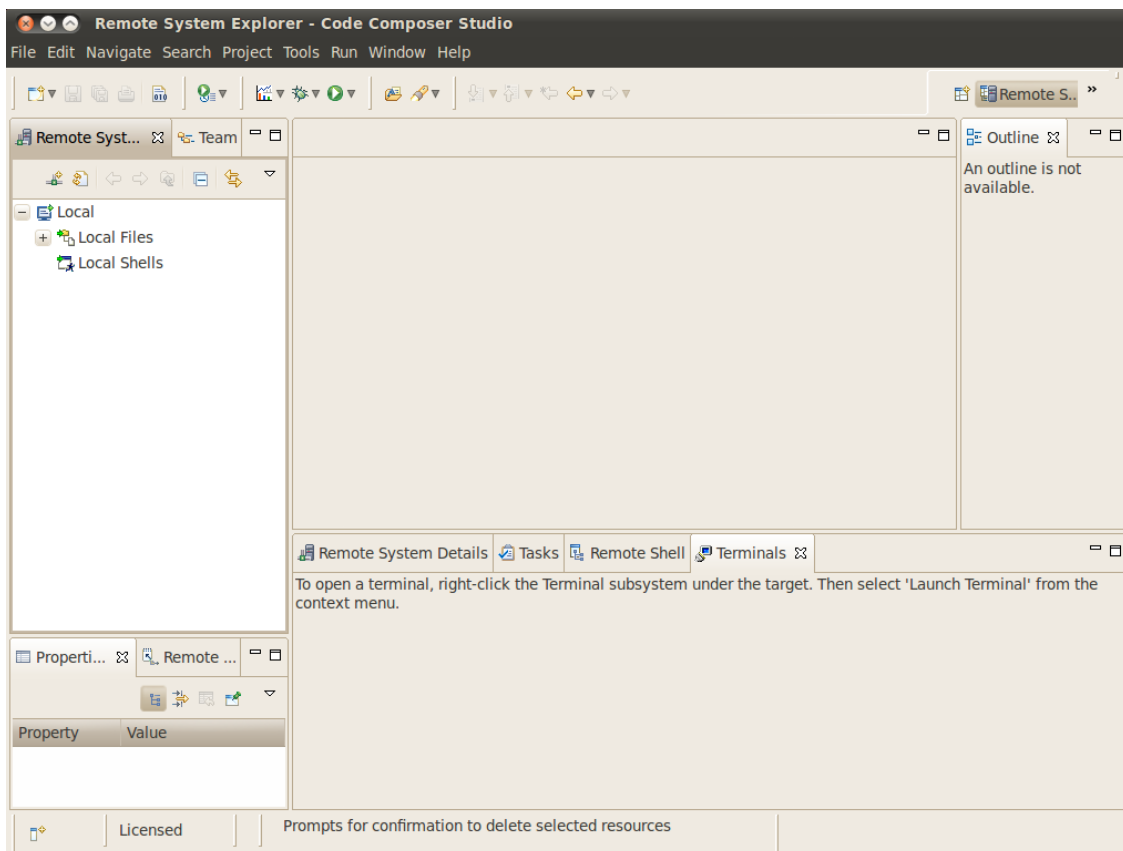
Opening the Remote System Explorer Perspective

1. Go to **Window -> Open Perspective -> Other...**
2. In the menu window select **Remote System Explorer** to open this perspective.

NOTE: If Remote System Explorer does not appear in this list you need to **enable additional CCS capabilities**



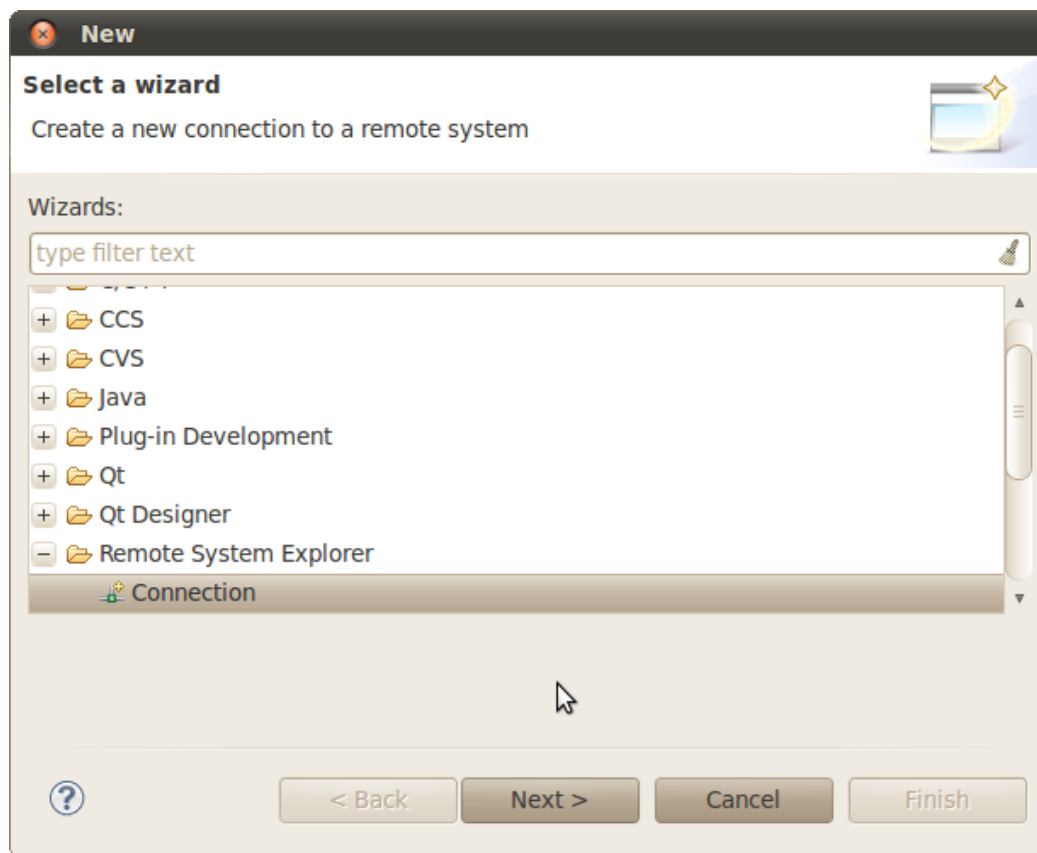
3. Click **OK**
4. You will now have the RSE view opened



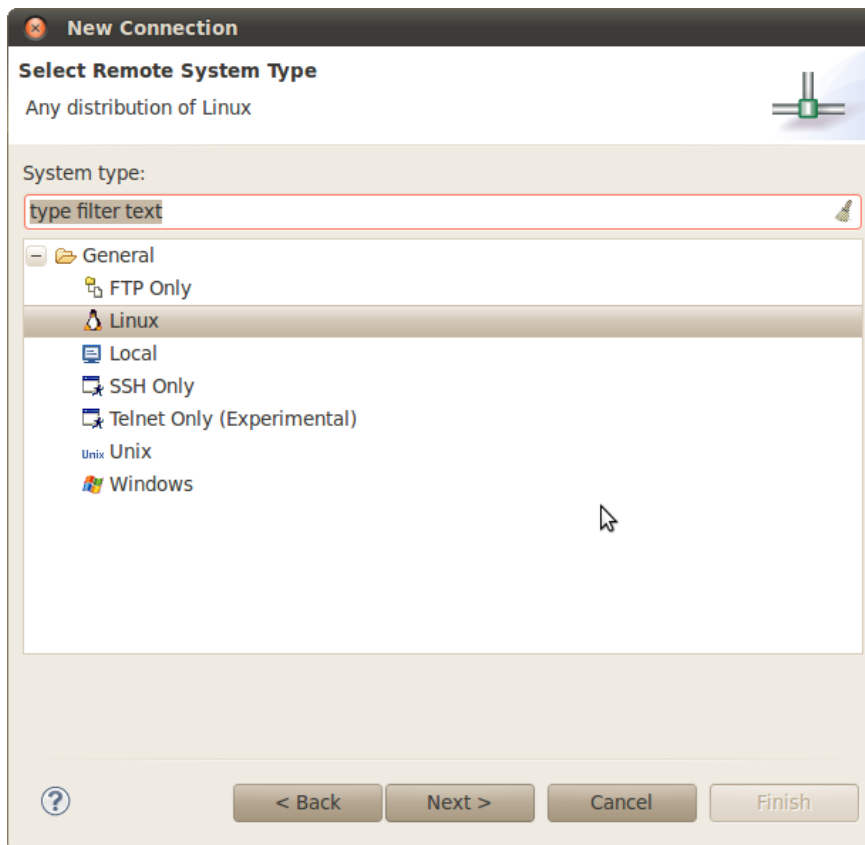
Creating a New Connection

To establish a new connection with the target EVM you must run the New Connection Wizard.

1. Click **File -> New -> Other...**
2. In the **Select a wizard** window select **Remote System Explorer -> Connection**



3. Click **Next**
4. In the **Select Remote System Type** window select the **Linux** system type



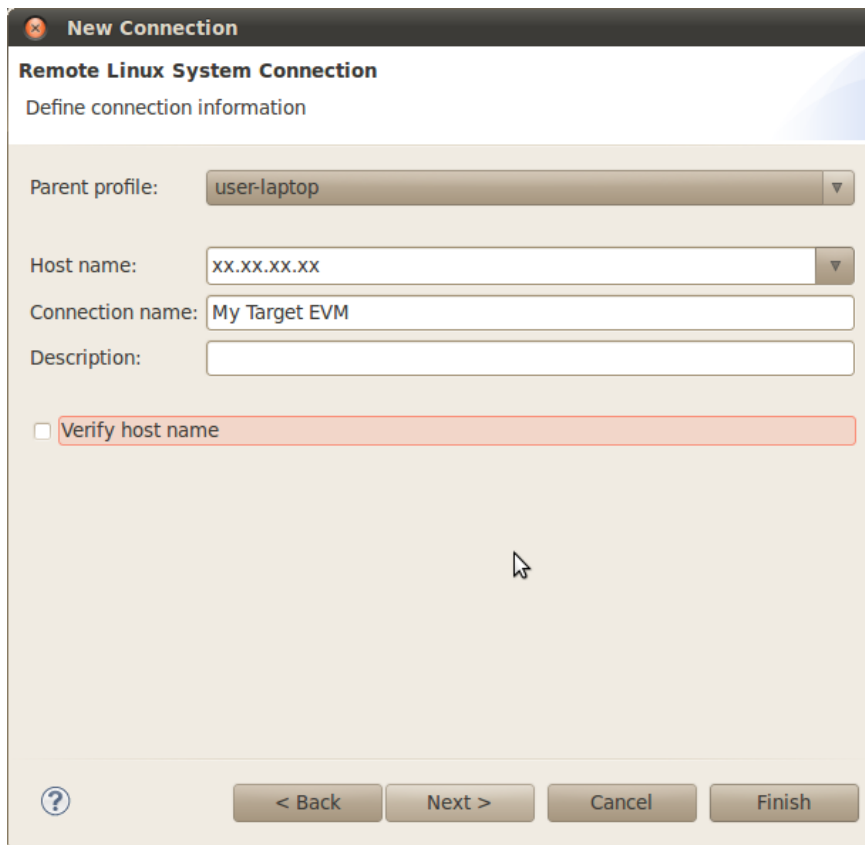
5. Click **Next**

6. In the **Remote Linux System Connection** window enter

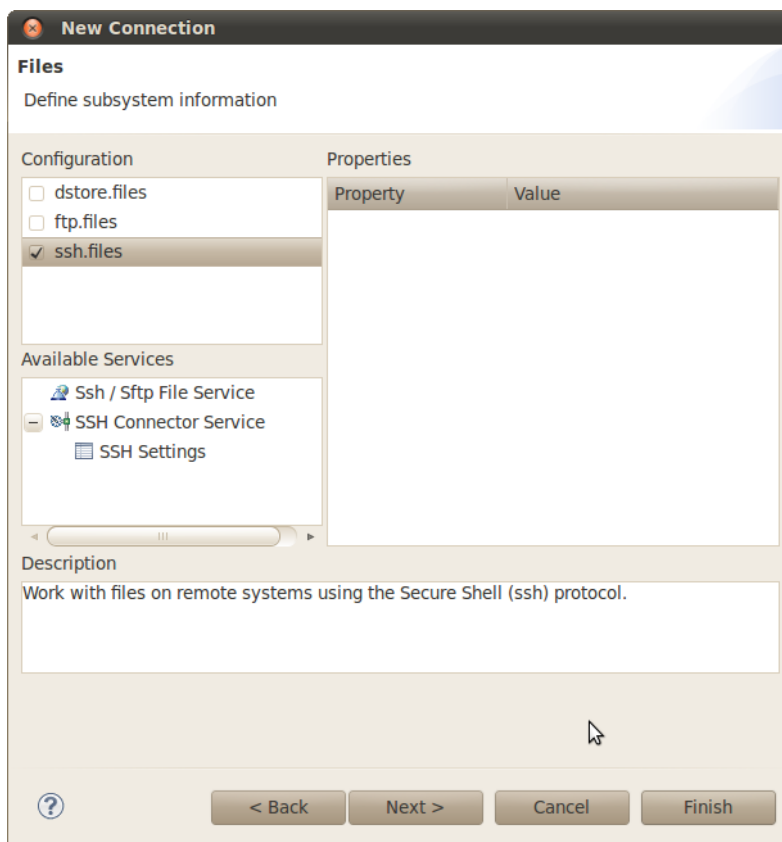
Host name: Enter the IP address of your target EVM. This can be determined as detailed in the **Prerequisites** section above

Connection name: The default value is the same as the host name, but this can be changed to a more human readable value like *My Target EVM*

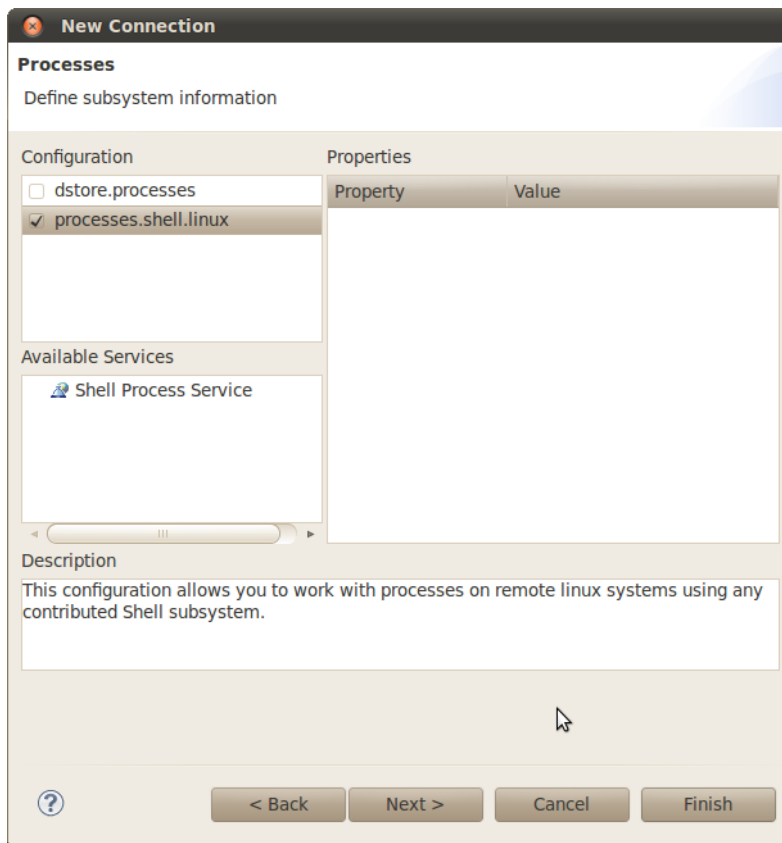
You can un-check **Verify host name** or leave it checked depending on whether you want to verify the IP address you entered for the *Host name* field.



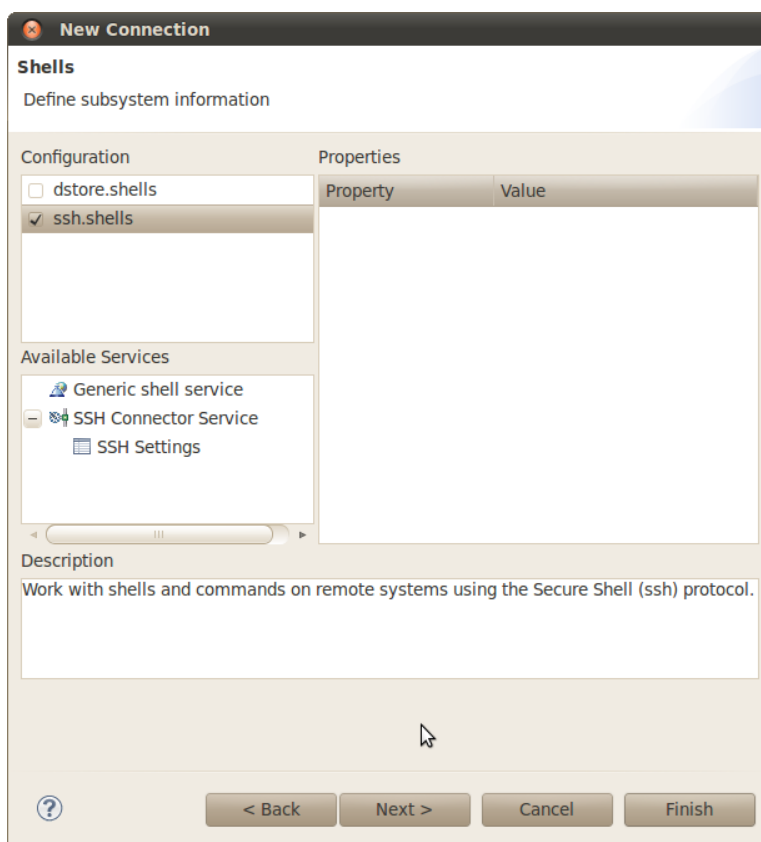
7. Do NOT click the Finish button. Click **Next**
8. Check **ssh.files** to use the *Secure Shell* protocol for communication



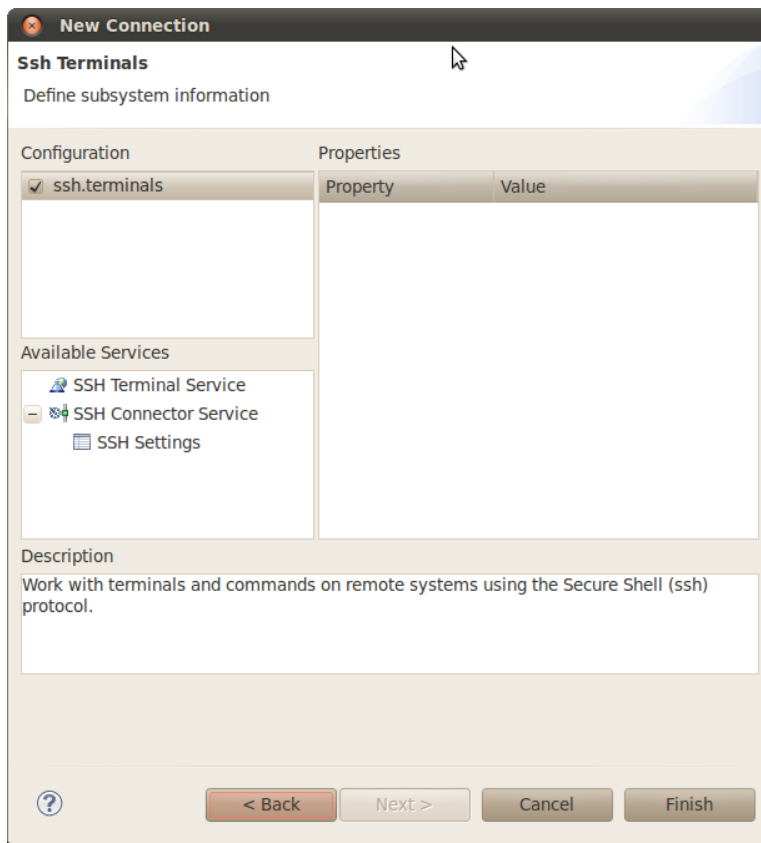
9. Do NOT click the Finish button. Click **Next**
10. Check **processes.shell.linux** to use a shell to work with processes on the remote system



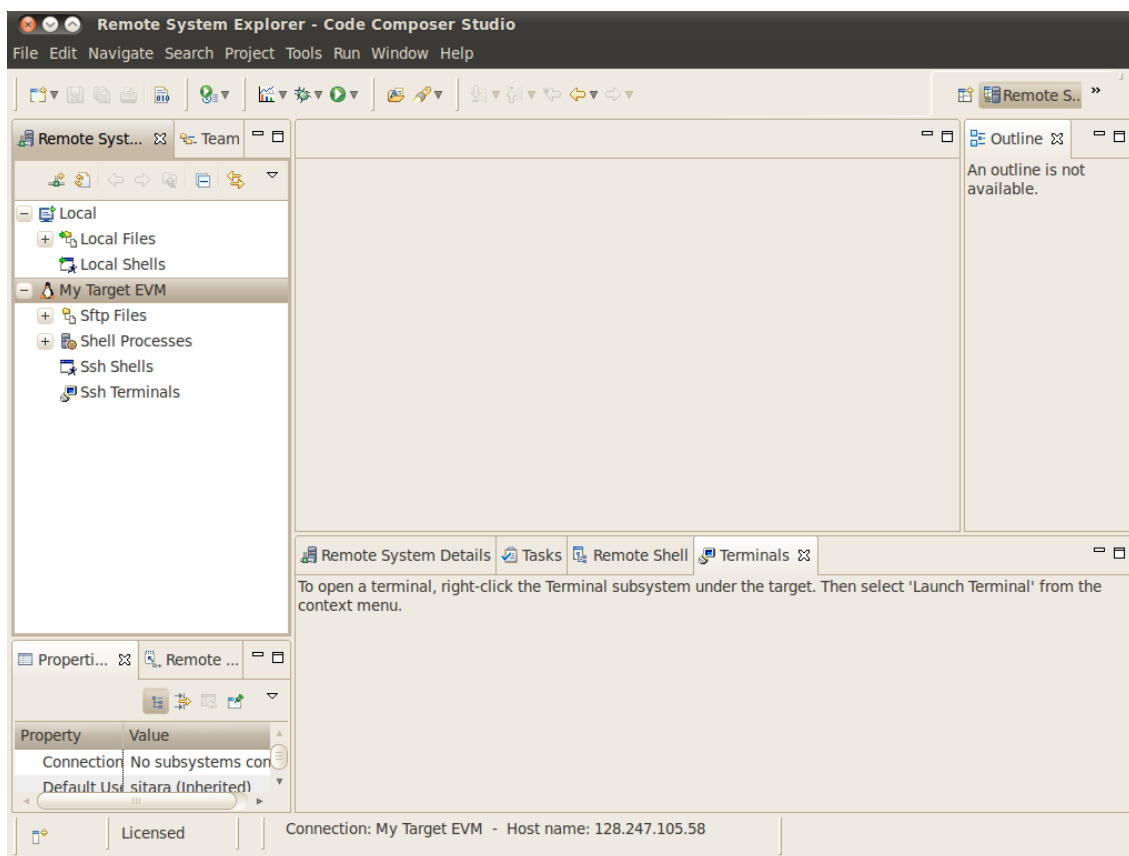
11. Do NOT click the Finish button. Click **Next**
12. Check **ssh.shells** to use *Secure Shell* to work with shell commands



13. Do NOT click the Finish button. Click **Next**
14. Check **ssh.terminals** to use *Secure Shell* to work with terminals



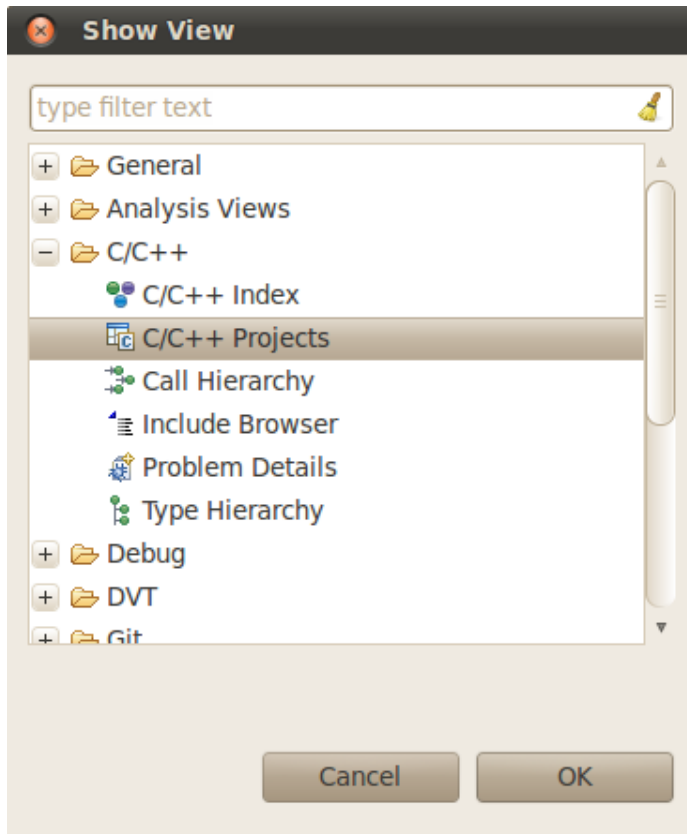
15. Click **Finish**
16. You will now see your EVM configuration in the RSE view



Re-Opening the C/C++ View

If when you enabled RSE and opened the RSE perspective your C/C++ view disappeared you can re-open it using the following commands. This is useful to get back to your projects list to enable copying and pasting files to transfer to the remote system as detailed in the Transferring Files section below.

1. Select **Window -> Show View -> Other...**
2. In the **Show View** dialog select **C/C++ -> C/C++ Projects**

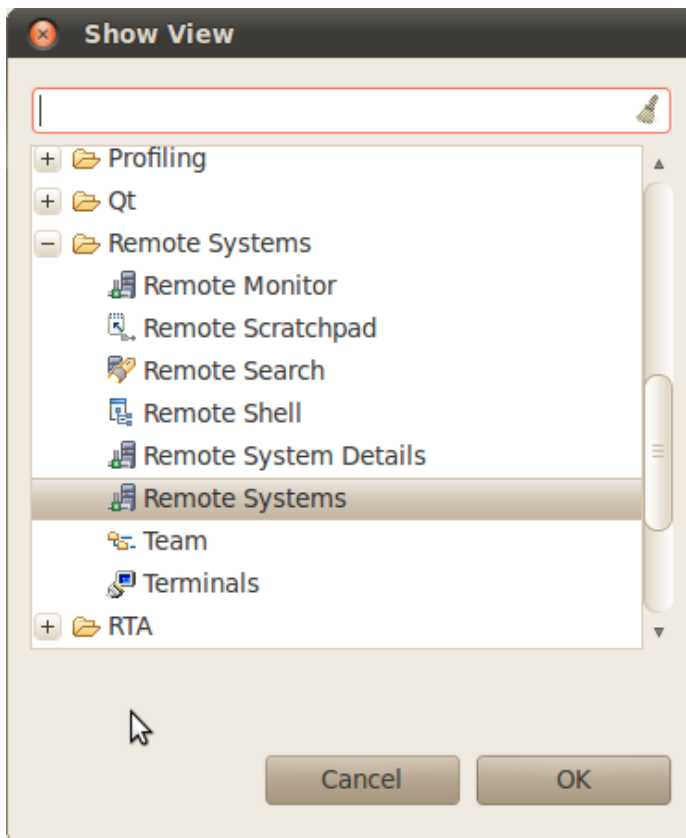


3. Click **OK**
- **NOTE:** If you do not like the location of the C/C++ Projects view you can drag it to another location in CCS by dragging and dropping the Tab.

Re-Opening the Remote System Explorer View

If you have closed the RSE view and wish to re-open it you can use these steps:

1. Select **Window -> Show View -> Other...**
2. In the **Show View** dialog select **Remote Systems -> Remote Systems[]**



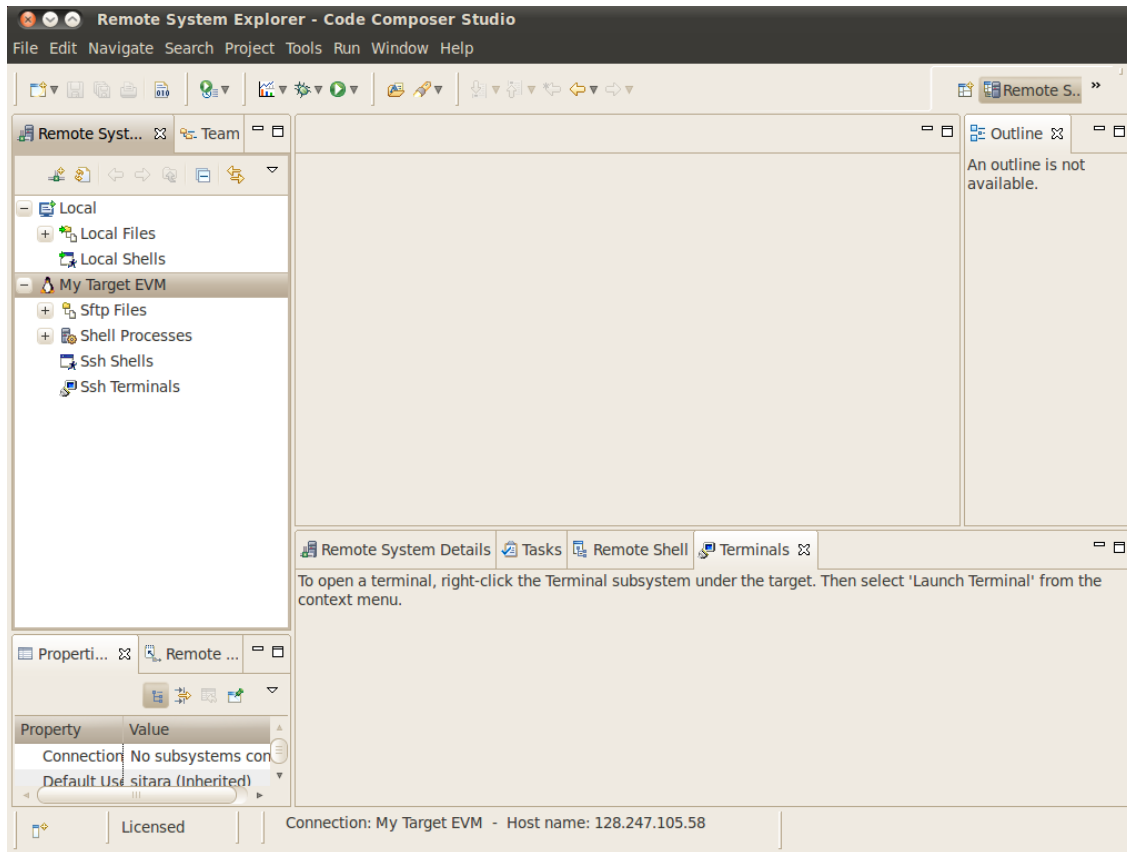
3. Click **OK**
 - **NOTE:** If you do not like the location of the Remote Systems view you can drag it to another location in CCS by dragging and dropping the Tab.
4. A Remote Systems tab appears in the CCS perspective. The target connection named My Target EVM is shown in a tree structure with branches for the various Remote System functions which communicate with the target EVM using a secure SSH connection.

Sftp Files - Provides a drag and drop GUI interface to the target file system.

Shell Processes - Provides a listing of processes running on the remote system and allows processes to be remotely killed.

Ssh Shells - Provides a Linux shell window for the remote system within CCS.

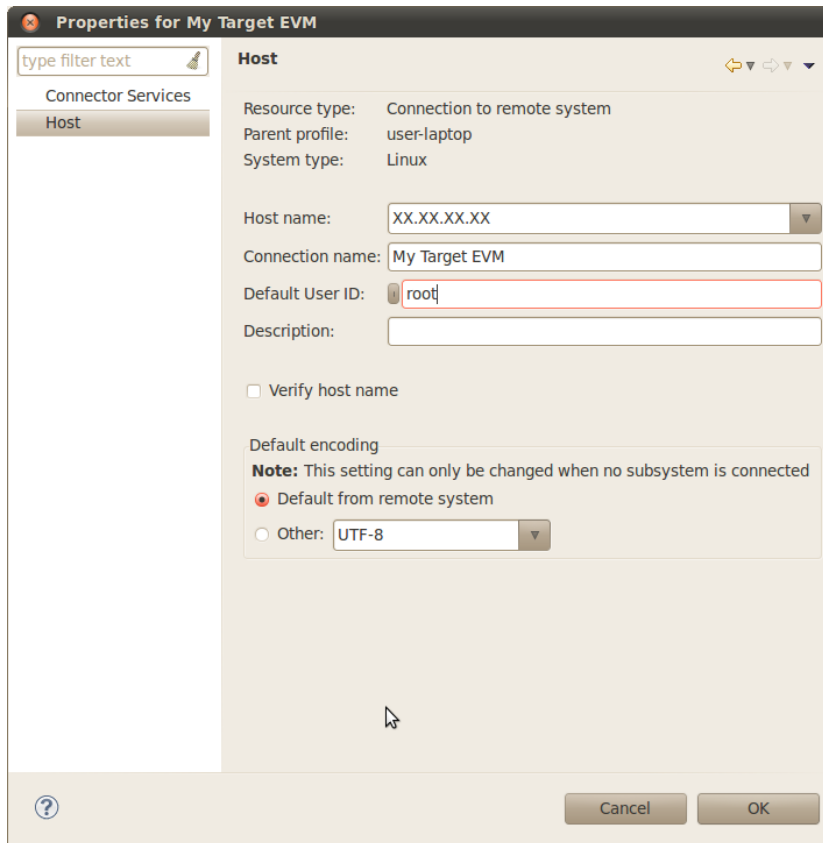
Ssh Terminals - Provides a terminal window for the remote system within CCS.



Configuring the Target EVM Connection

After the New Connection Wizard has been completed and the Remote System Explorer view has been opened, the new connection must be configured to communicate with the target EVM.

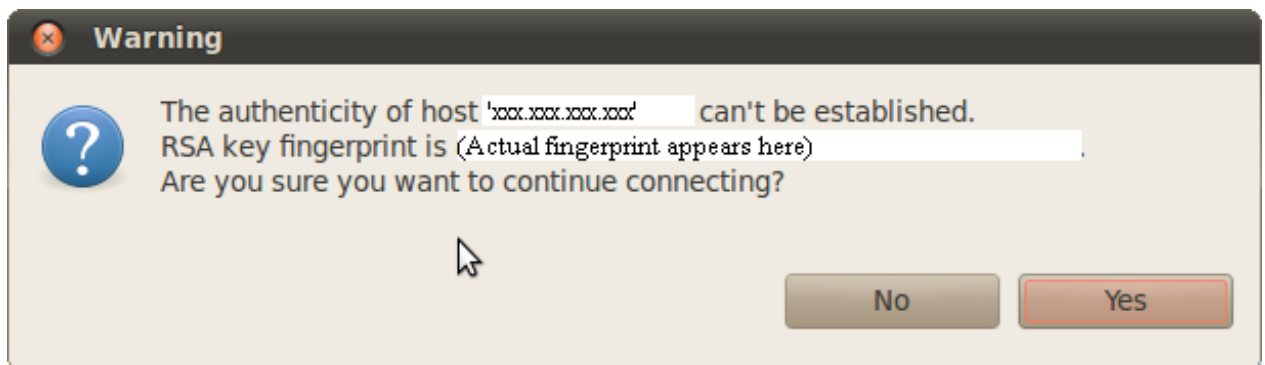
1. Right-Click on the *My Target EVM* node and select **Properties** from the context menu.
2. In the **Properties** window click on **Host**
3. Change the **Default User ID:** to **root**



4. Click **OK**

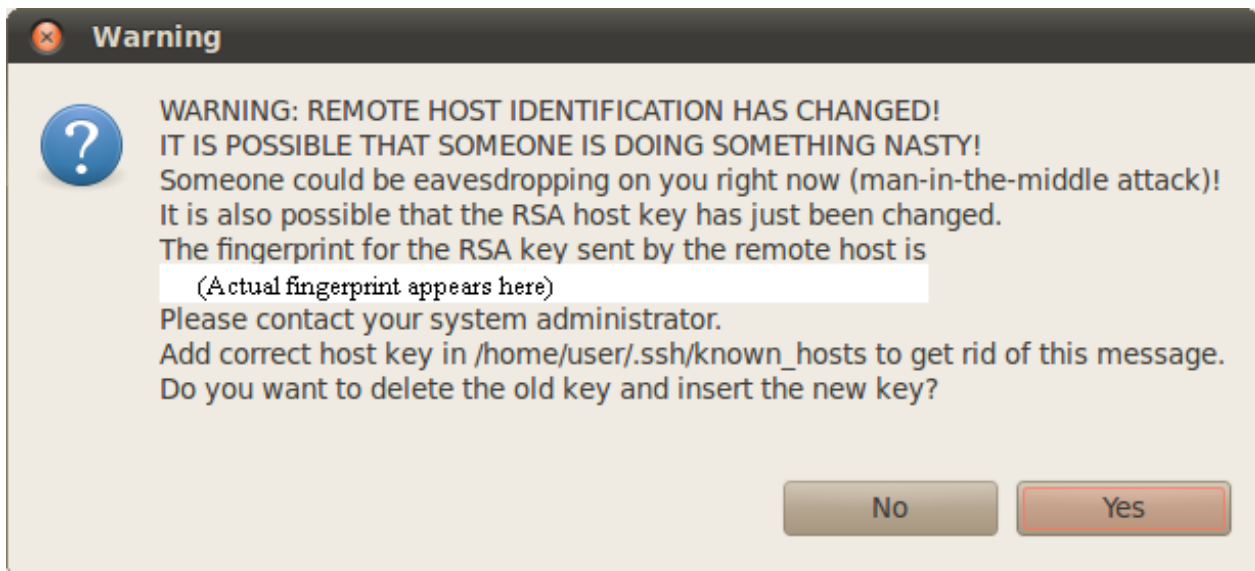
The Remote System Explorer is now ready for use. The first time the target EVM file system is booted a private key and a public key is created in the target file system. Before connecting to the target EVM the first time, the public key must be exported from the target EVM to the Linux host system. To configure the key do

1. Right-Click the **My Target EVM** node and select **Connect**
2. A dialog like the one shown below will appear



3. Click **Yes** to accept the key

Under certain circumstances a warning message can appear when the initial SSH connection is made as shown below. This could happen if the user deletes the target file system and replaces it with another target file system that has a different private RSA SSH key established (and the target board IP address remains the same). This is normal. In this case, click Yes and the public key from the target board will be exported to the Ubuntu host overwriting the existing public key.



At this point, all Remote System Explorer functions will be functional.; After this, each time CCSv5 is started, the first time a Remote System Explorer function is accessed, a login prompt will appear. Just click OK and leave the password blank.

NOTE: You can save the blank password and bypass any future prompts as well.

Configuring with a Proxy

In the case that you are behind a proxy you may need to configure CCS to use this proxy information to connect to remote servers. However, you want to make sure you also bypass the proxy for your target devices so that your connection does not attempt to go out the proxy and then come back in through the proxy. To configure the proxy you can do:

1. Click the **Window -> Preferences** menu item
2. Go to **General -> Network Connections**
3. Change the **Active Provider** from **Native** to **Manual**
4. Highlight the **HTTP** item and click the **Edit** button
5. enter your company's *host proxy URL* and *port number*
6. Do the same for the **HTTPS** item. Both items should be checked as shown below.



7. In the **Proxy Bypass** section click **Add Host...**
8. Add the **IP address** of target board (in place of xx.xx.xx.xx)
9. Click **OK**.



Using Remote System Explorer

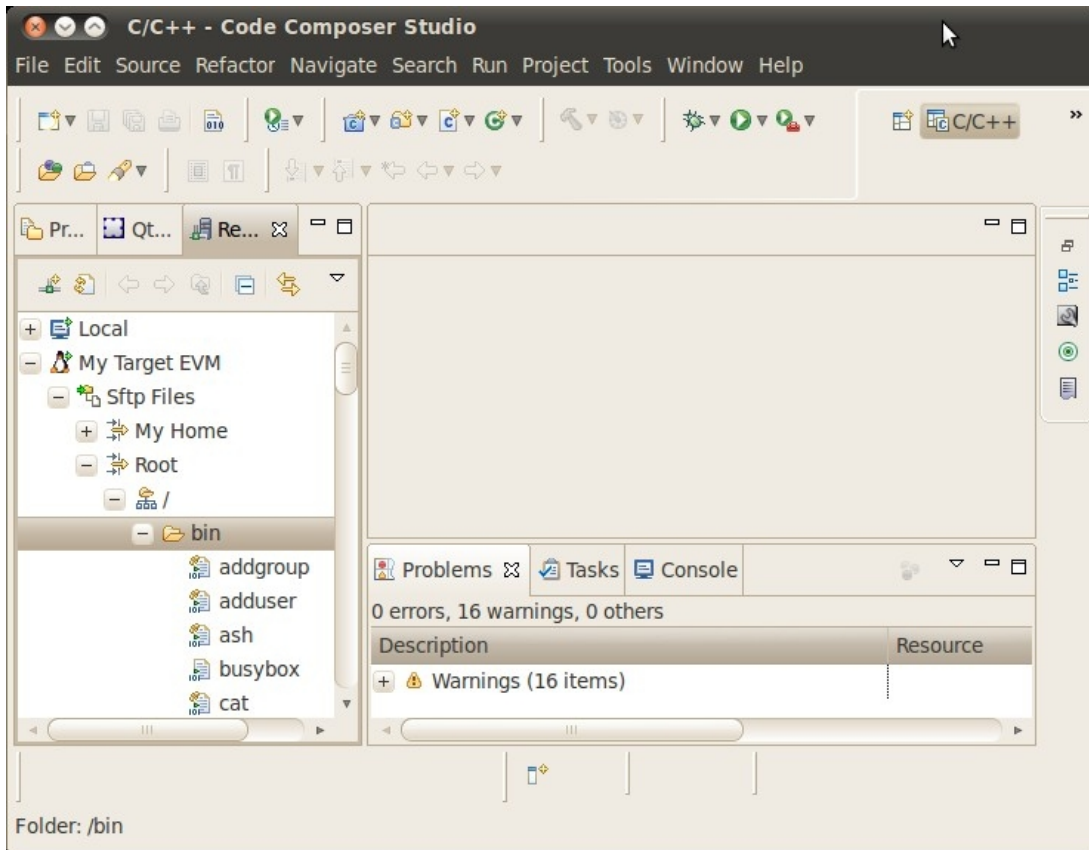
If Remote System Explorer is not included in the current CCS perspective using the steps at [Re-Opening the Remote System Explorer View](#) section above.

If prompted for a login use **root** for the *user ID* and leave the *password* blank. **NOTE:** you can save the user ID and password values to bypass this prompt in the future



Target File System Access

Expand the **Sftp Files** -> **Root** node. The remote system file tree should now show the root directory. You can navigate anywhere in the remote file system down to the file level. Files can be dragged and dropped into the remote file tree. A context menu allows you to create, rename or delete files and folders.

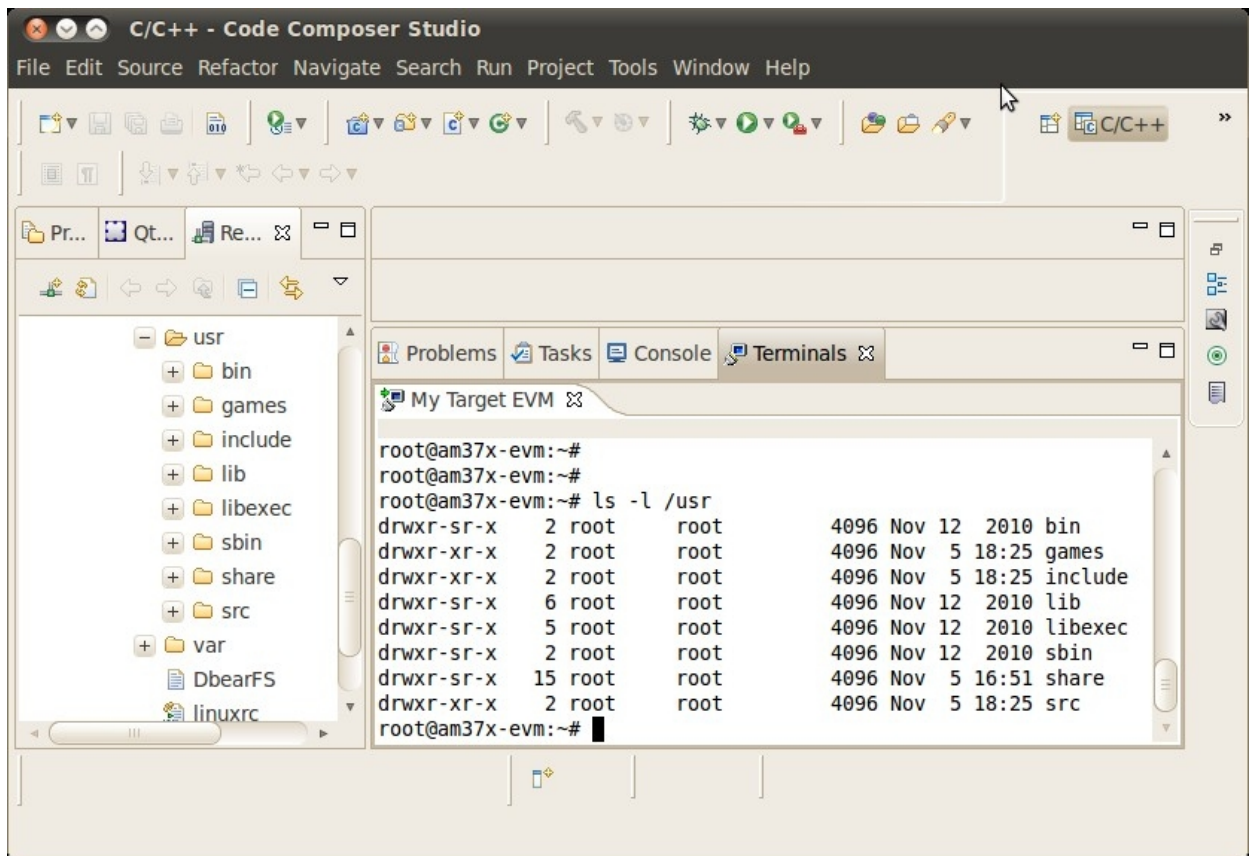


The local file system on the Linux host can also be accessed by expanding the **Local** -> **Local Files** node.

SSH Terminals

To open an SSH Terminal view

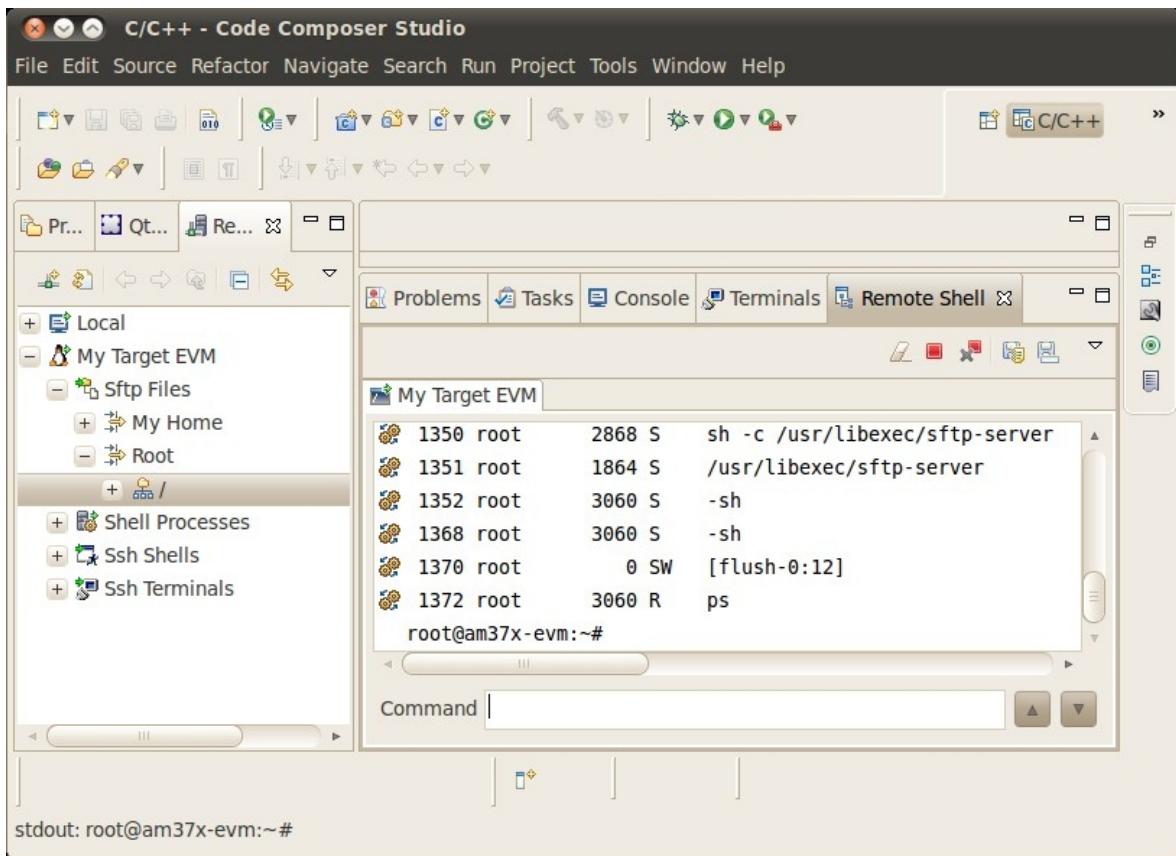
1. Right-Click the **Ssh Terminals** node under the target EVM connection
2. Select **Launch Terminal** from the context menu
3. Type shell commands at the prompt in the terminal window. Below is a sample command to list the contents of the remote /usr folder.



SSH Shells

To open an SSH Shell view

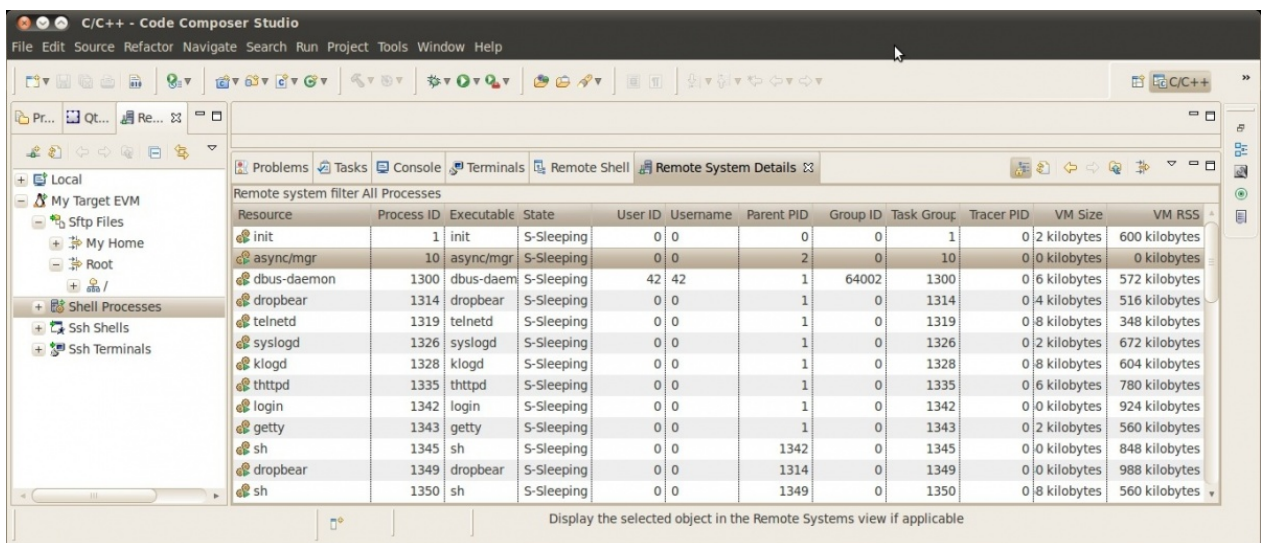
1. Right-Click the **Ssh Shells** node under the target EVM connection
2. Select **Launch Shell** from the context menu.
3. Type a shell command into the Command edit box and press the Enter key. Below is the output from the ps command which displays the processes running on the remote system



Shell Process Monitor

To open the Shell Processes view

1. Right-Click the **Shell Processes** node under the target EVM connection
2. Select **Show in Table** from the context menu
3. Double-click **All Processes** to see the list of processes running on the target
 - By clicking any column header, the list can be sorted by the parameter in the selected column, such as executable name, memory size, or PID
 - You can Right-Click on a particular process and click **Kill** in the context menu to kill a process.



Archived Versions

- [Sitara SDK 05.03 \(Archived\)](#) ^[1]

References

[1] http://processors.wiki.ti.com/index.php?title=How_to_setup_Remote_System_Explorer_plug-in&oldid=83884

How to Run GDB on CCSv5

Return to the [Sitara Linux Software Developer's Guide](#)

Preliminary Requirements

1. The Sitara SDK for your hardware platform has been installed on your Ubuntu host machine.
2. A serial connection between your host machine and the target board with a serial terminal emulator running on the host. Minicom is recommended. You could also use a telnet window to the target system.
3. An Ethernet connection between your host machine and the target board.
4. This article assumes that the Remote System Explorer plug-in has been setup to provide an SSH connection to the remote file system. See [How_to_setup_Remote_System_Explorer_plug-in](#) ^[1]
5. Boot up the target EVM.
6. Start CCSv5 and open the project you wish to debug.
7. The executable to be debugged must reside in the target file system. It must have been built from the debug build configuration so that it contains the symbol information.

NOTE: For this example, we are going to debug the dhrystone project

Determining the Target Board Ethernet IP Address

1. Get the target Ethernet IP address:

```
root@am335x-evm [2]:~# ifconfig
```

You should see a response that is similar to this. The actual target Ethernet IP address will appear where the xxx.xxx.xxx.xxx is shown below. (This IP address will be used in a later step):

```
eth0      Link encap:Ethernet  HWaddr 00:50:C2:7E:8F:D4
          inet addr:xxx.xxx.xxx.xxx  Bcast:0.0.0.0  Mask:255.255.254.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:24  errors:0  dropped:0  overruns:0  frame:0
          TX packets:3  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3174 (3.0 KiB)  TX bytes:1770 (1.7 KiB)
          Interrupt:80

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0  errors:0  dropped:0  overruns:0  frame:0
          TX packets:0  errors:0  dropped:0  overruns:0  carrier:0
```

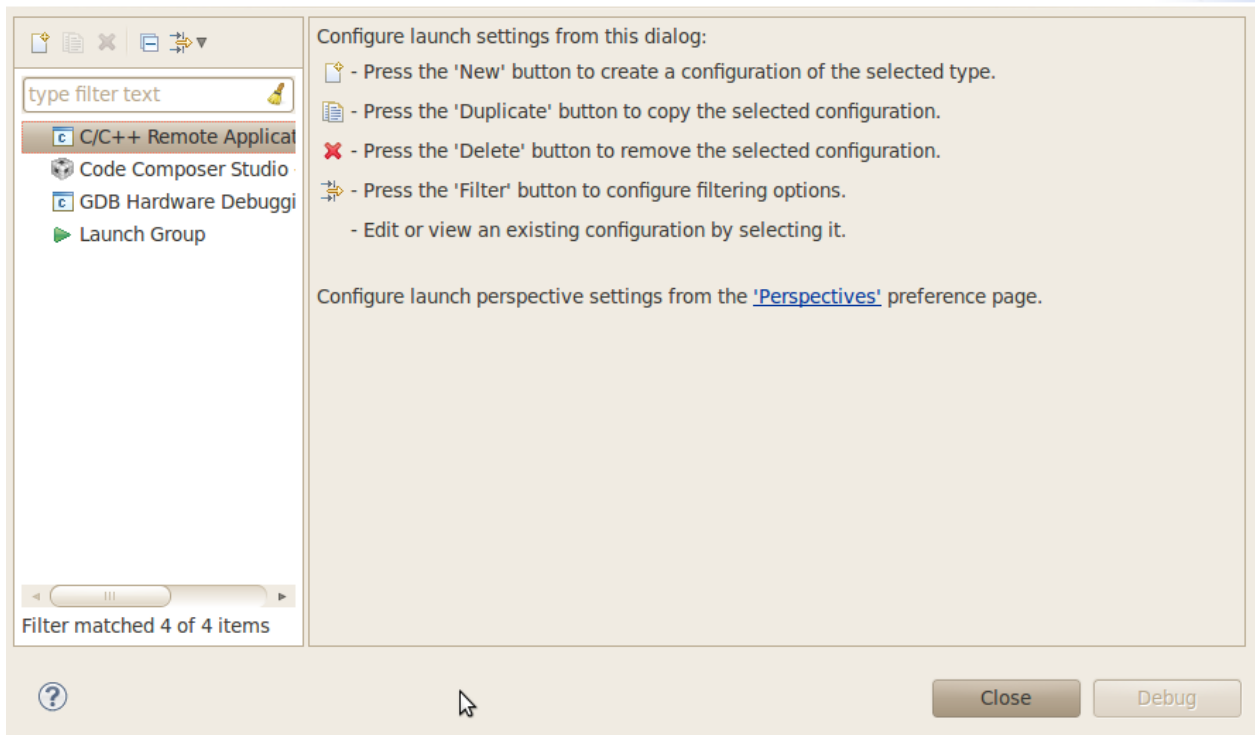


```
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

Creating the Debug Configuration for the Project

1. In CCS, select the project you wish to work with by clicking on it and highlighting it.
2. Select the Run -> Debug Configurations menu item. This opens a dialog box as shown below.

Create, manage, and run configurations

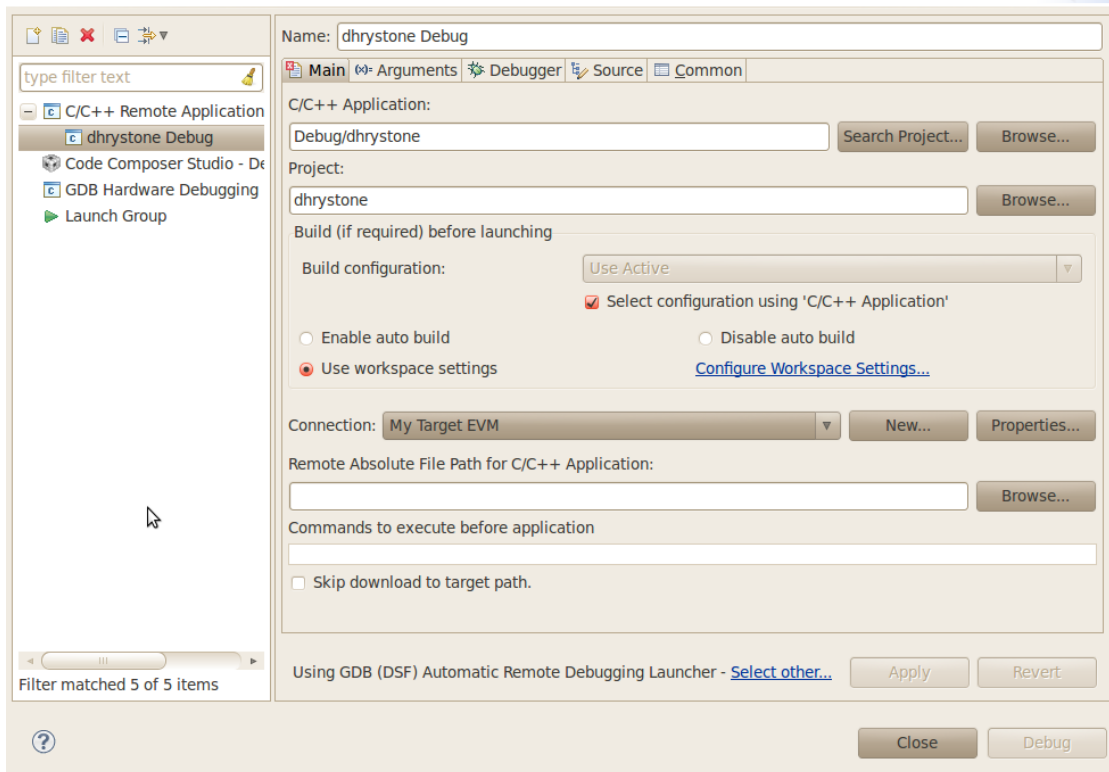


3. Double click C/C++ Remote Application. You should then see a new debug configuration named "dhrystone Debug" as shown below.

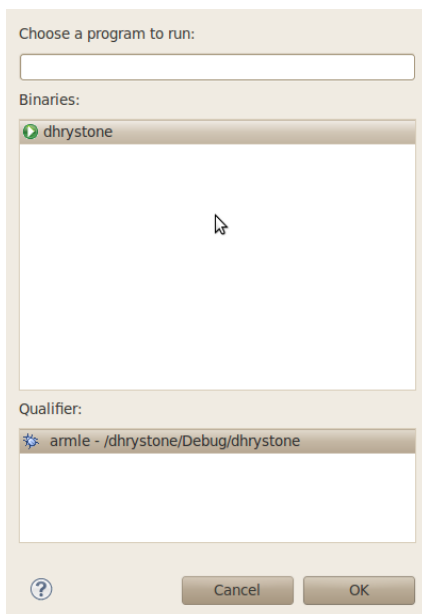
Select your target connection from the Connection drop-down box. In the example the target connection is called My Target EVM.

Create, manage, and run configurations

Remote executable path is not specified.

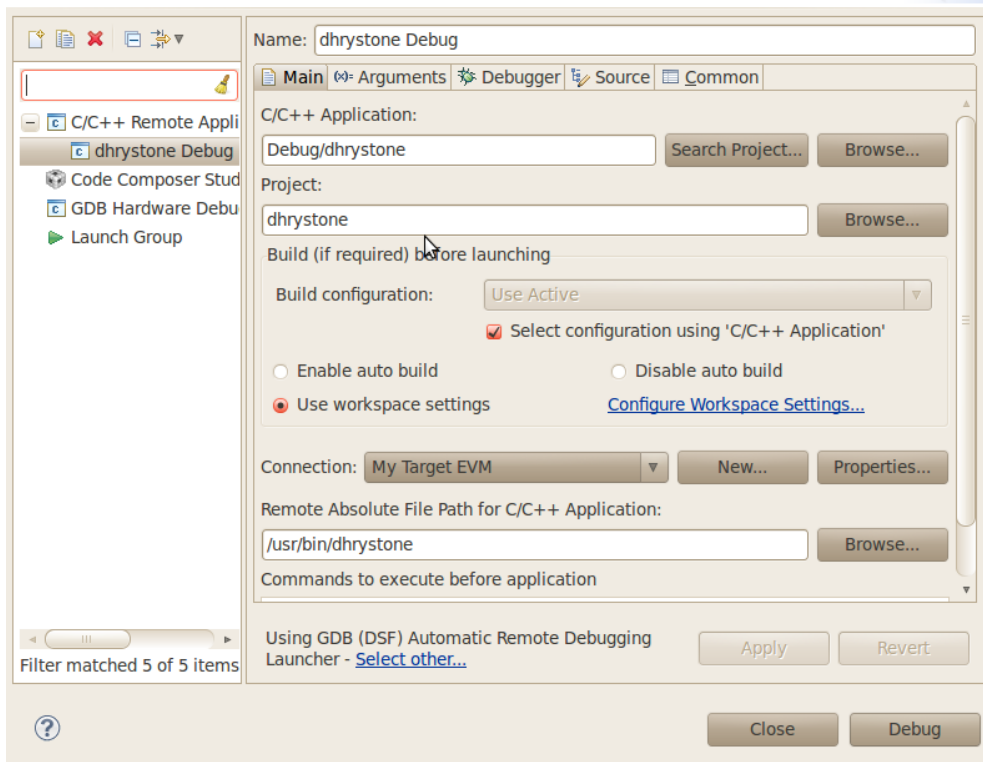


4. Click the Search Project button to open the Program Selection dialog box below. Click on the "armle - /dhrystone/Debug/dhrystone" item and click OK.

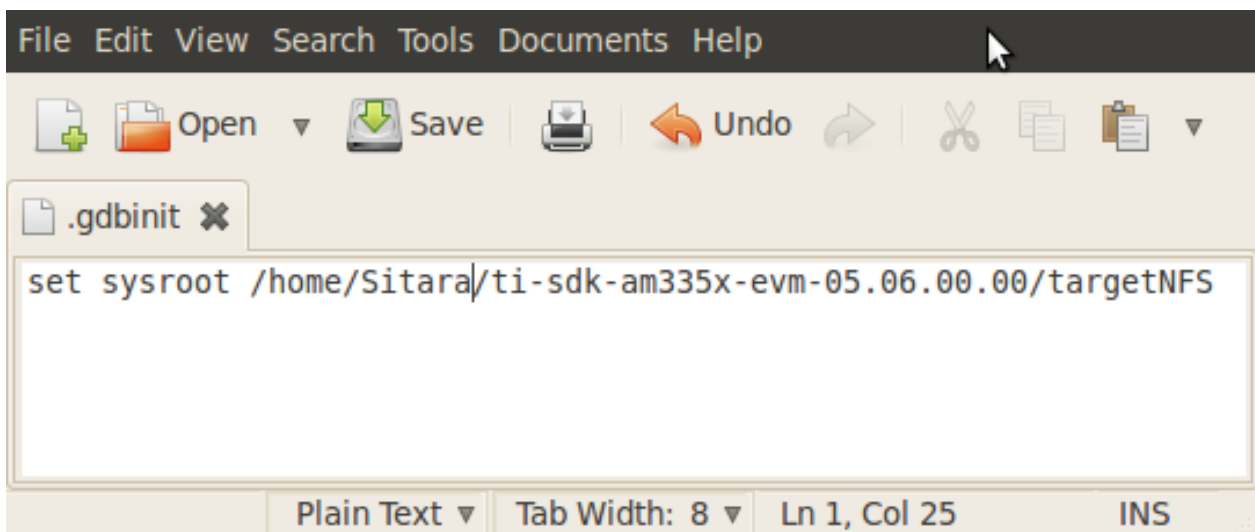


5. Click the "Browse..." button for "Remote Absolute File Path for C/C++ Application". Navigate to the executable file on the remote file system. For this example, the executable file is found at "/usr/bin/dhrystone".

Create, manage, and run configurations



The .gdbinit file is used by GDB to locate source files and library files on the target. The .gdbinit file is created when the SDK environment script runs. Here is an example of a .gdbinit file.



6. Click the Debugger tab. On the Debugger page, the Main tab should be selected. Click Browse next to "GDB debugger" and browse to the GDB executable.

Click browse next to "GDB command file" and browse to the .gdbinit file in the SDK install directory. When you try to browse to the .gdbinit file, you will need

to R-Click -> Show Hidden Files to see the file. Click the Close button and you are now ready to debug the application!

Running the Debug Session

1. Make sure that you are setup for the debug build configuration which contains symbol information. In the C/C++ perspective, click on the dhrystone project to select it and

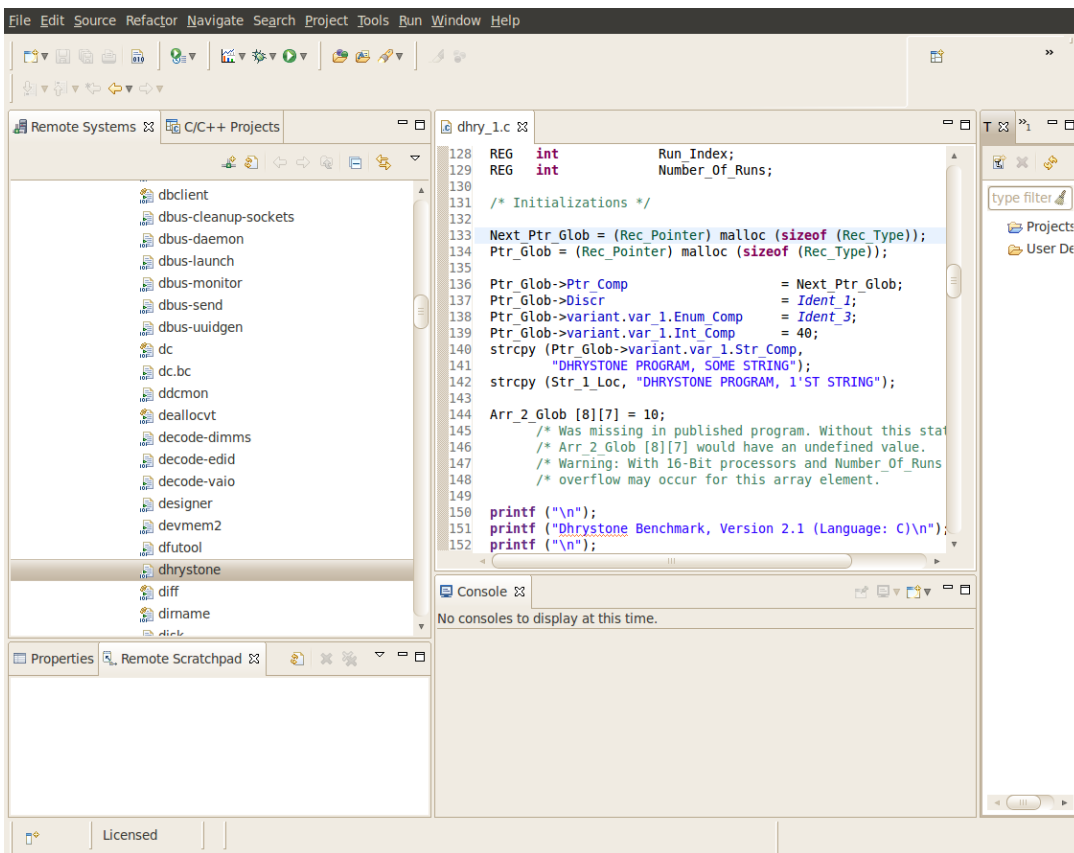
Project -> Build Configurations -> Set Active -> Debug.

2. The executable file and the folder containing the executable file need to be set to Read-Write-Execute permissions for the GDB Automatic Remote Debugging Launcher to work.

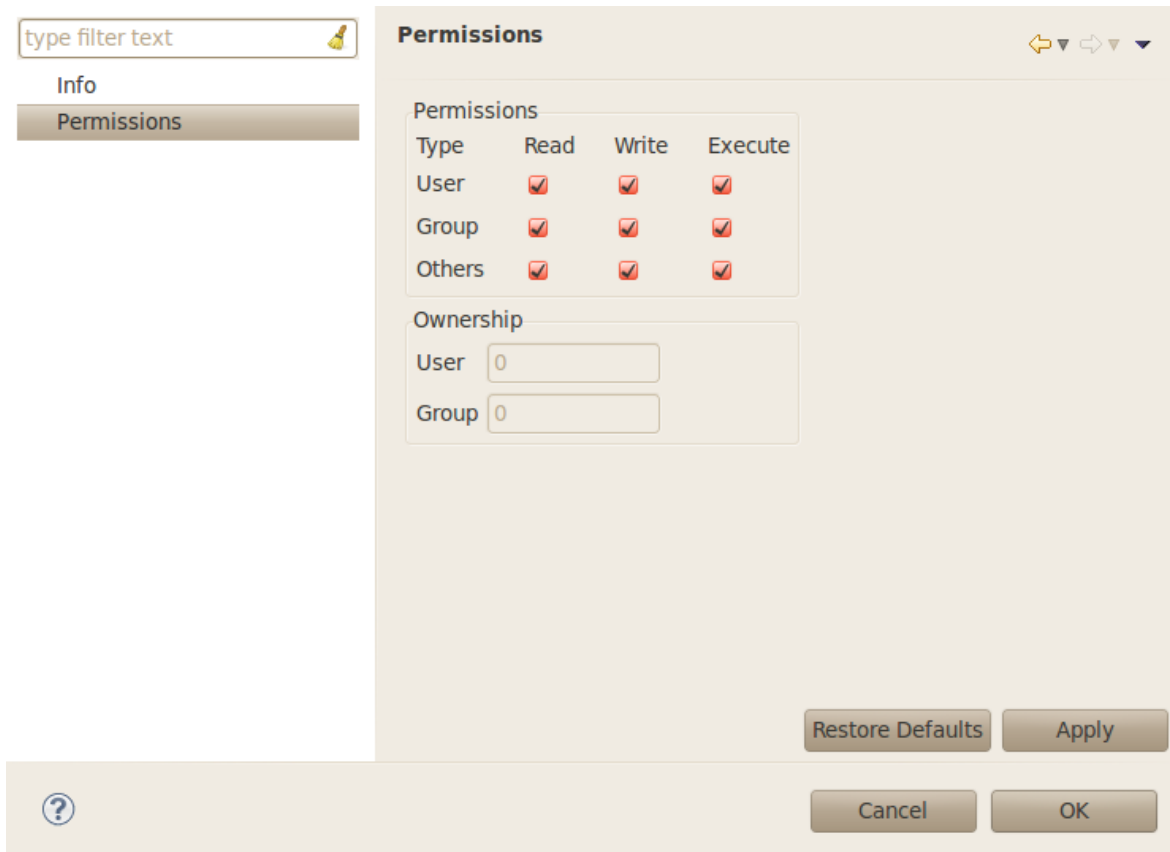
This only needs to be done the first time. Click the ">>" in the upper right corner of the CCS window and select the Remote System Explorer perspective. If you are prompted

to enter a password, the username should be "root" and the password should be left blank. Open the Sftp Files node in the Remote Systems tree and navigate under the root

node to the executable file.



3. R-Click on the dhrystone file and select Properties. Click Permissions and change it so that all permissions are set. Also set all permissions on the /usr/bin folder.

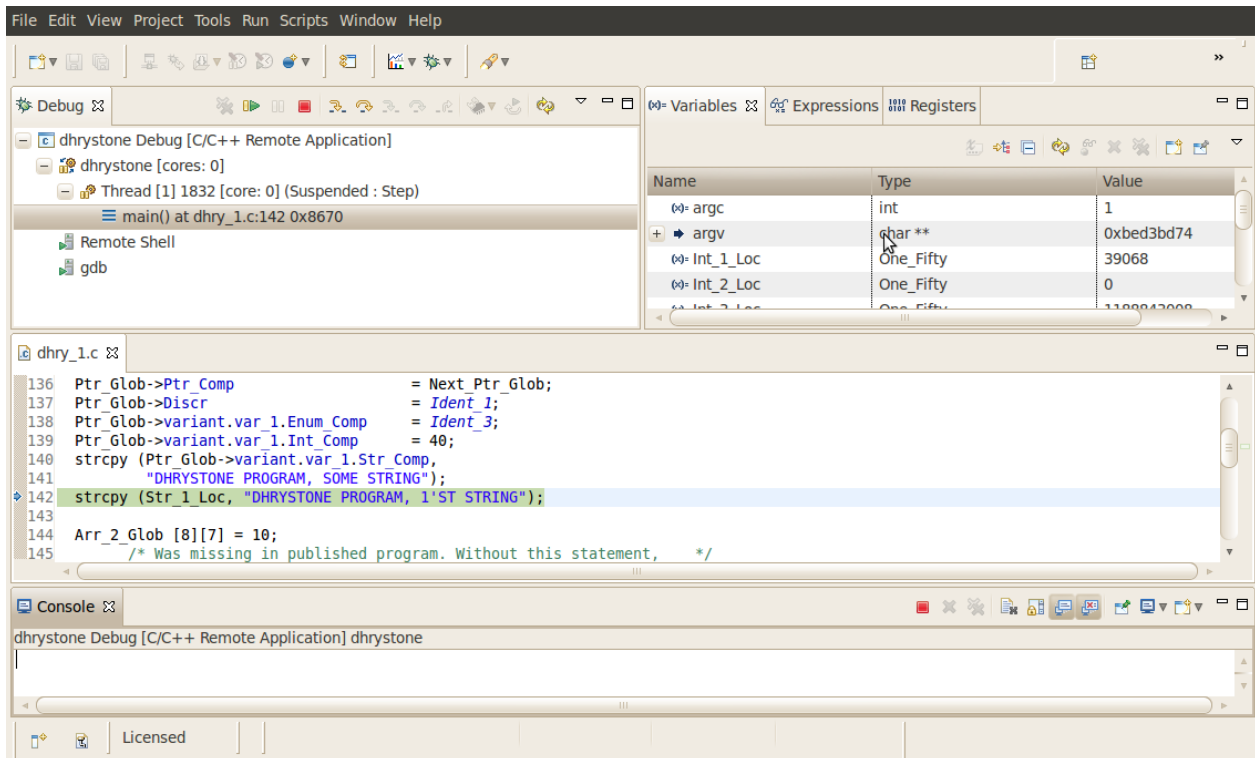


4. Click the green "bug" icon to build the executable, transfer the executable to the target, start gdbserver and start debugging.

CCS will change to the CCS Debug perspective. The debug tab will show the running threads and their status. The source code window will show

the program halted at the first executable source code line in the main() function. The Variables window will show the local variables and their current

values.



5. To toggle a breakpoint, highlight the line of code in the source code window. Then click the Run -> Toggle Breakpoint menu item.

6. Use the debugger "Step Over" and "Step Into" icons to step through the source code.

7. To resume program execution, click the Run -> Resume menu item.

NOTE: Do not click the Run -> Debug menu item, as that will attempt to start a new debug session.

From here, you can make changes to the C source files, save the changes and then just click the green "Bug" icon again and you will be debugging the new executable on the target.

(Each time you start the debugger the executable is built, automatically transferred to the target board and the gdbserver program is started for you.)

Stopping the Debug Session

When finished debugging the dhrystone application, click the Run -> Resume menu item. To terminate the program, click the Terminate icon in CCS (this icon is a red square).

NOTE: If the program being debugged ends abnormally or crashes, the target console running gdbserver may no longer respond. In that case, the target board must be rebooted to start another debug session.

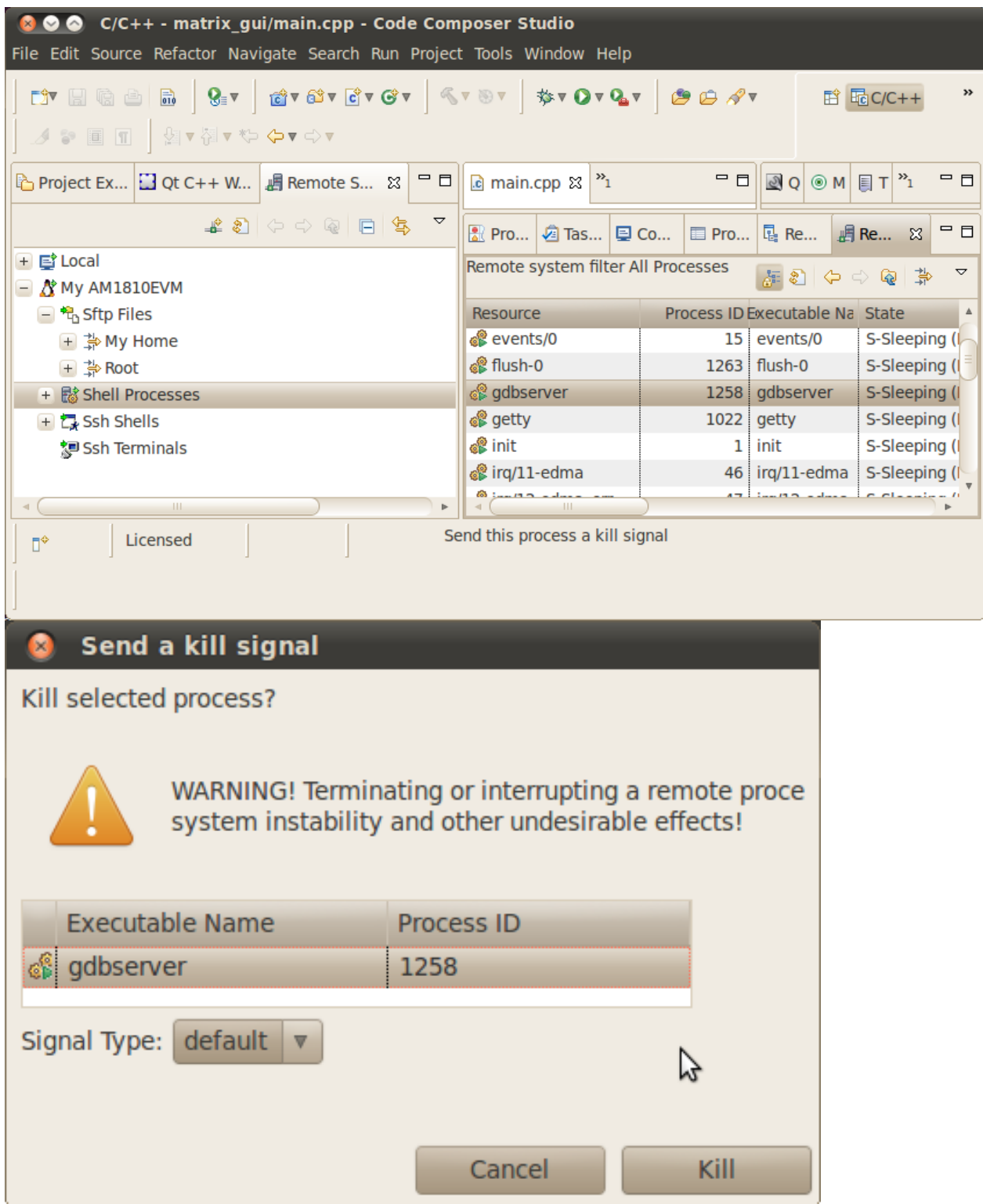
To avoid having to reboot the target system, the Remote System Explorer Shell Processes feature can be used to terminate the gdbserver process.

See [How_to_setup_Remote_System_Explorer_plug-in](#)^[1] to setup the Remote System Explorer plug-in.

Once setup, you can follow these steps to terminate gdbserver:

- 1) Change to the Remote System Explorer perspective. R-Click on Shell Processes in the target connection tree and select Show in Table to open a Remote System Details window.
- 2) Double-click on "All Processes" in the table to display the list of processes running on the target system.
- 3) Click on "Executable Name" in the table headers to sort the list by executable name.

4) Find the gdbserver process. R-Click on it and select Kill. This will open a "Send a Kill Signal" dialog box. Click the Kill button.



Archived Versions

- [Sitara SDK 05.03 \(archived\)](#) ^[3]
- [Sitara SDK 05.02 \(archived\)](#) ^[4]

References

- [1] http://processors.wiki.ti.com/index.php/How_to_setup_Remote_System_Explorer_plug-in
[2] <mailto:root@am335x-evm>
[3] http://processors.wiki.ti.com/index.php?title=How_to_Run_GDB_on_CCSv5&oldid=87960
[4] http://processors.wiki.ti.com/index.php?title=How_to_Run_GDB_on_CCSv5&oldid=77290

Pin Mux Utility for ARM MPU Processors



This User's Guide supports Pin Mux Utility v2.4.1.0 (APR 2012) or later which is available in the Sitara Linux SDK version 05.06

For any previous version of Pin Mux Utility see [Pin Mux Utility v1.x User's Guide \(archived\)](#) ^[1]

Introduction

The Pin Mux Utility for TI ARM Cortex-A8 based microprocessors is a Windows-based software tool for configuring pin

multiplexing settings and I/O cell characteristics for the AM335x, AM35x, AM37x / DM37x, AM387x / DM814x, AM389x / C6A816x / DM816x and OMAP35x devices. The program has been tested on Windows XP and Windows 7.

Sitara processors provide pad configuration programmability to control the routing of internal signals to the external balls of the device. Pad configuration also allows I/O cell characteristics to be controlled. These include enabling of internal pull-up / pull-down resistors and specifying I/O cell behavior independently in active and standby modes of operation.

The Pin Mux Utility provides a graphical user interface for selecting the peripheral interfaces that will be used in the system design and for resolving pin multiplexing conflicts. The current state of all settings can be saved as a design state file which can be reloaded to restore the state of all tool settings.

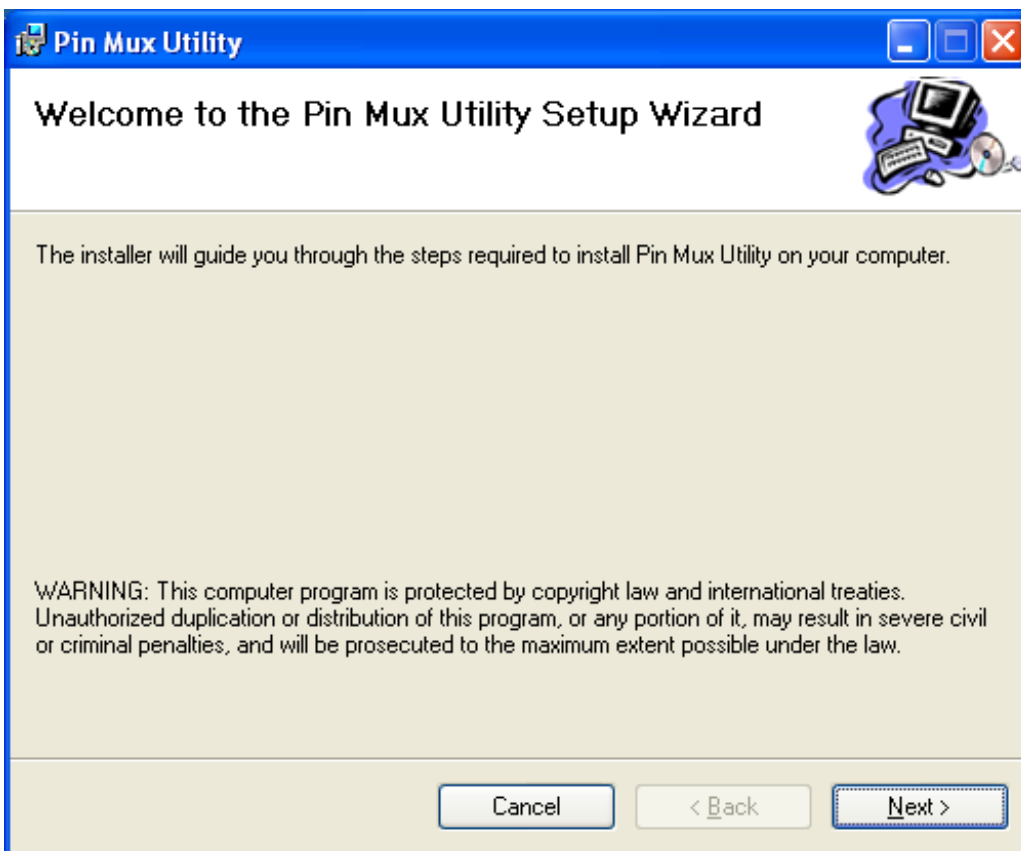
The results can also be output as C header files. The first header file is device-specific and it specifies all the pad configuration registers that can be programmed on the device. The second header file is board-specific and it specifies all the pin multiplex and pad configuration settings that were generated for your specific system design. The output sample header files are in the format used for the AM35x / AM37x / OMAP35x U-Boot source code. For other supported devices, the source code provides a complete description of all settings but is not directly usable as U-Boot source code.

Software Installation

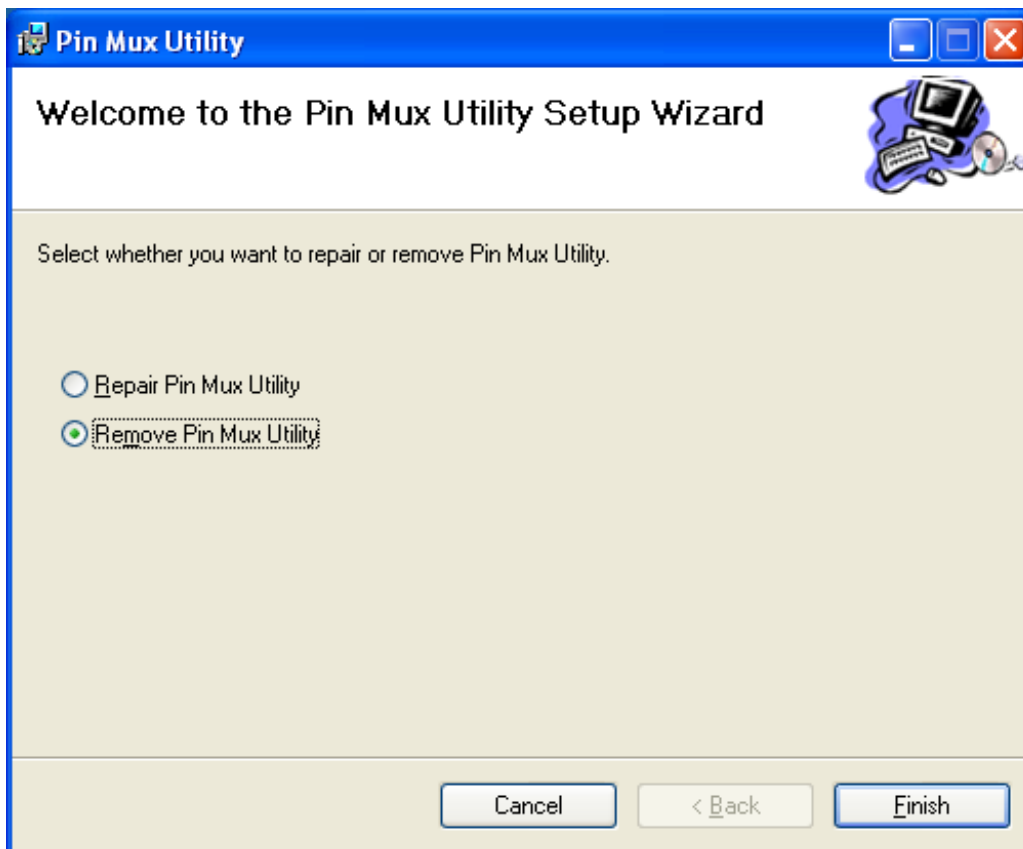
Pin Mux Utility v2_04_01_00 is available for download from the ti.com website [here](#)^[2]. It is also installed automatically as part of the [Linux EZ Software Development Kit for Sitara ARM Microprocessors](#)^[3] but the version from the ti.com website is recommended to get the most recent updates.

If you are running Linux as a virtual machine on a Windows O/S, the Windows Installer and Pin Mux Utility program can be run on the Linux host PC under the Windows O/S. The generated source files can then be transferred to the Linux virtual machine via shared folders or another available method.

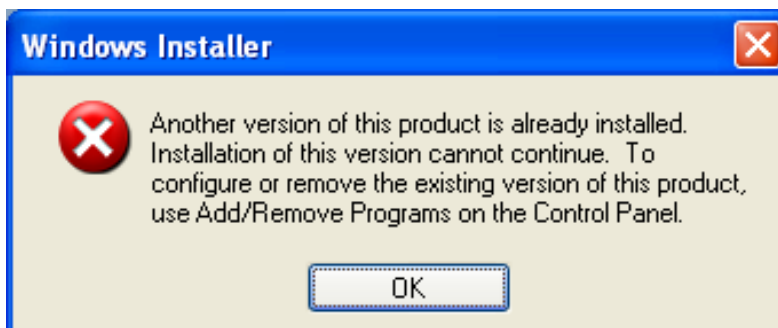
If your Linux O/S is not a virtual machine, the Windows installer and Pin Mux Utility program must be run on a separate PC running the Windows O/S. To run the Windows Installer, run Setup.exe. The installer should start running as shown below.



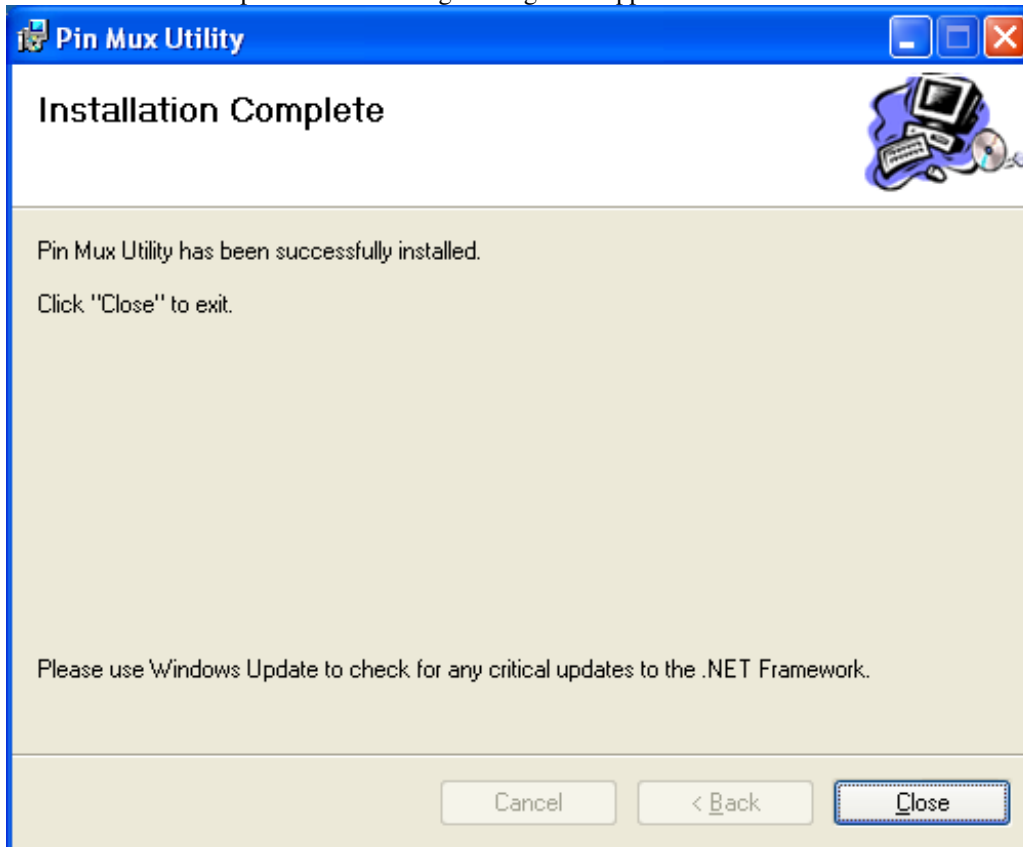
If a previous version of Pin Mux Utility was installed, the following message MAY appear. Select "Remove Pin Mux Utility" and click Finish to uninstall the previous version. Then, run Setup.exe again to install the new version.



In some cases, if Pin Mux Utility was already installed the following message will appear instead. In this case, use Add/Remove programs from the Windows Control Panel to uninstall the original version, then run the installer again.



After installation completes the following message will appear.



Software User's Guide

Program Startup

To see the current release notes, click on this in the Windows Start menu:

All Programs > Pin Mux Utility > Release_Notes.txt

To start the program, double-click the Pin Mux Utility icon on the desktop, or click

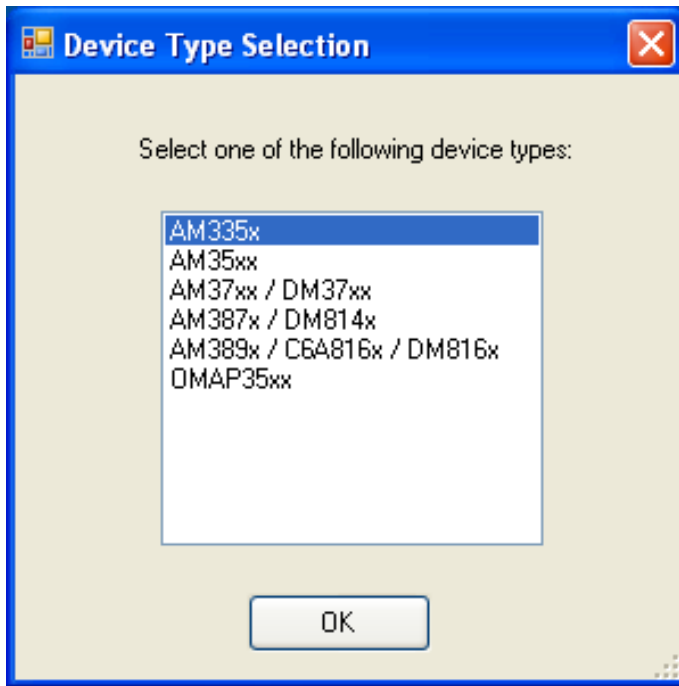
All Programs > Pin Mux Utility > Pin Mux Utility

in the Windows Start menu.



Pin Mux Utility

A dialog box will appear indicating which devices are supported by this installation of the Pin Mux Utility as shown below. Select your device and click OK.



Main Window

- The main window opens with device data populated for the selected device type. All pin mux selections and pad configurations are in the power-up-reset state.
- The main window (and the Peripheral Interface View windows) may be maximized or resized so that the complete signal names can be seen.
- Double-click at the divider between columns in the table row header area to set width of a column to match the widest column text.
- The title bar indicates the path where output source files will be stored.
- The Peripheral Interfaces data grid shows the current peripheral interface status for the member signals of each peripheral interface.
- The Device Package selector controls which package type is used to populate the "Bot/Top Ball" ball location column.

When a signal has been selected and the IO Pad is not connected to any device ball this ball has a RED background in the "Bot/Top Ball"

column and the "Pkg Conflict!" indicator is shown below the Device Package drop-down box. IO Pads that are not connected to any device ball

should have all signals Not Selected and be defaulted to the "Selected (Ball Available)" state (unless the ball has a fixed mux mode).

- The Legend describes the color scheme that is used to give a visual indication of the state of each peripheral interface and the state of each signal in the Pin Mux Grid.

- The main Pin Mux Grid represents one device ball in each row and contains columns to display the current pad configuration and ball location(s).

Columns labeled Mode 0 to Mode N represent the pin mux state at each ball. The state may be changed by double clicking on a cell.

Multiple cells may be selected in either the Pad Configs column or in the Mux Mode columns. R-Click and use

context menu to edit these settings.

- The status bar indicates the total number of device balls, number of balls remaining and the number of balls with conflicting mux settings.
- The Reset button is used to set all mux selections and pad configurations to the power-on-reset state.
- The close button exits the program

NOTE: For devices other than AM335x the IO Set Violation, IO Set Subset and IO Set Match statuses are not shown. The Peripheral Interface uses a GREEN=Selected status in place of those, meaning that at least one interface member signal is selected and there are no conflicts.

For devices other AM335x, the IO Power column is not used.

Menu Items

File Menu

- File > Save > Design

Saves current pin mux selections and pad configurations to a data file that can be reloaded at any time or on a subsequent run of the program. This file, by default, will be stored at:

<My Documents Folder>\Pin Mux Utility\Design1\<device_name>\PinMuxDesignState_<device_name>.dat

- File > Open > Design

Opens data file stored previously with the File > Save > Design menu item. Must open a file for the same device type that was selected at program startup. This will populate all mux selections and pad configuration parameters from the data file.

- File > Save > Source > Linux

Saves the current pin mux selections and pad configurations to C header files.

- File > Exit

Exits the program. You will be prompted to "Save changes to mux selections and/or pad configurations?". If Yes is clicked, this will open a file save dialog to save the design data file as when the File > Save > Design menu command is used.

Help Menu

- Help > ReadMe

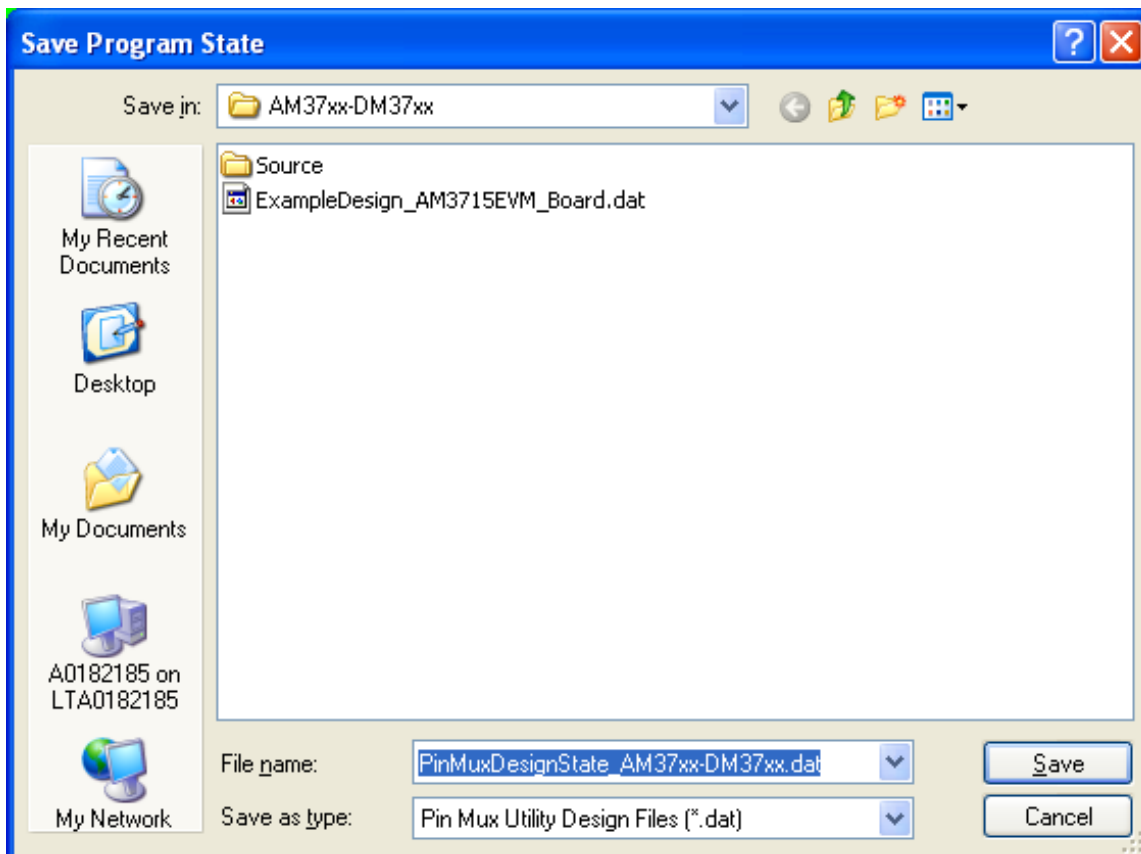
Displays a window describing features of the program and a link to this User Guide Wiki article.

- Help > About

Displays the program version.

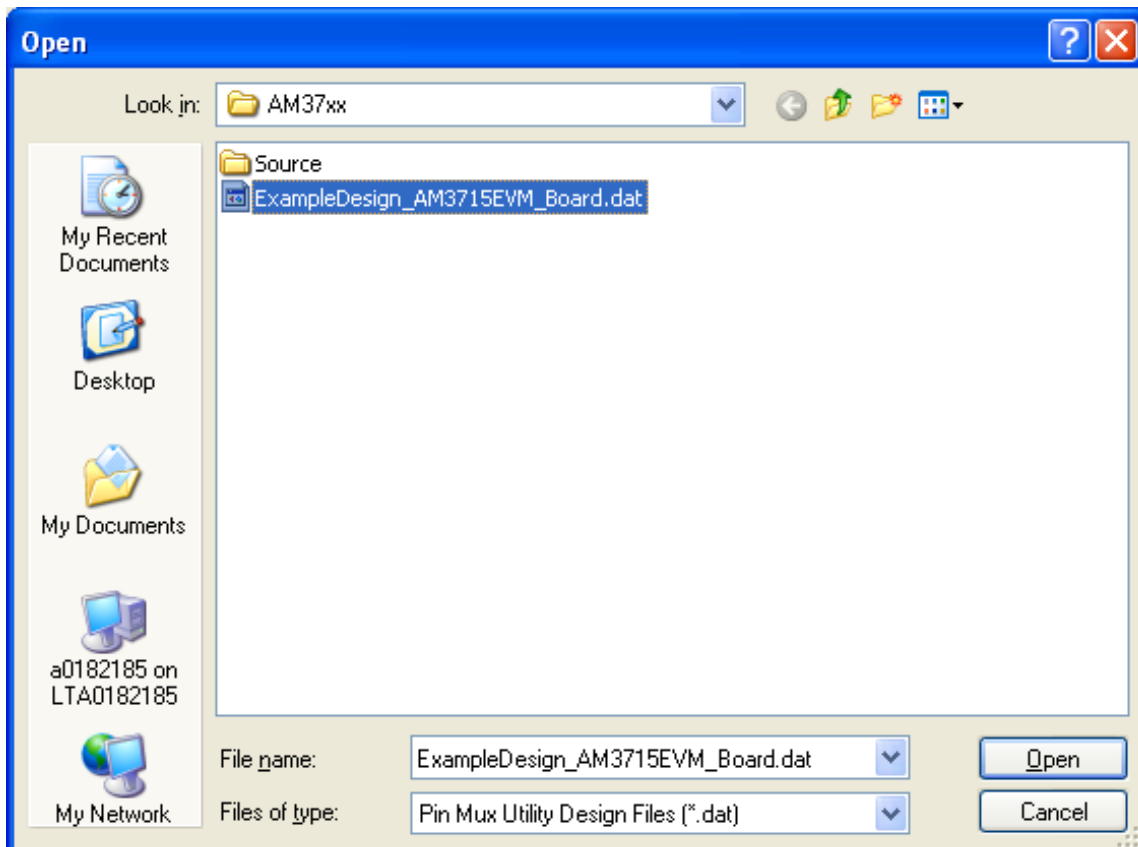
Saving Your Work to a Design State File

Use File > Save > Design, to save your work to a design state file. This file can later be reloaded to continue working on your design.



Opening an Existing Design State File

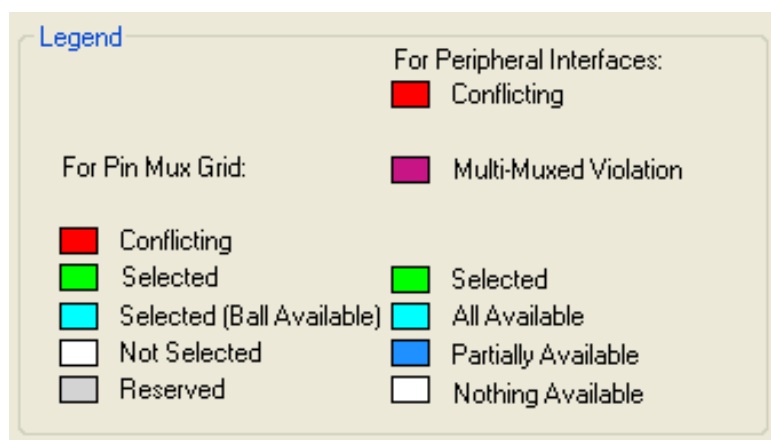
Use File > Open > Design , to open saved design state file. For some devices, an example design file is provided with settings that match the TI EVM board.



Legend

The legend below shows the visual indicators that are used to display the status of the pin mux settings at the peripheral interface and individual ball levels. The tables below describe how these states are to be interpreted.

Legend: As displayed for all device types except AM335x



Legend: As displayed for AM335x device



Description - Peripheral Interface Status Used with All Device Types

Peripheral Interface Status	Description
RED = Conflicting	At least one interface member signal is involved in a conflict with another signal on the same ball. These conflicts must be resolved before obtaining a final pin mux configuration for the device.
VIOLET = Multi-Muxed Violation	This involves cases where the same internal signal is muxed to multiple ball locations and the signal has been selected at more than one of those ball locations. The interface view will contain a status message (at upper left) that indicates the first two multi-muxed signals that are found to be selected. These conflicts must be resolved before obtaining a final pin mux configuration for the device.
GREEN = Selected	At least one interface member signal is selected and there are no conflicts or violations.
AQUA = All Available	All member signals of the interface are available for selection.
BLUE=Partially Available	At least one but not all member signals of the interface are available for selection. One or more member signal(s) are not available because another interface is already using the ball(s).
WHITE=Nothing Available	None of the member signals of the interface are available for selection. All member signals are unavailable because other interface(s) are already using the ball(s).

Description - Additional Peripheral Interface Status Used with AM335x Device

Peripheral Interface Status	Description
BROWN = IO Power Violation	The peripheral interface has member signals selected and the IO Cells are not all connected to the same voltage.
ORANGE = IO Set Violation	The selected members of the interface do not satisfy the requirements of any of the pre-defined IO Sets for the interface. The selected signals are a superset of all the pre-defined IO Sets. These violations must be resolved before obtaining a final pin mux configuration for the device.
YELLOW = IO Set Subset	The selected members of the interface are a subset of the specified signals for one or more IO Sets. This is an informational status and is acceptable for a final pin mux configuration for a device.
GREEN = IO Set Match	The selected members of the interface match one of the pre-defined IO Sets exactly. Or, if no IO Sets are defined for the interface, it means at least one interface member signal is selected without conflicts or violations.

Below is the description for the status that occurs on the cells of an individual device ball.

A device ball corresponds to one row in the Pin Mux Grid in the main window or in an Interface View.

Legend for Pin Mux Grid or Interface View

Pin Mux Grid Status	Description
RED = Conflicting	The Conflicting status means the user has tried to select two or more signals at the same ball. Conflicting signals are all shown in RED. Conflicting signals must be deselected until not more than one signal is selected at each ball to generate a valid pin mux configuration for the device.
GREEN = Selected	The Selected status means this signal is selected for this ball without any conflict. (There still may be violations reported at the peripheral interface level.)
AQUA = Selected (Ball Available)	The Selected (Ball Available) status means the mux mode is selected by default because no other signal in the row is selected. In addition, the Selected (Ball Available) status means the ball is available for use - you may select any other signal in the row without causing a conflict.
WHITE = Not Selected	The Not Selected status means this signal is not selected for this ball.
GREY = Reserved	The Reserved status means this mux mode is not a valid selection. These cells cannot be changed.

Main Data Grid

The main data grid provides a view of all multiplexed signals that can be brought out to the pads of the die.

The "Bot / Top Ball" column indicates the ball location(s) that are connected to the die pads. If this column indicates "- / -", the die pad does not connect to a ball of the device (for the selected device package).

Each row represents one pad on the die. Each column labeled Mode 0 to Mode 7 represents one mux mode.

Each row is independent. To scroll to the first signal of a peripheral interface, double click the corresponding cell in the peripheral interfaces grid.

Changing State of an Individual Cell

To change the state of an individual cell, double-click the cell. This applies to the Mode0 - Mode7 columns in the Pin Mux Grid. Changes occur according to the following table.

Cell State Change when Double-Clicking a Cell

Original Cell State	New Cell State
Conflicting	Not Selected
Selected	Not Selected
Selected (Ball Available)	Selected (Ball Available) - Nothing happens when clicking cell with status Selected (Ball Available). Select any other signal in the row to remove the Selected (Ball Available) status.
Not Selected	Selected if no other cells in row are already selected or conflicting Conflicting if any other signal in row is already selected or conflicting
Reserved	Reserved - Cannot be changed

Changing State of a Group of Cells

You may select a group of cells in the Mode 0 - Mode N columns. To select a group of cells, click on a cell in this area and drag the mouse down. R-Click on any selected cell and select Change Selected Mux Mode Cells to change the state of a group of cells at once. You can also click and drag to select a group of cells in the Pad Config column. R-Click any selected cell in the Pad Configs column and select Edit Selected Pad Configs from the context menu to edit the pad configuration for a group of cells.

Interface Views

The interface view displays only the balls that make up a peripheral interface. To open an Interface View, R-Click on a cell in the Peripheral Interfaces data grid in the main window and select View Pins from the context menu. Any number of Interface View windows may be opened at the same time. The signals that are members of the peripheral interface are shown in bold text.

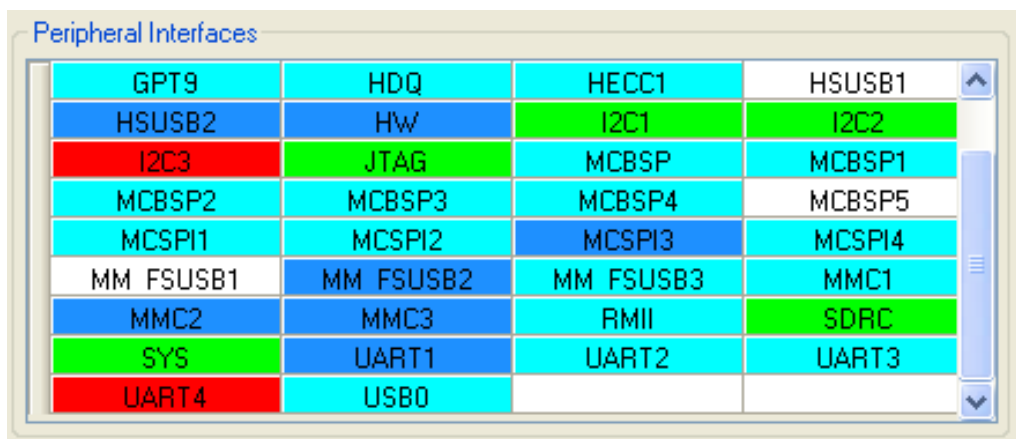
Interface View with Pin Mux Conflicts (all device types)

Here is an example of a Pin Mux conflict (AM35xx device in this example). The Peripheral Interfaces grid indicates a conflict between the I2C3 and UART4 interfaces (RED status). This conflict occurs if more than one signal has been selected on the same device ball. There is a conflict at Ball ID AA21. R-Click and select View Pins in the Peripheral Interfaces grid on the I2C3 and UART4 interfaces to open the Interface Views for these two interfaces. These views show all the I2C3 and UART4 signals. Note that i2c3_sda and i2c3_scl are multiplexed to two

different locations (the same names occur twice). In some devices, such as AM335x multi-muxed signals have an added suffix, so that these

would appear as i2c3_scl_mux0 & i2c3_sda_mux0 and i2c3_scl_mux1 and i2c3_sda_mux1 to more clearly show that the same internal signals are

multiplexed to two locations.



I2C3 Interface View

Note: Pad Config is [Signal ID Type] [Input Enable] [Active Pull] Deselect Interface

Bold Signals are Interface Members

Pad Config.	Bot/Top Ball	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
IO IEN PD	Y22 / -	code_data0			i2c3_sda	qpio_99			safe_mode
Conflict	AA21 / -	code_field	code_data8	uart4_tx	i2c3_scl	qpio_95	hw_dbq1		safe_mode
I IEN PU	B9 / -	hdq_sio	sys_alotlk	i2c2_scbbe	i2c3_scbbe	qpio_170			safe_mode
I IEN PU	W16 / -	i2c3_scl				qpio_184			safe_mode
I IEN PU	W17 / -	i2c3_sda				qpio_185			safe_mode

UART4 Interface View

Note: Pad Config is [Signal ID Type] [Input Enable] [Active Pull] Deselect Interface

Bold Signals are Interface Members

Pad Config.	Bot/Top Ball	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
Conflict	AA21 / -	code_field	code_data8	uart4_tx	i2c3_scl	qpio_95	hw_dbq1		safe_mode
O IDIS PD	Y21 / -	code_hd		uart4_rts		qpio_96			safe_mode
I IEN PD	Y22 / -	code_vd		uart4_cts		qpio_97	hw_dbq2		safe_mode
I IEN PD	W21 / -	code_wen	code_data9	uart4_rx		qpio_98	hw_dbq3		safe_mode
I IEN PU	AA16 / -	qpmc_wait1	uart4_tx			qpio_63			safe_mode
I IEN PU	Y14 / -	qpmc_wait2	uart4_rx			qpio_64			safe_mode
I IEN PD	M21 / -	mme2_clk	mespi3_clk	uart4_cts		qpio_130			safe_mode
I IEN PU	M20 / -	mme2_cnd	mespi3_simo	uart4_rts		qpio_131			safe_mode
I IEN PU	K20 / -	mme2_dat0	mespi3_somi	uart4_tx		qpio_132			safe_mode
I IEN PU	L19 / -	mme2_dat1		uart4_rx		qpio_133			safe_mode

An example solution to resolving this conflict is shown below. In the I2C3 interface view double click i2c3_sda and i2c3_scl in the Mode 3 column to deselect those

signals. Double-click i2c3_scl and i2c3_sda in the Mode 0 column to resolve the conflict. I2C3 is now selected on balls W16 and W17 and there is no more conflict

with the UART4 interface. For devices without IO Set support the tool does not restrict which signals are chosen.

For those devices, it is generally best to select

signals that are grouped together in the same mux mode as shown below.

Note: Pad Config is [Signal ID Type] [Input Enable] [Active Pull] Deselect Interface

Bold Signals are Interface Members

Pad Config.	Bot/Top Ball	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
I IEN PD	W22 / -	code_data0			i2c3_sda	qpio_99			safe_mode
O IDIS PD	AA21 / -	code_field	code_data8	uart4_tx	i2c3_scl	qpio_95	hw_dbq1		safe_mode
I IEN PU	B9 / -	hdq_sio	sys_altclk	i2c2_sccbe	i2c3_sccbe	qpio_170			safe_mode
IO IEN PU	W16 / -	i2c3_scl				qpio_184			safe_mode
IO IEN PU	W17 / -	i2c3_sda				qpio_185			safe_mode

Interface View with Multi-Muxed Violation (all device types)

Below is an example (AM35xx device) of a interface with a Multi-Muxed Violation. The same name (i2c3_sda appears twice in

the interface view in bold text meaning that the same internal signal is muxed to two different locations.) The Peripheral

Interfaces grid flags a Multi-Muxed Violation (VIOLET status) if the same internal signal is selected in more than one ball location.

i2c3_sda is selected at ball W22 and at ball W17. The interface view shows the status text "Multi-Muxed Violation: i2c3_sda, i2c3_sda

to indicate the first pair of conflicting multi-muxed signals that were found. To remove this violation, double-click i2c3_sda in the Mode 3 column to

deselect it. For devices like AM335x, the multi-muxed signals would have a suffix and would be shown as i2c3_sda_mux0

and i2c3_sda_mux1 and selecting both of those would cause a multi-muxed violation.

Peripheral Interfaces

GPT9	HDQ	HECC1	HSUSB1
HSUSB2	HW	I2C1	I2C2
I2C3	JTAG	MCBSP	MCBSP1
MCBSP2	MCBSP3	MCBSP4	MCBSP5
MCSPI1	MCSPI2	MCSPI3	MCSPI4
MM_FSUSB1	MM_FSUSB2	MM_FSUSB3	MMC1
MMC2	MMC3	RMII	SDRC
SYS	UART1	UART2	UART3
UART4	USB0		

Multi Muxed Violation: i2c3_sda, i2c3_sda Dese

Note: Pad Config is [Signal ID Type] [Input Enable] [Active Pull] Bold Signals are Interfac

Pad Config.	Bot/Top Ball	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
IO IEN PD	W22 / -	code_data0			i2c3_sda	qpio_99			safe_mode
O IDIS PD	AA21 / -	code_field	code_data8	uart4_tx	i2c3_scl	qpio_95	hw_dbq1		safe_mode
I IEN PU	B9 / -	hdq_sio	sys_altclk	i2c2_sccbe	i2c3_sccbe	qpio_170			safe_mode
IO IEN PU	W16 / -	i2c3_scl				qpio_184			safe_mode
IO IEN PU	W17 / -	i2c3_sda				qpio_185			safe_mode

Repeated Rows in Interface Views

The interface view highlights all the member signals of a peripheral interface (in bold text) -- one signal per row.

In cases

where there is more than one interface member signal at the same ball, that ball will be repeated in the interface view.

In the example below, SPI1_CS1_MUX1 and SPI1_SCLK_MUX0 are both muxed to ball C18, and ball C18 is repeated in

the interface view. The repeated rows behave as a single row - if a cell is changed in either row, the change occurs in both

rows.

IO Sets Functionality (AM335x device only)

Currently, only the AM335x device provides IO Set functionality. Instead of allowing selection of any set of interface

member signals, limited pre-defined sets of interface member signals are allowed. Information describing the available

IO Sets is read in after the AM335x device type is selected at program startup. The IO Sets are used to guide the user

into using a tested configuration that meets timing requirements. Either all or a subset of the interface member signals

specified by an IO Set may be selected. See the legend description and examples below for a description of the peripheral

interface statuses IO Set Match, IO Set Subset and IO Set Violation.

Interface View With IO Set Support (AM335x device only)

Below is an example of an interface view that provides IO Sets (the I2C2 Interface View for the AM335x device). The IO Set information for each peripheral

interface is read in by the program after the device type is selected. The IO Sets information is used to limit the selections to a supported group of pin-muxing

configurations.

In this example, the following IO Sets are pre-defined:

IO Set #1: I2C2_SCL_MUX0 & I2C2_SDA_MUX0

IO Set #2: I2C2_SCL_MUX1 & I2C2_SDA_MUX1

IO Set #3: I2C2_SCL_MUX2 & I2C2_SDA_MUX2

For interfaces that provide IO Sets, there are additional GUI controls as shown below:

Select IO Set to Apply (drop-down box)

You can see how many IO Sets are defined and choose a particular IO Set. Click Apply IO Set button and the currently selected interface member signals will be cleared and the selected IO Set will get selected.

Prev IO Set and Next IO Set Buttons

Use these buttons to quickly jump from one defined IO Set to the next or previous IO Set.

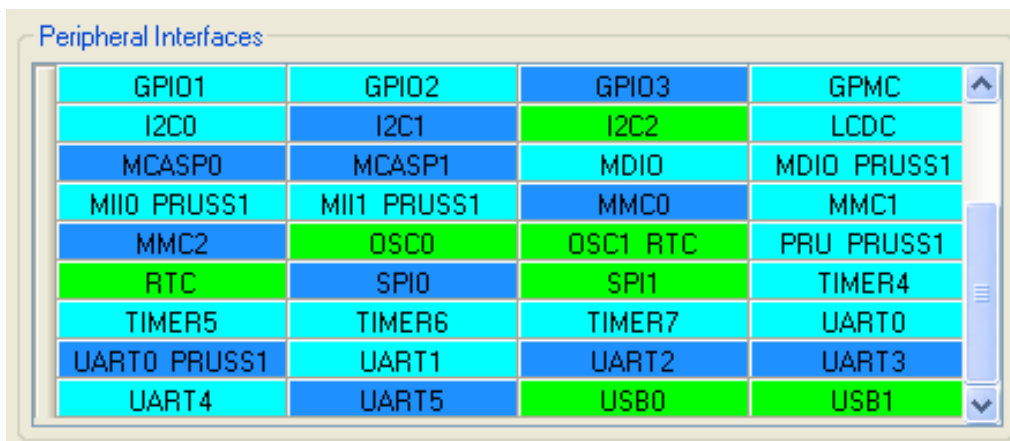
Deselect Interface Button

This button deselects all interface member signals and sets each ball to the Selected (Ball Available) state.

(Some signals do not have a programmable mux mode - for example they always have Mode 0 selected. These are not changed by the Deselect Interface button.)

IO Set Match Status

In this example the SPI1 interface has the IO Set Match (GREEN) status. This means that the signals selected for SPI1 match one of the provided IO Sets exactly.



The screenshot shows a table titled "Peripheral Interfaces" with 12 rows and 4 columns. The rows are: GPIO1, GPIO2, GPIO3, GPMC; I2C0, I2C1, I2C2, LCDC; MCASP0, MCASP1, MDIO, MDIO PRUSS1; MII0 PRUSS1, MII1 PRUSS1, MMC0, MMC1; MMC2, OSC0, OSC1 RTC, PRU PRUSS1; RTC, SPI0, SPI1, TIMER4; TIMER5, TIMER6, TIMER7, UART0; UART0 PRUSS1, UART1, UART2, UART3; UART4, UART5, USB0, USB1. The SPI1 cell is highlighted in green, indicating an IO Set Match status.

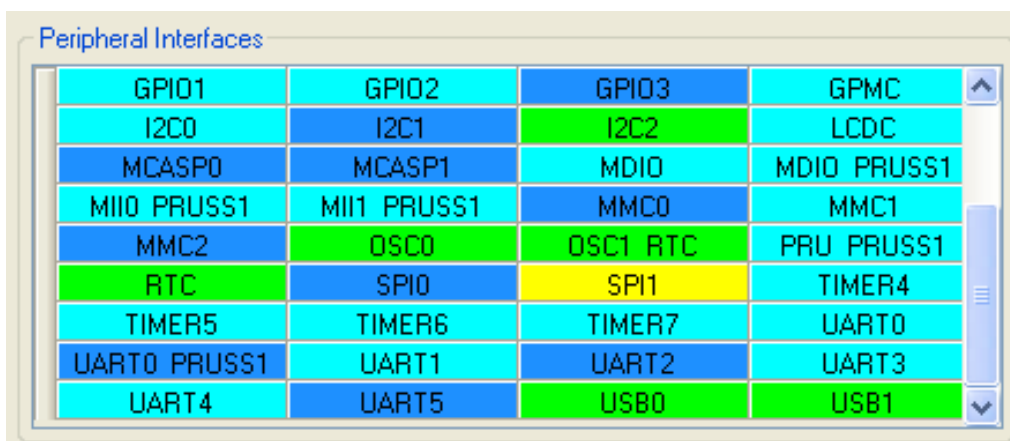
GPIO1	GPIO2	GPIO3	GPMC
I2C0	I2C1	I2C2	LCDC
MCASP0	MCASP1	MDIO	MDIO PRUSS1
MII0 PRUSS1	MII1 PRUSS1	MMC0	MMC1
MMC2	OSC0	OSC1 RTC	PRU PRUSS1
RTC	SPI0	SPI1	TIMER4
TIMER5	TIMER6	TIMER7	UART0
UART0 PRUSS1	UART1	UART2	UART3
UART4	UART5	USB0	USB1

In the SPI1 interface view, the user has set the drop-down box to "IO Set 1" and has clicked the Apply IO Set button. The IO Set status text reads

"Current selections match IO Set #1 exactly."

IO Set Subset Status

In this example the SPI1 interface has the IO Set Subset (YELLOW) status. This means that the signals selected for SPI1 comprise a subset of one or more pre-defined IO Sets but does not match any IO Set exactly. This is an acceptable status. In this case there are signals in the IO Set that have been deselected because they are not needed in the design. Since there is not an IO Set Violation status, there is no violation of the defined IO Sets.



The screenshot shows the same "Peripheral Interfaces" table as above. In this instance, the SPI1 cell is highlighted in yellow, indicating an IO Set Subset status.

GPIO1	GPIO2	GPIO3	GPMC
I2C0	I2C1	I2C2	LCDC
MCASP0	MCASP1	MDIO	MDIO PRUSS1
MII0 PRUSS1	MII1 PRUSS1	MMC0	MMC1
MMC2	OSC0	OSC1 RTC	PRU PRUSS1
RTC	SPI0	SPI1	TIMER4
TIMER5	TIMER6	TIMER7	UART0
UART0 PRUSS1	UART1	UART2	UART3
UART4	UART5	USB0	USB1

In the SPI1 interface view, the user has selected IO Set 1, then double-clicked the second chip select (SPI1_CS1_MUX1) to deselect it because this signal is

not required in the design. The IO Set status text reads "Current selections are a subset of IO Set(s) 1". This is an acceptable status.

IO Set Violation Status

In this example the SPI1 interface has the IO Set Violation (ORANGE) status. This means that the signals selected for SPI1 are not all contained in any of the pre-defined IO Sets. The pre-defined IO Sets are designed to screen out combinations of signal selections that are not allowed to be used together. This status needs to be eliminated before arriving

at a final Pin Mux solution.

Peripheral Interfaces			
GPIO1	GPIO2	GPIO3	GPMC
I2C0	I2C1	I2C2	LCDC
MCASP0	MCASP1	MDIO	MDIO PRUSS1
MII0 PRUSS1	MII1 PRUSS1	MMC0	MMC1
MMC2	OSC0	OSC1 RTC	PRU PRUSS1
RTC	SPI0	SPI1	TIMER4
TIMER5	TIMER6	TIMER7	UART0
UART0 PRUSS1	UART1	UART2	UART3
UART4	UART5	USB0	USB1

In the SPI1 interface view, the user has selected IO Set 1, then the user deselected SPI1_D0_MUX1 and instead selected SPI1_D0_MUX0. There is no IO Set that uses both SPI1_D0_MUX0 and SPI1_D1_MUX1 at the same time.

The IO Set status text reads "IO Set Violation!" This status must be eliminated. This can be done by selecting one of the pre-defined IO Sets.

Interface Views without IO Set Support

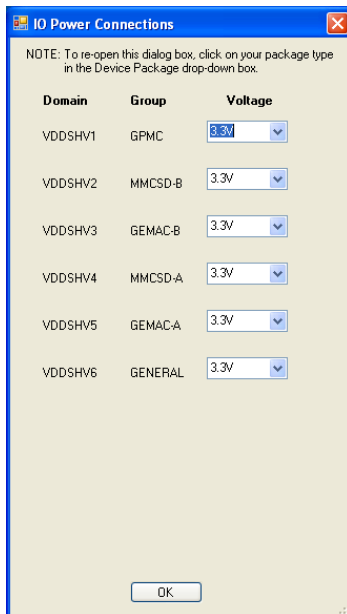
With the AM335x device some interface views do not use IO Sets. This example is called the MISC interface. These signals have been separated out here because they are not subject to the rules about being connected to the same IO power voltages (as are all other signals in an MMCx interface).

IO Cell Power Domain Checking (AM335x device only)

For AM335x, the GUI includes a column that displays the IO power supply domain name and voltage for each ball of the device.

At program startup, the user is prompted to select the package type and then the voltage that will be applied to the IO power

supply domains that support more than one voltage as shown below.

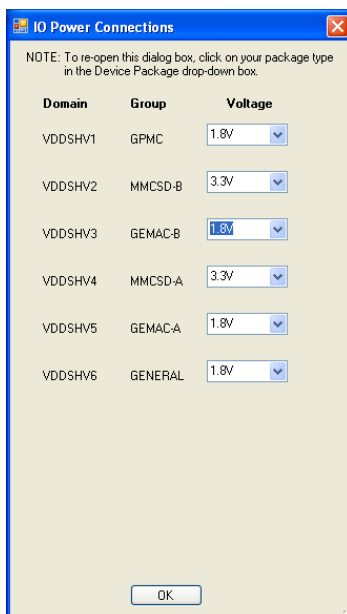


IO Power Violation Status (AM335x device only)

In this example, the AM335x device is used and the 15 x 15 ZCZ package type is selected.

At program startup, you will be prompted for the package type and to specify the IO power supply voltages. The voltages can be changed later by clicking the Change Voltages button in the Main Window or it will automatically open when the package type drop-down is changed.

In this example, the IO power supplies are selected to use 1.8V for most of the IO power domains, and to use 3.3V for the MMC IO power groups as shown below.



The Peripheral Interfaces grid indicates an IO Power Violation (BROWN) status for the MMC0 interface.

Peripheral Interfaces

ADC	ARM	DCAN0	DCAN1
DEBUGSS	ECAP0	ECAP0 PRUSS1	ECAP1
ECAP2	ECAT PRUSS1	EHRPWM0	EHRPWM1
EHRPWM2	EMIF4	EQEP0	EQEP1
EQEP2	GEMAC CPSW	GLUE	GPI00
GPI01	GPI02	GPI03	GPMC
I2C0	I2C1	I2C2	LCDC
MCASP0	MCASP1	MDIO	MDIO PRUSS1
MII0 PRUSS1	MII1 PRUSS1	MMC0	MMC1

In the MMC0 Interface View you can see in the IO Power column that only MMC0_DAT(3-0), MMC0_CLK and MMC0_CMD have their IO cells

supplied by VDDSHV4 (which user set to 3.3V). The remaining MMC0 signals are connected to 1.8V IO cell power. Selection of the MMC0_DAT(7-0)

signals has caused an IO Power Violation, because not all of the selected interface member signals are connected to the same IO power supply voltage.

(Using multiple IO Power domains is OK, as long as they are the same voltage.) In this example, MMC0 would be limited to operation with a 4-bit data bus.

MMC0 Interface View

Select IO Set to Apply: Current selections match IO Set #1 exactly.

Apply IO Set Prev IO Set Next IO Set Deselect Interface

Note: Pad Config is [Signal IO Type] [RX/ACTIVE] [Internal Pull]

Bold Signals are Interface Members

Pad Config.	Bot/Top Ball	IO Power	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
IO IEN PU	F17 /-	VDDSHV4=1.8V	MMC0_DAT3	GPMC_A20_MUX1	UART4_CTSN_MU...	TIMERS_MUX0	UART1_DCDN_M...	PR1_PRU0_PRU...	PR1_PRU0_PRU...	GPI02I26I
IO IEN PU	F18 /-	VDDSHV4=1.8V	MMC0_DAT2	GPMC_A21_MUX1	UART4_RTSN_MU...	TIMER6_MUX0	UART1_DSRN_M...	PR1_PRU0_PRU...	PR1_PRU0_PRU...	GPI02I27I
IO IEN PU	G15 /-	VDDSHV4=1.8V	MMC0_DAT1	GPMC_A22_MUX1	UART5_CTSN_MU...	UART3_RXD_MUX2	UART1_DTRN_MU...	PR1_PRU0_PRU...	PR1_PRU0_PRU...	GPI02I28I
IO IEN PU	G16 /-	VDDSHV4=1.8V	MMC0_DAT0	GPMC_A23_MUX1	UART5_RTSN_MU...	UART3_TXD_MUX2	UART1_RIN_MUX1	PR1_PRU0_PRU...	PR1_PRU0_PRU...	GPI02I29I
IO IEN PU	G17 /-	VDDSHV4=1.8V	MMC0_CLK	GPMC_A24_MUX1	UART3_CTSN_MU...	UART2_RXD_MUX2	DCAN1_TX_MUX2	PR1_PRU0_PRU...	PR1_PRU0_PRU...	GPI02I30I
IO IEN PU	G18 /-	VDDSHV4=1.8V	MMC0_CMD	GPMC_A25_MUX1	UART3_RTSN_MU...	UART2_TXD_MUX2	DCAN1_RX_MUX2	PR1_PRU0_PRU...	PR1_PRU0_PRU...	GPI02I31I
IO IEN PD	K18 /-	VDDSHV5=3.3V	GMI11_TXCLK	UART2_RXD_MUX0	RGMI11_TCLK	MMC0_DAT7	MMC1_DAT0_MU...	UART1_DCDN_M...	MCASP0_ACLKX...	GPI03I9I
IO IEN PD	L18 /-	VDDSHV5=3.3V	GMI11_RXCLK	UART2_RXD_MUX0	RGMI11_RCLK	MMC0_DAT6	MMC1_DAT1_MU...	UART1_DSRN_M...	MCASP0_FSX_M...	GPI03I10I
IO IEN PD	L17 /-	VDDSHV5=3.3V	GMI11_RXD3	UART3_RXD_MUX0	RGMI11_RD3	MMC0_DAT5	MMC1_DAT2_MU...	UART1_DTRN_MU...	MCASP0_AXR0...	GPI02I18I
IO IEN PD	L16 /-	VDDSHV5=3.3V	GMI11_RXD2	UART3_TXD_MUX0	RGMI11_RD2	MMC0_DAT4	MMC1_DAT3_MU...	UART1_RIN_MUX0	MCASP0_AXR1...	GPI02I19I

Changing Pad Configuration Parameters

Pad configuration parameters are used to set the values of other bit fields in each Pad Configuration Register.

The Pad Configuration parameter values are shown in the first column, labeled Pad Config, in the Pin Mux Grid.

The first parameter is always the signal direction (I, O or IO). This value is set depending on the signal direction associated with the selected signal in the row. This parameter cannot be otherwise edited. The second parameter (RX_ACTIVE) determines if the logic input is driven by the signal at the I/O pad. This parameter defaults to Input Enabled

(IEN) if the selected signal has a direction of I or IO, and defaults to Input Disabled (IDIS) if the selected signal has a

direction of Output-Only (O). Theoretically, all pads could be set to IEN, but if the signal is output-only it can be set to IDIS, and this provides some power savings. RX_ACTIVE and the remaining parameters can be programmed by the

Pad Configuration Editor. The Internal Pull parameter (PD, PU, OFF) defaults to the value in the "PUPD State During

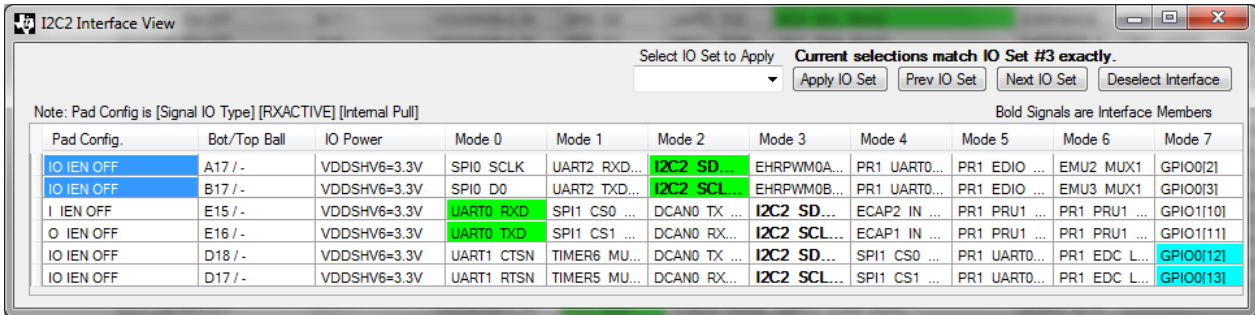
HHV Reset" column in the device data spreadsheet for each pad. To open the Pad Configuration Editor, select one or

more cells in the Pad Config column (by holding down the left mouse button and dragging downwards). For example,

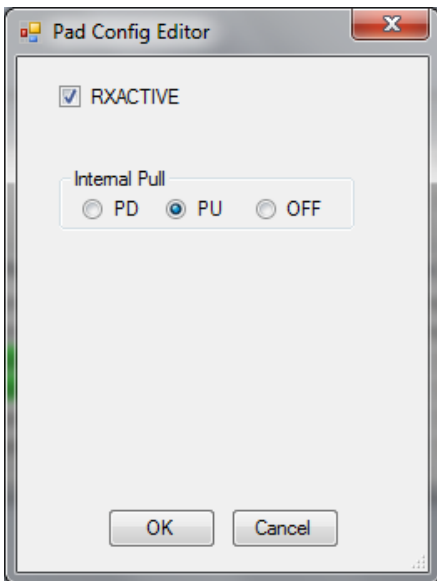
you can select an entire 32-bit data bus and set them all to the same pad configuration at once. In the example below,

a pair of pads with I2C signals selected have been highlighted for pad config editing. R-Click in the highlighted area and select

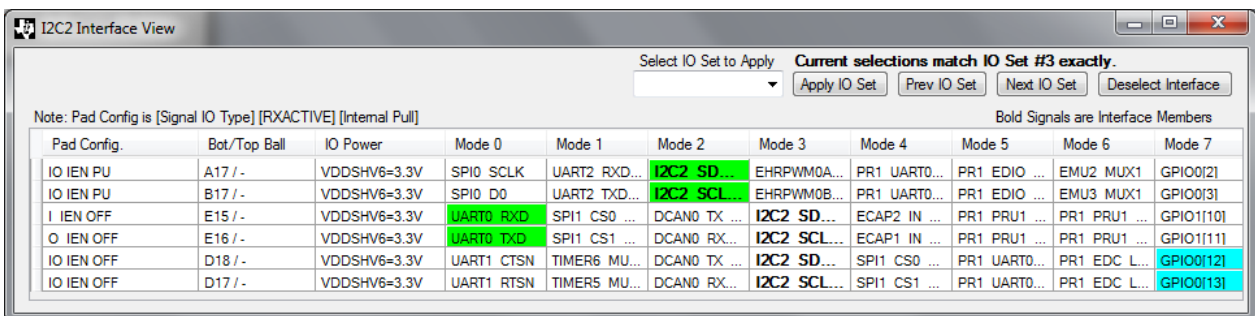
Edit Selected Pad Configs to open the Pad Config Editor.



Here the user has changed the Internal Pull from OFF to PU (pull-up). Click OK.

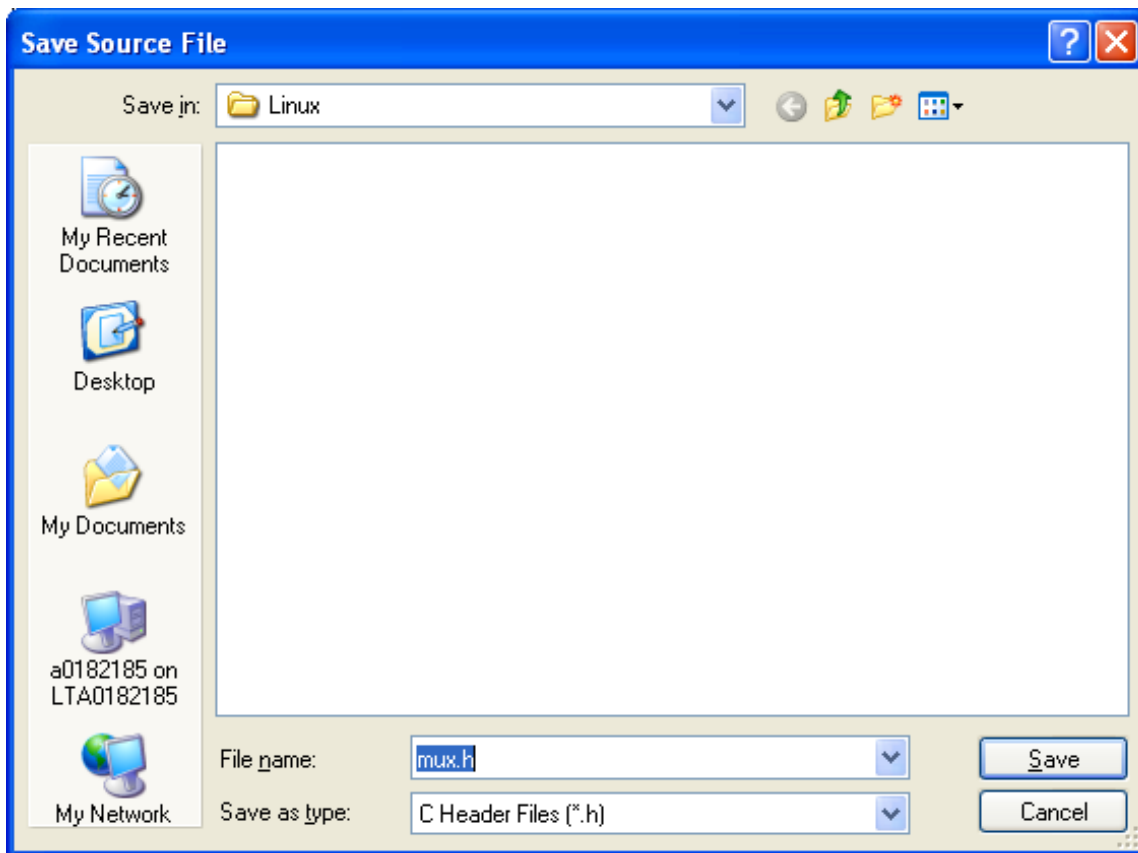


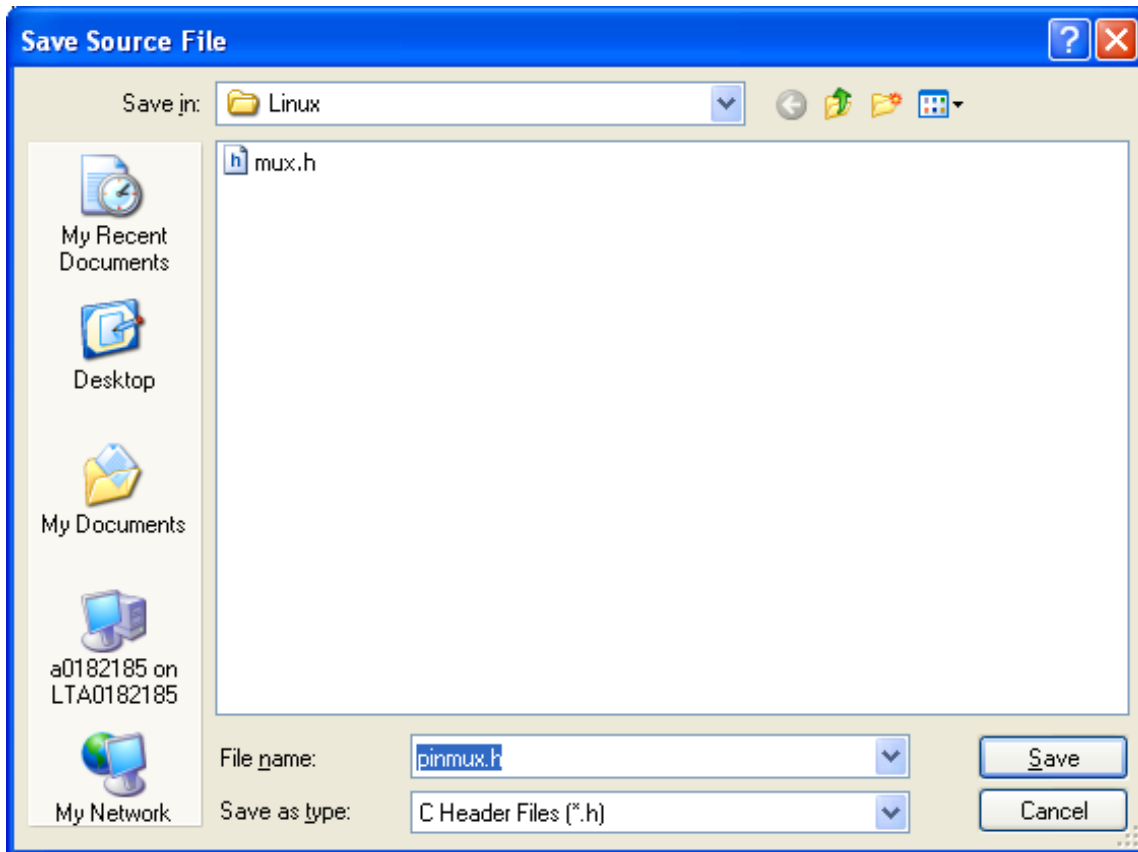
Notice that these two pads have had the third parameter changed to PU



Saving Results to Source Files

Click the "File > Save > Source > Linux" menu item to save results as C header files. File save dialogs will open to specify the file names for a device-dependent header file (default name: mux.h) and a board-dependent header file (default name: pinmux.h).





Examples of the device-dependent and board-dependent header files are shown below.

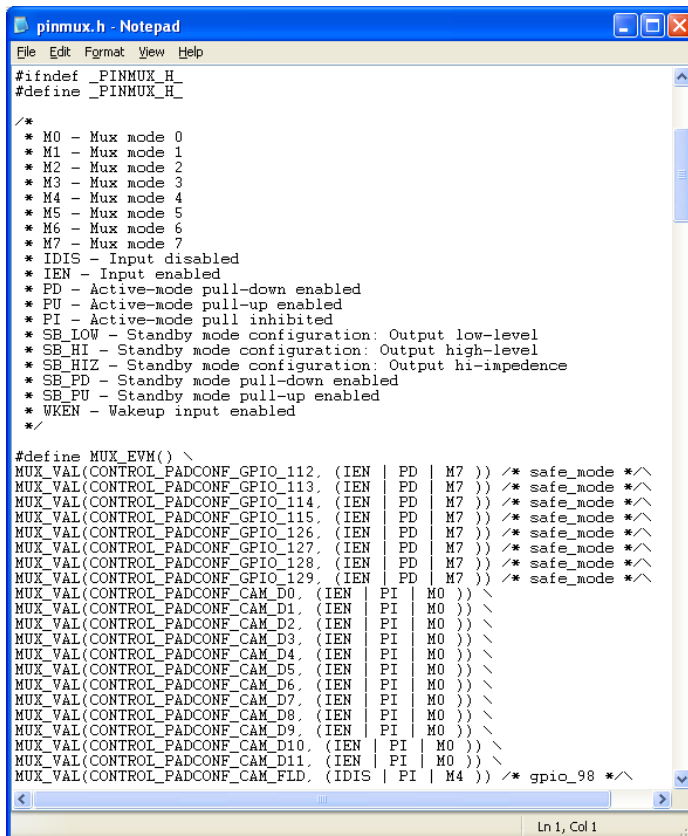
```

mux.h - Notepad
File Edit Format View Help
#define M0 0
#define M1 1
#define M2 2
#define M3 3
#define M4 4
#define M5 5
#define M6 6
#define M7 7
#define IDIS (0 << 8)
#define IEN (1 << 8)
#define PD (1 << 3)
#define PU (3 << 3)
#define PI (0 << 3)
#define SB_SIG (0 << 9)
#define SB_LOW (1 << 9)
#define SB_HI (5 << 9)
#define SB_HIZ (2 << 9)
#define SB_PD (1 << 12)
#define SB_PU (3 << 12)
#define SB_PI (0 << 12)
#define WKEN (1 << 14)

/*
 * To get the physical address the offset has
 * to be added to OMAP34XX_CTRL_BASE
 */

#define CONTROL_PADCONF_GPIO_112 0x0134
#define CONTROL_PADCONF_GPIO_113 0x0136
#define CONTROL_PADCONF_GPIO_114 0x0138
#define CONTROL_PADCONF_GPIO_115 0x013A
#define CONTROL_PADCONF_GPIO_126 0x0A54
#define CONTROL_PADCONF_GPIO_127 0x0A56
#define CONTROL_PADCONF_GPIO_128 0x0A58
#define CONTROL_PADCONF_GPIO_129 0x0A5A
#define CONTROL_PADCONF_CAM_D0 0x0116
#define CONTROL_PADCONF_CAM_D1 0x0118
#define CONTROL_PADCONF_CAM_D2 0x011A
#define CONTROL_PADCONF_CAM_D3 0x011C
#define CONTROL_PADCONF_CAM_D4 0x011E
#define CONTROL_PADCONF_CAM_D5 0x0120
#define CONTROL_PADCONF_CAM_D6 0x0122
#define CONTROL_PADCONF_CAM_D7 0x0124
#define CONTROL_PADCONF_CAM_D8 0x0126

```



```

pinmux.h - Notepad
File Edit Format View Help
#ifndef _PINMUX_H_
#define _PINMUX_H_

/*
 * M0 - Mux mode 0
 * M1 - Mux mode 1
 * M2 - Mux mode 2
 * M3 - Mux mode 3
 * M4 - Mux mode 4
 * M5 - Mux mode 5
 * M6 - Mux mode 6
 * M7 - Mux mode 7
 * IDIS - Input disabled
 * IEN - Input enabled
 * PD - Active-mode pull-down enabled
 * PU - Active-mode pull-up enabled
 * PI - Active-mode pull-inhibited
 * SB_LOW - Standby mode configuration: Output low-level
 * SB_HI - Standby mode configuration: Output high-level
 * SB_HIZ - Standby mode configuration: Output hi-impedance
 * SB_PD - Standby mode pull-down enabled
 * SB_PU - Standby mode pull-up enabled
 * WKEN - Wakeup input enabled
 */

#define MUX_EVM() \
MUX_VAL(CONTROL_PADCONF_GPIO_112, (IEN | PD | M7 )) /* safe_mode */\
MUX_VAL(CONTROL_PADCONF_GPIO_113, (IEN | PD | M7 )) /* safe_mode */\
MUX_VAL(CONTROL_PADCONF_GPIO_114, (IEN | PD | M7 )) /* safe_mode */\
MUX_VAL(CONTROL_PADCONF_GPIO_115, (IEN | PD | M7 )) /* safe_mode */\
MUX_VAL(CONTROL_PADCONF_GPIO_126, (IEN | PD | M7 )) /* safe_mode */\
MUX_VAL(CONTROL_PADCONF_GPIO_127, (IEN | PD | M7 )) /* safe_mode */\
MUX_VAL(CONTROL_PADCONF_GPIO_128, (IEN | PD | M7 )) /* safe_mode */\
MUX_VAL(CONTROL_PADCONF_GPIO_129, (IEN | PD | M7 )) /* safe_mode */\
MUX_VAL(CONTROL_PADCONF_CAM_D0, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D1, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D2, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D3, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D4, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D5, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D6, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D7, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D8, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D9, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D10, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_D11, (IEN | PI | M0 )) \
MUX_VAL(CONTROL_PADCONF_CAM_FLD, (IDIS | PI | M4 )) /* gpio_98 */\

```

Using the Generated Header Files (AM35x / AM37x / OMAP35x)

The generated header files are in the format used for the AM35x / AM37x / OMAP35x U-Boot source code.

For other supported devices, the source code provides a complete description of all settings but is not directly usable as U-Boot source code.

For AM35x / AM37x / OMAP35x the generated pinmux settings may be used to customize the U-Boot source code.

Before rebuilding U-BOOT for your system, the following steps are necessary:

- 1) Replace the mux.h header file with the mux.h output file from Pin Mux Utility
- 2) Copy the pinmux.h output file from Pin Mux Utility into the directory containing the evm.h file.
- 3) Modify the original evm.h file, commenting out or deleting the original section of code that makes the pin mux programming macro calls.
- 4) Replace this code with `#include "pinmux.h"`

The following shows modifications to the original evm.h file to include pinmux settings from a separate header file.

```

/*
#define MUX_EVM() \
MUX_VAL(CP(GPIO_112), (IEN | PTD | EN | M7 )) /* safe_mode */ \
MUX_VAL(CP(GPIO_113), (IEN | PTD | EN | M7 )) /* safe_mode */ \
MUX_VAL(CP(GPIO_114), (IEN | PTD | EN | M7 )) /* safe_mode */ \
MUX_VAL(CP(GPIO_115), (IEN | PTD | EN | M7 )) /* safe_mode */ \
MUX_VAL(CP(GPIO_126), (IEN | PTD | EN | M7 )) /* safe_mode */ \
MUX_VAL(CP(GPIO_127), (IEN | PTD | EN | M7 )) /* safe_mode */ \
MUX_VAL(CP(GPIO_128), (IEN | PTD | EN | M7 )) /* safe_mode */ \
MUX_VAL(CP(GPIO_129), (IEN | PTD | EN | M7 )) /* safe_mode */ \
MUX_VAL(CP(CAM_D0), (IEN | PTD | DIS | M0 )) \
...
MUX_VAL(CP(UART3_TX_IRTX), (IDIS | PTD | DIS | M0 ))
*/
#include "pinmux.h"
#endif

```

Locations for U-Boot Linux Source Files

Device Type	Device-Dependent Header File	Board-Dependent Header File
OMAP35xx	arch/arm/include/asm/arch-omap3/mux.h	board/ti/evm/evm.h board/ti/evm/pinmux.h (new)
AM37xx	arch/arm/include/asm/arch-omap3/mux.h	board/ti/evm/evm.h board/ti/evm/pinmux.h (new)
AM35xx	arch/arm/include/asm/arch-omap3/mux.h	board/logicpd/am3517evm/am3517evm.h board/logicpd/am3517evm/pinmux.h (new)

The original mux.h file includes register name defines for OMAP35xx, AM35xx and AM37xx. By setting the configuration

for the make utility, U-BOOT can be built for any of these platforms. However, the Pin Mux Utility generated mux.h file

will only contain register name defines for the device that was selected when Pin Mux Utility was run. So, U-BOOT can be

rebuilt for that selected device only.

See also: [Pinmux Utilities for Davinci Processors](#)

References

- [1] http://processors.wiki.ti.com/index.php?title=Pin_Mux_Utility_for_ARM_MPU_Processors&oldid=58259
- [2] <http://focus.ti.com/docs/toolsw/folders/print/pinmuxtool.html>
- [3] <http://focus.ti.com/docs/toolsw/folders/print/linuxsdk-am37x.html>

Pin Setup Tool for AM18xx ARM Microprocessors

Return to the [Sitara Linux Software Developer's Guide](#)



Introduction

The Pin Setup Tool for AM18xx ARM Microprocessors is a Windows-based software tool for configuring pin multiplexing settings for the AM1802, AM1806, AM1808 and AM1810 devices.

Pin Setup provides a graphical user interface for selecting the peripherals that will be used in the system design and for resolving pin multiplexing conflicts. Results are saved as a C header file that includes a list of all selected signals and settings for the pin mux registers.

Software Installation

The Pin Setup program is a Windows application. It can be obtained from:

1. Download ZIP archive containing the executable and associated data from here [Pin Setup for AM18xx](#) ^[1]. This is most frequently updated.

To install, simply unzip this archive to any directory on a PC running the Windows OS. For this document, we have unzipped this

to a directory named C:\PinSetup.

OR

2. If the above link does not work, you can obtain the same ZIP archive on www.ti.com ^[3] on any of the AM18xx product pages in the Application Notes section. Look for "AM18xx Pin Muxing Utility".

OR

3. The program is installed on your Ubuntu Linux host when an AM18x SDK is installed. Copy the entire directory structure from your Ubuntu Linux host

to a PC running the Windows O/S (which could be the same PC if you are running a virtual Linux machine under the Windows O/S). You would need

to transfer the directory structure by copying it under a virtual machine shared folder or by using a Samba server.

Copy the entire directory structure at the location below to any directory on the Windows side (suggest: C:\PinSetup*.*)

```
/home/user/ti-sdk-am180x-evm-4.0.1.0/host-tools/pinmux_utils/windows
```

OR

```
/home/user/ti-sdk-am181x-evm-4.0.1.0/host-tools/pinmux_utils/windows
```

OR

4. Running in Linux with wine (tested on Ubuntu 10.04 LTS with wine 1.2.2 and Windows mono 2.10.8):

Install the *Windows* version of mono by downloading and running the installer with wine.

```
wine ~/Downloads/mono-*.exe
```

Add C:\Program Files\Mono-VERSION\bin to your wine/windows by editing the registry key 'HKEY_CURRENT_USER/Environment/PATH'. This can be done by running

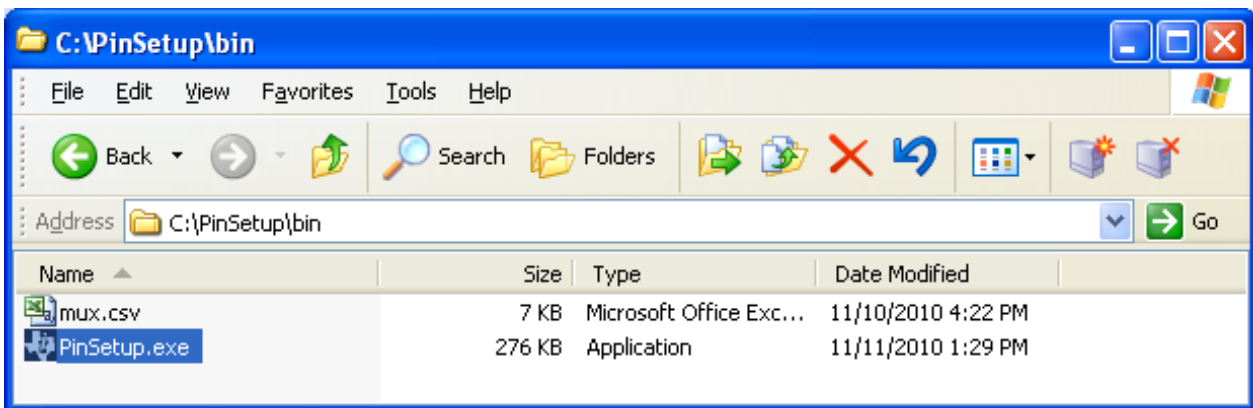
```
wine regedit
```

Run the pin setup utility through wine with windows mono by running:

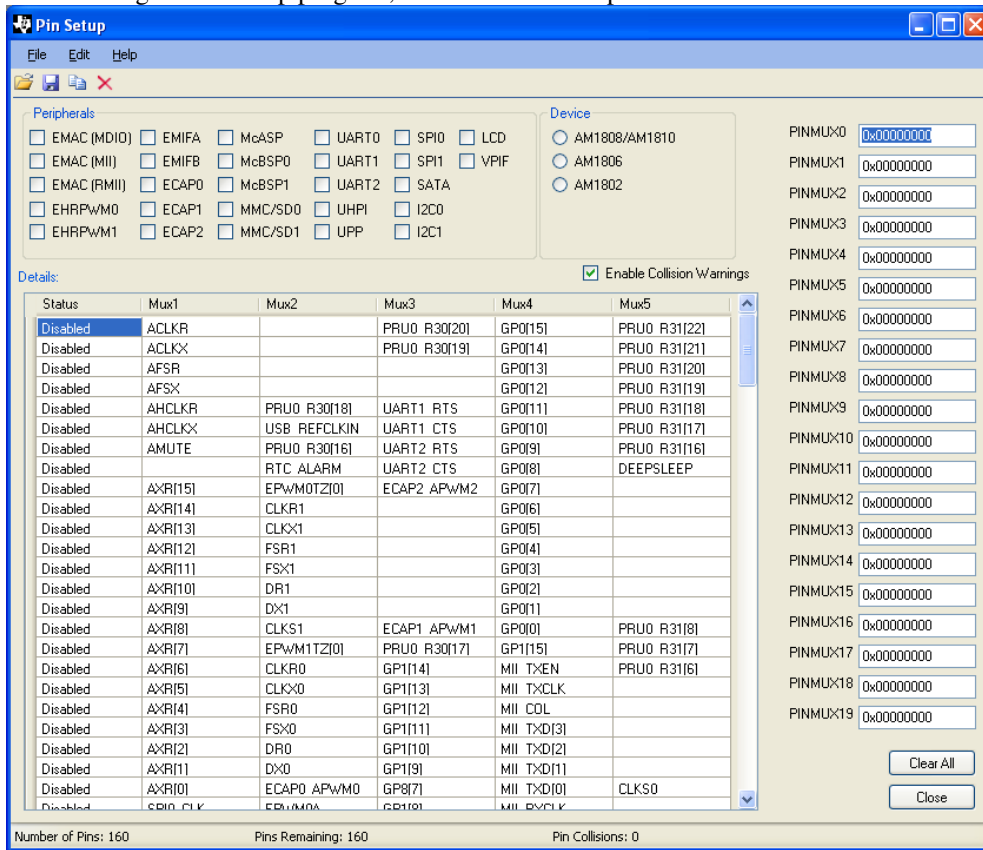
```
wine mono Pin_Setup_AM18XX/bin/PinSetup.exe
```

Starting the PinSetup Program

To start running Pin Setup, open the bin subdirectory and run the program named PinSetup.exe.



After starting the PinSetup program, the main window opens as shown below.



Using the PinSetup Program

Main Window Features

Device Selector Radio Buttons

The device selector initially shows nothing selected (it defaults to AM1808 / AM1810). AM1808 and AM1810 have the same pin muxing options and represent the superset device which uses all peripherals in the peripheral check box area. If AM1806 or AM1802 is selected, the peripheral check boxes that do not apply will be greyed out. Also, the signals in the details grid that are associated with the unavailable peripherals will have a grey background and will not be selectable. If there are unsaved pin mux selections, they can be saved using the File - Save - Pin Selections menu item before clicking a Device Selector Radio Button.

Registers Display

The registers display is specific to the pin mux architecture of the AM18xx (and similar) devices. There are 20 pin mux registers. Each register is 32-bits wide and programs the signal selection for 8 device balls. Each device ball in the details grid has up to 5 different multiplexed signals and controls one nibble of the registers display. The signal selection is programmed with a 4-bit code as shown in the table below.

Pin Mux Selection	Mux1	Mux2	Mux3	Mux4	Mux5
Binary Code	b0001	b0010	b0100	b1000	b0000

Clear All Button

The Clear All Button clears all pin mux selections that have been made. Use File - Save - Pin Selections before doing this to save work, if necessary. All pin mux registers will be set to zero, and all device balls in the details grid will indicate disabled.

Close Button

The Close Button exits the program. This can also be done using the File - Exit menu item, or by double-clicking the title bar icon, or by clicking the title bar X icon. Unsaved work must be saved first using the File - Save - Pin Selections menu item.

Details Grid

Each row of the details grid represents a programmable device ball. Device balls that only have a single signal definition are not shown in the tool. Each device ball has up to 5 selectable signal definitions shown in the Mux1 - Mux5 columns. The status column indicates "Enabled" if a signal is selected, "Disabled" if no signal is selected or "Conflict" if a previously selected signal was unselected due to a conflict (conflicts are shown with a red background). The details grid view can be scrolled up and down using the vertical scrollbar.

Enable Collision Warnings

If Enable Collision Warnings is checked, a collision warning dialog box will open each time a signal selection is attempted and another signal in the same row of the details grid is already selected. The option is given to

either not select the new signal or to unselect the existing signal and select the new signal. In the later case, the newly selected signal changes to green (selected) and the previously selected signal changes to red (conflict) and the status column indicates "Conflict".

If Enable Collision Warnings is not checked, the collision warning dialog box does not appear. When the program attempts to select new signals, the selection will only occur if there is no conflict.

Changing Pin Mux Selections

Pin Mux selections and unselections can be made in several ways.

Using Peripheral Check Boxes

The peripheral check boxes are used to select or unselect a group of signals that comprise a peripheral interface.

When a peripheral is checked, the program attempts to select all member signals of the interface.

When a peripheral is unchecked, the program unselects all member signals of the interface.

Double-Clicking Individual Cells

An individual signal in the details grid can be selected or unselected by double-clicking its cell.

Using the Context Menu

The details grid contains a context menu with the menu items shown below. Click on a cell first, to select it, before using the context menu.

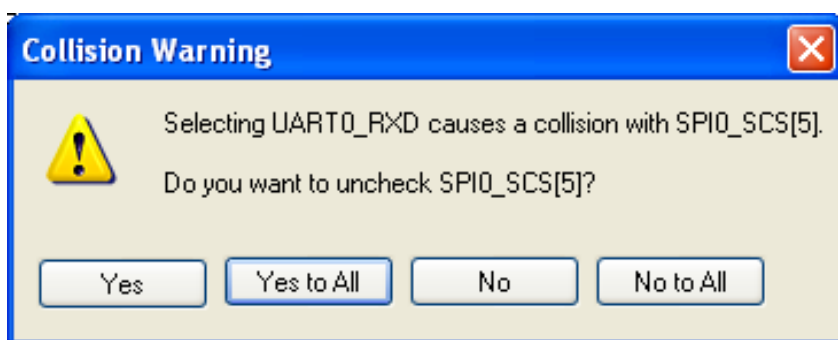
Context Menu Item	Function
Select Pin	Selects the signal
Unselect Pin	Unselects the signal
Clear Collision	Removes collision status and red background.
Find...	Open signal finder dialog box
Clear All	Clears all pin mux selections

Resolving Conflicts

As an example of conflict resolution, first select the SPI0 peripheral, then the UART0 peripheral. A collision dialog box appears

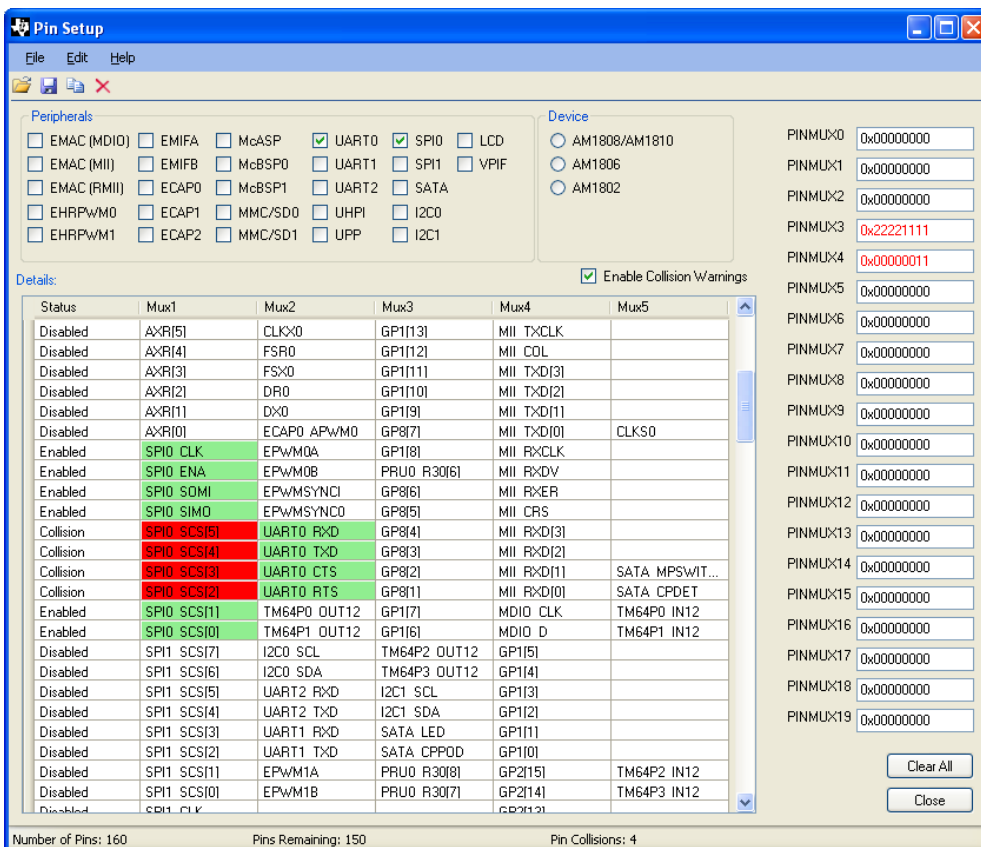
indicating a conflict. The decision to select the new signal or not can be made on a signal-by-signal basis or you can click

Yes to All or No to All.

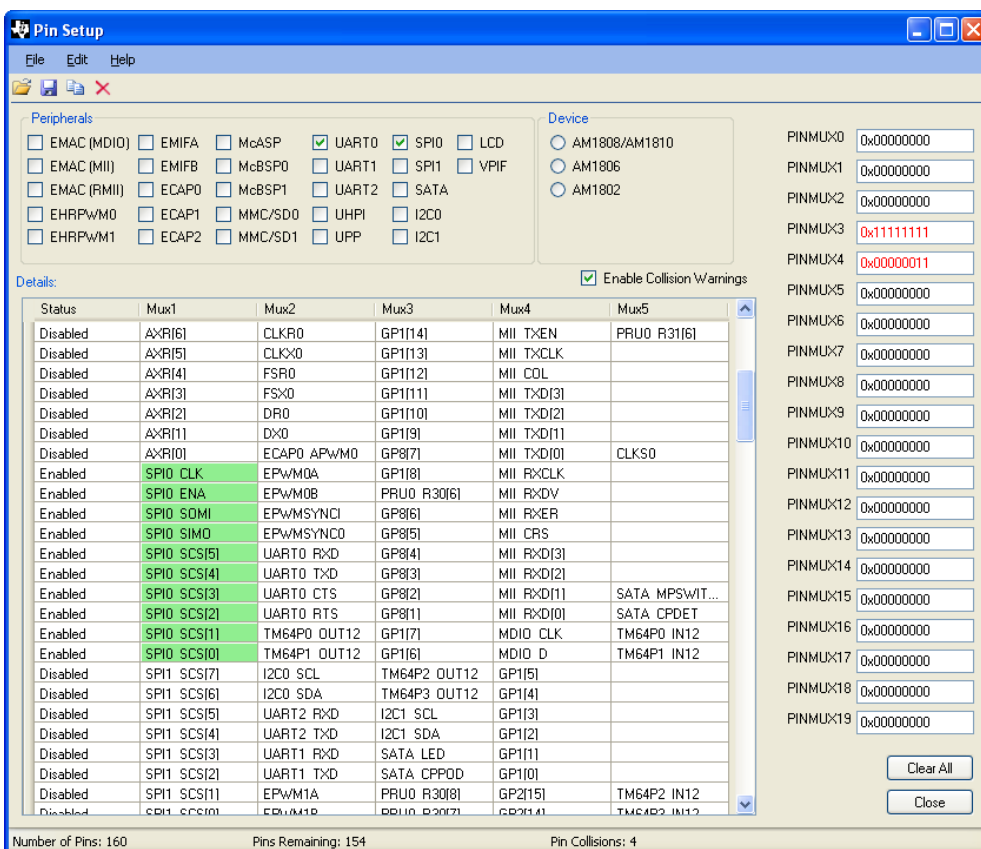


If Yes to All is selected, the UART0 signals will get selected as shown below. Note that the PINMUX3 register is selecting Mux2

for the UART0 signals. The conflicting SPI0 signals are no longer selected but are shown in red to indicate that a conflict occurred there.



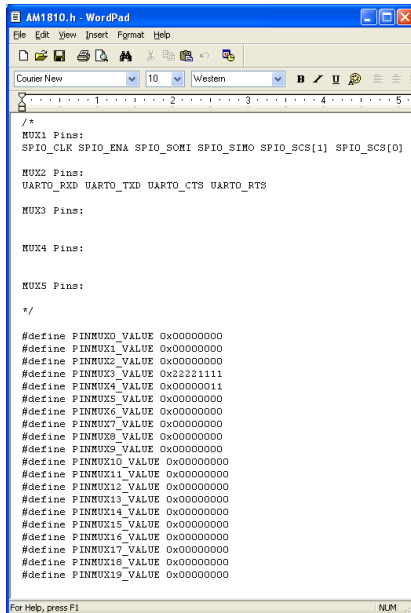
If No to All is selected, the UART0 signals do not get selected and no change is made as shown below. Note that the PINMUX3 register is selecting Mux1 for the SPIO signals.



Menu Items

File - Save - Header File

The File - Save - Header File menu item saves a list of the selected signals and #define constants for the Pin Mux register values. The default location is the My Documents folder. A file save dialog allows you to specify where to save the header file and to give the file a name.



```

/*
MUX1 Pins:
SPI0_CLK SPI0_ENA SPI0_SOMI SPI0_SINO SPI0_SCS(1) SPI0_SCS(0)

MUX2 Pins:
UART0_RXD UART0_TXD UART0_CTS UART0_RTS

MUX3 Pins:

MUX4 Pins:

MUX5 Pins:

*/

#define PINMUX0_VALUE 0x00000000
#define PINMUX1_VALUE 0x00000000
#define PINMUX2_VALUE 0x00000000
#define PINMUX3_VALUE 0x22221111
#define PINMUX4_VALUE 0x00000011
#define PINMUX5_VALUE 0x00000000
#define PINMUX6_VALUE 0x00000000
#define PINMUX7_VALUE 0x00000000
#define PINMUX8_VALUE 0x00000000
#define PINMUX9_VALUE 0x00000000
#define PINMUX10_VALUE 0x00000000
#define PINMUX11_VALUE 0x00000000
#define PINMUX12_VALUE 0x00000000
#define PINMUX13_VALUE 0x00000000
#define PINMUX14_VALUE 0x00000000
#define PINMUX15_VALUE 0x00000000
#define PINMUX16_VALUE 0x00000000
#define PINMUX17_VALUE 0x00000000
#define PINMUX18_VALUE 0x00000000
#define PINMUX19_VALUE 0x00000000

```

File - Save - Pin Selections

The File - Save - Pin Selections menu item is used to save your work. The saved file can be read in using the File - Load - Pin File menu item. This file contains a list of all currently selected signals. saves the list of the selected signals and #define constants. Save your work before exiting the program, changing the device selector radio buttons or clicking the Clear All button.

File - Load - Pin File

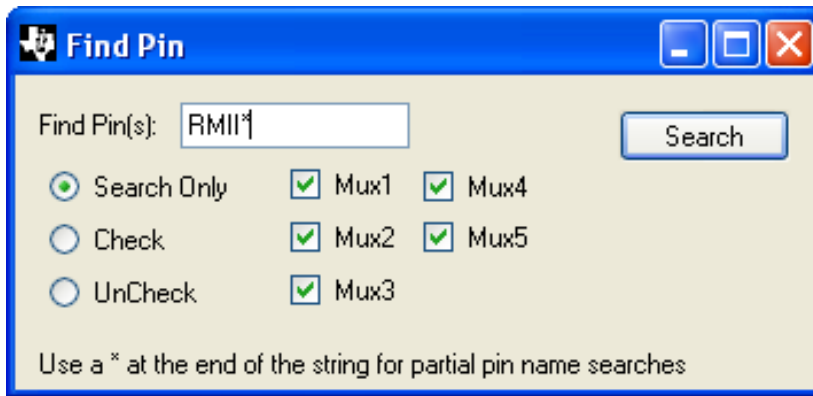
The File - Load - Pin File loads a file that was saved using the File - Save Pin Selections menu item.

File - Exit

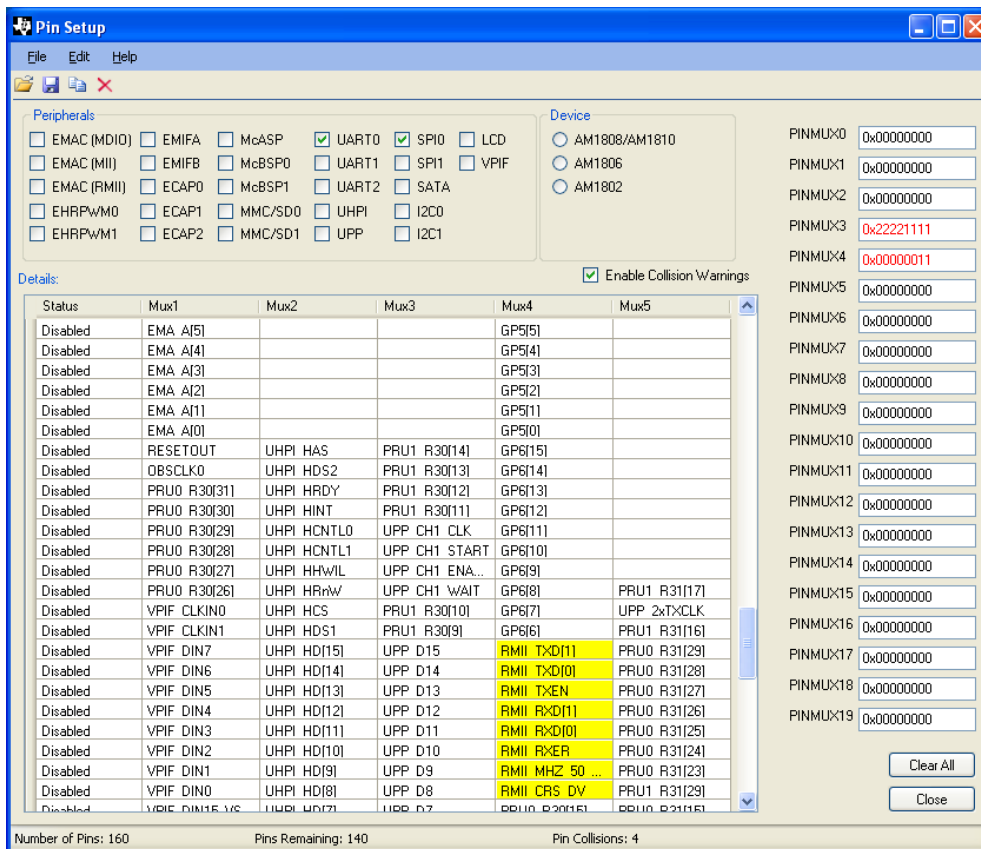
The File - Exit menu item closes the program. Program can also be closed by double-clicking the title bar icon at in upper left corner or by clicking the X icon. Save work using the File - Save - Pin Selections before exiting the program.

Edit - Find

The Edit - Find menu item open the Find Pin dialog. A signal name is entered. A asterisk "*" may be used as a wild card character. The is the option to search for the matching signal(s) or to select or unselect the matching signals.



Below is the result of searching for the RMII* signals.



Help - About

The Help - About menu item displays the program version information.

References

[1] <http://www-s.ti.com/sc/techlit/spraba2.zip>

AM335x Flashing Tools Guide



AM335x CCS Flashing Tools Guide

Linux PSP

NOTE

The pre-built flashing tools binaries provided with this release were tested with CCS v5.1. Please re-build if you are using a different CCS version.

About this manual

This document describes steps to build the different flashing tools and procedure to use them on AM335x platform using CCS v5.1. In case if you are using the pre-built binaries, you can skip the portion of Build Flashing Tool binaries and directly jump to section Steps For Flashing

Build Flashing Tool binaries

Requirements

Windows machine with CCS v5.1 installed.

Building NAND Flash Writer

NOTE

Pre-built image is available as `host-tools/nand-flash-writer.out` of PSP release package ^[1]

- NAND flasher source should be available as part of AM335x release package ^[1] under `host-tools/src/nandflash-<version>.tar.gz`
- untar source zip
- Start CCS v 5.1 and import this project using following steps.
 1. Start CCS v 5.1 from windows start menu.
 2. Open `c/c++` projects view by `View > C/C++ Projects`.
 3. Open Import dialogue box from `File->Import`.
 4. Expand CCS option and select Existing CCS/CCE Eclipse Project and select Next.
 5. Click Browse button and select the NAND flasher folder.
 6. Click on checkbox which says NAND Flasher and click Finish.
- From projects menu click Build Project, `Project > Build Project`.
- The flashing tool out file should be available in `Nand_flasher/Debug`.

Steps For Flashing

System Requirements

CCS v 5.1 or above installed on Windows XP.

Connecting CCS to AM335x EVM.

NOTE

Target is connected using CCS v5.1.

- Connect the XDS560 emulator to the AM335x EVM.
- Start CCS v5.1 by navigating to 'Start' menu in Windows XP.
- Use target configuration file AM335x.ccxml. If there is a need to create a new configuration, then follow steps below.
 1. Create new Target Configurations pane by **File > New > Target configuration file**.
 2. Name it as AM335x.ccxml and click on Finish.
 3. In the new Target configuration select Basic pane and select connection type as **TI XDS560 Emulator** and device type as **AM335x** by checking the box.
 4. Save the target configuration, eg: AM335x.ccxml.
 5. Select **coretex A8** core in the advanced pane and update the initialization script with AM335x.gel file.
 6. Save the target configuration.
- Select **Debug** in CCS if not there already: View-> Debug
- Select View -> Target Configurations. Expand User defined till you reach **AM335x.ccxml**.
- Right click and click "Launch Selected Configuration".
- This should launch debug session.
- In Debug view, (scroll till the end) Select "**TI XDS560 Emulator_0/Cortex A8**" connection.
- Right click and select "Set Debug Scope" option. This will make remove all the cores except **Cortex A8** from the debug view.
- Right click on the **Cortex A8** core listed in Debug view and click on "**Connect Target**".
- Run the initialization script under scripts tab, **AM335x System initialization ==>> AM335x_EVM_Initialization**

Loading GEL File

NOTE

Flashing tool expects the GEL file to initialize the **DDR** and **on-chip RAM**. The latest GEL file can be found at http://processors.wiki.ti.com/index.php/OMAP_and_Sitara_CCS_support^[2]

Load the Gel file by selecting the GEL Files under Tools tab after connecting to target. After loading Gel File initialize the EVM using scripts under Scripts tab. To initialize EVM select the **AM335x_EVM_Initialization** under **AM335x System Initialization**

Burning NAND Flash (using CCS)

This section describes how to burn an image to the NAND flash on the EVM. The pre-built CCS tool (host-tools/nand-flash-writer.out) for NAND flash should be available as part of PSP release package ^[1].

NAND flashing tool used to flash SPL and U-boot. SPL should be flashed to first four blocks as in AM335x, RBL checks for SPL at 1st 4 Blocks. If the 1st block fails, RBL checks in 2nd blocks and so on till 4th Block. To support this feature, SPL needs to be flashed to 1st 4 blocks. So the offset for flashing SPL are 0x0, 0x20000, 0x40000 & 0x60000. For U-Boot flashing offset to 0x80000, on the 4th block. Flash layout of NAND is shown here

NOTE

Flashing utility run each time to flash each component. While Flashing, flashing utility taken care of erasing the required blocks in NAND flash device.

Due to heavy pin-muxing, NAND flash is only available on few AM335x EVM and in few profiles. The user has to make sure that the d profiles have been properly selected to enable NAND before flashing the image to NAND. On unsupported profiles NAND flash tool will fail to detect NAND.

NOTE

For more information about EVM Configuration & profile Descriptions, please refer to EVM reference manual ^[3].

Flashing image to NAND Flash

- Start CCS using steps mentioned in Connecting CCS to AM335x EVM.
- Load GEL file using steps mentioned in Loading GEL File .
- After the GEL file is loaded reset the board by running scripts from GEL File. go to scripts -> AM335x System Initialization and click "AM335x_EVM_Initialization".
- Ensure that the image (e.g. MLO) to be flashed is present in the Windows XP Machine.
- Load nand-flash-writer.out and run it. nand-flash-writer.out should be available in the release package under board-utilities\
- Select option for flashing.

```
Choose your operation
Enter 1 ---> To Flash an Image
Enter 2 ---> To ERASE the whole NAND
Enter 3 ---> To EXIT
```

Select option 1 when prompted. Select option 2 in case if you want to erase the whole NAND.

- Enter the image path to flash when prompted as shown below.

```
Enter image file path
```

Provide the complete path (e.g. C:\images\MLO)

- Enter the offset when prompted when prompted as shown below.

```
Enter offset (in hex):
```

This offset is the start location from where the image should be flashed.

NOTE

Use hex format (For example, Enter 0x0 for flashing MLO image (i.e. 1st Stage)) in first block. 0x80000 for U-Bootto flash it in 4th block.

- Select ECC for flashing.

```
Choose the ECC scheme from given options
Enter 1 ---> BCH 8 bit
```



```
Enter 2 ---> HAM
Enter 3 ---> T0 EXIT
Please enter ECC scheme type:
```

Always select BCH8 for MLO and U-Boot as the ROM code uses the BCH8 ECC scheme. **Enter 1** for selecting BCH8.

- Ensure that the flash info displayed by the tool matches the NAND flash in the EVM.
- After this the tool should first erase the required region in flash and then start flashing the new image.
- Finally you should see the following message.

```
Application is successfully flashed
NAND boot preparation was successful!
```

- Disconnect CCS and then turn off the board.
- Make sure that Profile is selected for the NAND boot mode.
- Boot the Board & verify

References

- [1] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/psp/LinuxPSP/AM335x_04_06/index.html
- [2] http://processors.wiki.ti.com/index.php/OMAP_and_Sitara_CCS_support
- [3] <http://www.ti.com/tool/tmdxevm3358>

Flash v1.6 User Guide



Applicability

This tool can only be used on Sitara AM35x, AM37x devices.

This article applies to Sitara SDK 5.0.3.x. For SDK 5.0.2 see archived versions section.

1. This Windows XP-based Flash Tool can be used to program MLO, u-boot and kernel to NAND flash.
2. As an alternate method, you can use U-Boot to program MLO, u-boot and kernel to NAND flash. See [How_To_Program_NAND_Flash_Using_U-Boot](#) ^[1].
3. File systems must be programmed using Linux kernel commands.

Installation Instructions

1. Download "FlashTool for AM35x, AM37x, DM37x and OMAP35x Devices" from here: [Flash_v1.6.0.0](#) ^[2] onto a PC running the Windows XP O/S.
2. Open the downloaded FlashTool_vX.X.X.X zip file and run setup.exe to start the Windows Installer.
3. Follow the installer instructions.
4. The default installation directory is "C:\Program Files\Texas Instruments\Flash v1.6", but this can be changed when running the installer.
5. The FlashTool software can be uninstalled at any time by accessing the Add/Remove Programs function of Windows (Start -> Control Panel -> Add/Remove Programs).

About Flash v1.6

This page contains a description of a tool – Flash v1.6 – that can be used to transfer binary images from a host PC to certain TI ARM-based target platforms. The tool consists of two main components:

- a GUI host application, called Flash v1.6
- a CLI host application, called OMAPFlash.

It is recommended to use the GUI for performing flashing functions. This documentation covers usage of the GUI. Future documentation will cover the CLI interface in more detail. This application has been designed with flexibility and portability in mind. It is now possible to modify target register configurations without rebuilding the tool. This allows for easy modifications to various target peripheral configurations (such as SDRC, GPMC, Pad Control, etc.). This capability makes it possible to support new DDR devices and NAND devices without software changes. Check out the section [#Porting_Guide](#) for more information on this feature. Internally, the tool makes use of a ROM code mechanism for peripheral boot from UART or USB to transfer compatible drivers to the internal memory of the OMAP device. These drivers provide the mechanism by which the host applications can program binary images into the internal memories (NAND and SDRAM) of the target. All of this operation is hidden from the user.

Release Notes

Version 1.6.0.0 (6/24/2011)

New Features

- Added GUI controls for Target OS and 1-Bit ECC Layout.
- Set offset to ECC parity data in spare area based on new GUI controls.
- Added support for ECC offsets used with WinCE for 4/8-bit BCH.
- Eliminated requirement that image size be a mult of 4 bytes.
- Added software NAND unlock command.

Limitations

- Download and execute can only branch to Thumb code

Version 1.5.1.0 (3/25/2011)

New Features

- Programming flash via UART target connection is now supported.

Limitations

- Download and execute can only branch to Thumb code

Version 1.5 (3/13/2011)**New Features**

- Added support for programming NAND devices with Internal ECC

Limitations

- Programming flash via UART target connection is not supported
- Otherwise, similar functionality to previous releases.

Version 1.4 (2/16/2011)**New Features**

- Added AM35xx and Custom_AM35xx target types
- Updated "What's This?" help text for UART and USB target connection types
- Greyed out 4-bit BCH error correction for OMAP35xx devices (not supported by silicon)

Limitations

On-chip ECC is not supported

- Programming flash via UART target connection is not supported
- Otherwise, similar functionality to previous releases.

Version 1.3 (12/23/2010)**New Features**

- GUI updates, including "What's This?" style help, link to user's guide.
- Defined Custom target types for AM37xx and OMAP35xx, to allow customers to more easily make changes for custom boards.
- Enhanced NAND capabilities: SW vs. HW ECC Selection, 1b/4b/8b Error Correction, ONFI NAND selection via check box.

Limitations

- Still no AM35xx support (coming in future release).
- Otherwise, similar functionality to previous releases.

Version 1.2:**New Features**

- USB Support
- Can install over previous versions without manual uninstall
- Selectable NAND vs. ONFI NAND mode

Limitations

- Same as before

Version 1.1:**New Features**

- OMAP35xx Support.

Limitations

- Same as before

Version 1.0:**Features**

- UART support
-

- AM37xx/DM37xx support
- Can support new NAND devices via text config file modification.
- Can modify target registers via text config file modification (for example SDRC, GPMC, pad config settings)
- ONFI NAND Support
- Supported operations: Download, Download and Execute (to Thumb mode code), Erase Region, Erase All.
- Windows GUI (no Linux yet).
- Scriptable Windows CLI (no Linux yet).
- Fully open source, BSD-style license.

Limitations

- Download and execute can only branch to Thumb code.
- Storing images to NAND uses HWECC. Therefore, you cannot use a standard xloader to load uboot from NAND. Xloader uses SWECC when reading from NAND. The workaround is to use the uboot provided on the SD card to flash itself into NAND memory.
- GUI support does not yet exist to define new platforms. However, two memory choices are supplied, Hynix and Micron. For experimenting with modifications to the configuration text file (for instance, to port to a new platform) it is recommended that you modify the Micron files (and save a copy if you need the originals).

Accessing Online Help

There are two methods to get help from the user interface:

1. Click the "What's This Do?" Button. This changes the cursor type. Move the cursor over a widget until you see the cursor change to a question mark. Then, you may click and get context sensitive help.
2. Click the "Open User's Guide" Button. This will open your normal HTML browser to the Flash User's Guide Page (this page).

Setup for USB Peripheral Boot Mode

1. Ensure that your EVM is set up for peripheral boot from USB.

For Mistral OMAP EVM (AM37x, DM37x or OMAP35xx devices)

- Set SW4 switches #2 and #3 to the ON position and all others OFF.
- Connect a USB cable from the board's USB "On the Go" port to a USB port on the PC.
- Power on the EVM or press the reset pushbutton (labelled "OMAP_RST")

For LogicPD AM3517EVM:

- Set SW7 switches #1 and #4 to the ON position and all others OFF.
 - Connect a USB cable from the board's USB "On the Go" port to a USB port on the PC.
 - Power on the EVM
2. If this is the first time to connect the EVM to your PC via USB you will need to perform the following steps to install the required USB driver:
 - You will see the "Found New Hardware Wizard" dialog box. Select "No, not this time",

then "Next".

- Select "Install from a list or specific location (Advanced)".
- Use the browse box to select the following folder: <Flash-install-dir>\usb_drv_windows.

Normally, the full path will be:

C:\Program Files\Texas Instruments\Firmware vX.X\usb_drv_windows.

- Make sure the box "search removable media" is unchecked.
- Make sure the box "Include this location in the search" is checked.
- Click Next. The driver should install correctly without error messages.
- Click Finish to exit the Found New Hardware Wizard.

Setup for UART Peripheral Boot Mode

1. Ensure that your EVM is set up for peripheral boot from UART.

For Mistral OMAP EVM (Rev G):

- Set SW4 switches #2 and #4 to the ON position and all others OFF.
- Connect a 9-pin serial NULL-MODEM cable, from your host PC to the "UART3" port on the target board.
- Power on the EVM or press the reset pushbutton (labelled "OMAP_RST")

For LogicPD AM3517EVM:

- Set SW7 switches #1 and #4 to the ON position and all others OFF.
- Connect a 9-pin serial NULL-MODEM cable, from your host PC to the "UART" port on the target board.
- Power on the EVM.

Making Binary Images Accessible to the Windows XP O/S

The binary images for XLOADER, U-Boot and the Linux Kernel must be made accessible to the Windows XP O/S. There are a few methods for doing this:

1) If you are running your Linux Host as a virtual machine (using VMWare or using Sun Virtual Box) you can setup a shared folder that can be accessed by both the Linux Host and the Windows O/S. (You can also create sub-folders below that as needed - the resulting directory tree will be accessible to both Linux and Windows.)

See How to Set up a Shared Folder in VMWare^[3] or How to Setup a Shared Folder in Virtual Box^[4] for details.

OR

2) You can setup a Samba server that makes a designated location in your Linux Host file system appear as a drive to the Windows O/S.

The binary images can be copied to the shared folder or be accessed directly via a Samba drive.

1) Pre-built images can be found in the ./psp/prebuilt-images subdirectory of the Sitara SDK installation.

OR

2) Rebuilt images would be found where they were rebuilt under the ./psp subdirectory of the Sitara SDK installation.

Starting the Flash Application

To start the Flash application under Windows XP:

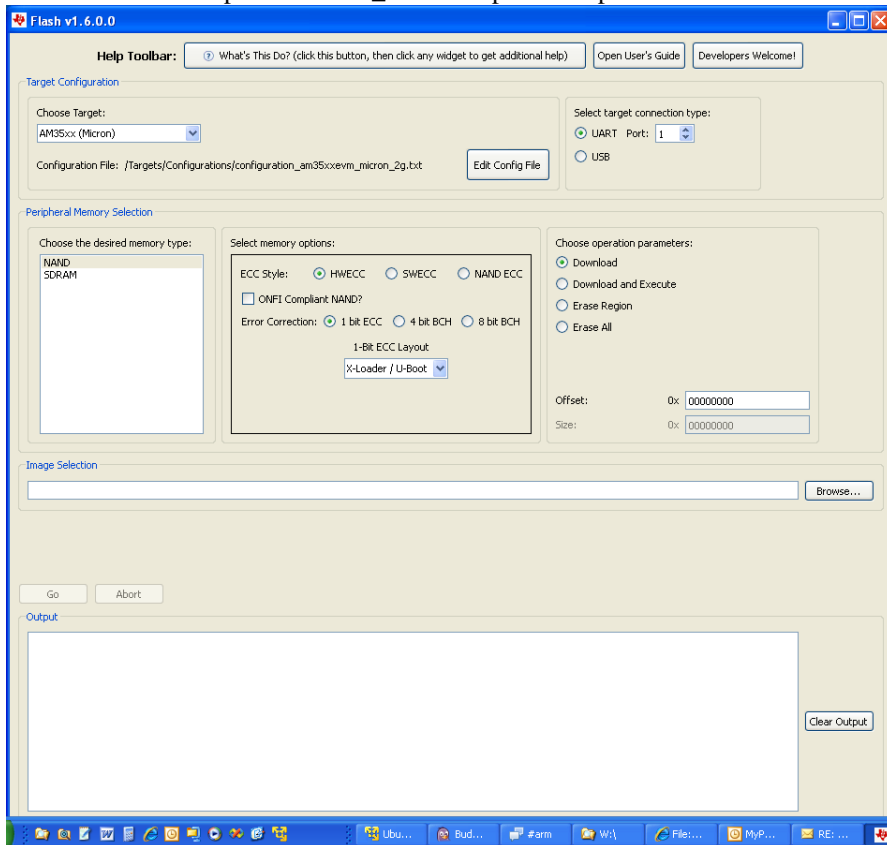
- Double click the "Shortcut to Flash.exe" icon on the desktop

OR

- Use the Windows Start menu:

"Start -> All Programs -> Texas Instruments -> Flash v1.6 -> Flash.exe".

Below is a screen capture of Flash_v1.6.0.0 upon startup.



Details about the Flash GUI

1. CHOOSE TARGET (drop-down box)

This allows selection of pre-defined target configuration files for the different EVM boards. The target configuration files contain register initialization code that mimics what is performed by XLOADER to initialize clocks and memory interfaces prior to flash programming. The following pre-defined EVM configurations and custom configurations are provided. Choose a custom target configuration and click the "Edit Config File" button to view or modify the target configuration for your custom design. See the Porting Guide ^[5] section for more information on how to design a custom target configuration file.

Pre-Defined EVM Target Board Configurations

Target Board	Target Configuration Description
AM35xx (Micron)	LogicPD AM3517EVM with AM3517A SOM and Micron discrete NAND and DDR2 memories
AM37xx (Hynix)	Mistral OMAP3 EVM with AM37x or DM37x SOM containing Hynix POP Memory
AM37xx (Micron)	Mistral OMAP3 EVM with AM37x or DM37x SOM containing Micron POP Memory
OMAP35xx (Micron)	Mistral OMAP3 EVM with OMAP35x SOM containing Micron POP Memory

Custom Target Board Configurations

Target Board	Configuration Description
Custom AM35xx Board	Same as AM35xx (Micron) - Modify for your custom AM35xx design.
Custom AM37xx Board	Same as AM37xx (Hynix) - Modify for your custom AM37x/DM37x design.
Custom OMAP35xx Board	Same as OMAP35xx (Micron) - Modify for your custom OMAP35xx design.

2. SELECT TARGET CONNECTION TYPE (Radio Buttons)

This allows selection of the target connection. Select USB or UART for communication between the Host PC and the target board (and also the peripheral boot mode that will be used).

For the UART connection, also select the PC COM port that will be used (typically 1 for COM1).

3. CHOOSE THE DESIRED MEMORY TYPE

This selects the destination memory type. Selecting NAND enables Erase and Download operations. Selecting SDRAM allows Download and Download and Execute operations.

4. SELECT MEMORY OPTIONS

If you have selected SDRAM, there are no other memory options to select. If you have selected NAND, choose from the following group of options:

- ECC STYLE (Radio Buttons)

This selects the method used to generate, check and correct ECC data in the NAND spare area. Choose HW ECC, SW ECC or NAND ECC. HW ECC uses the GPMC controller to calculate ECC parity and uses software error detection and correction. SW ECC uses software to calculate ECC parity and to detect and correct errors. NAND ECC uses Internal ECC controller on the NAND device to calculate ECC parity and to perform error detection and correction. Before NAND is programmed using NAND ECC mode, the Internal ECC of the NAND device is activated.

When programming XLOADER, always choose HW ECC and 1-bit ECC, as this is required by the ROM Boot Loader. When programming U-Boot or the Linux Kernel the ECC style must be set to be compatible with the software that will read the downloaded image from NAND.

- ONFI COMPLIANT NAND? (check box)

Check this box for operations on an ONFI-compliant NAND device. When checked, data specifying NAND memory geometry and features are read from the NAND device. Otherwise, this data is taken from the header in the Target Configuration file.

- ERROR CORRECTION (Radio Buttons)

Select the error correction algorithm that will be used to program the ECC data into the OOB area.

For SWECC ECC format, only 1-Bit ECC (Hamming Code) is allowed. For HWECC ECC format, 1-bit ECC, 4-bit BCH or 8-bit BCH algorithms can be selected. (4-bit BCH is not available for the OMAP35xx device due to silicon limitations). For NAND ECC mode, the ECC algorithm provided by the NAND Internal ECC is used.

- TARGET OS (drop-down box)

This GUI control shows only when HWECC 4-bit BCH or 8-bit BCH is selected. The ECC Offset in spare area is changed to 2 if the WinCE Target OS is selected.

- 1-Bit ECC Layout (drop-down-box)

This GUI control shows only when HWECC 1-bit ECC is selected. Choose "X-Loader/U-Boot" when programming XLOADER or U-Boot into NAND. For Linux kernel 2.6.37 or later, you must change this setting to Kernel when programming the Linux kernel into NAND. This changes the ECC offset in spare area from 2 to 40, as is needed to be compatible with the JFFS2 NAND file system. For earlier Linux kernels, "X-Loader/Uboot" setting should be used for programming the Linux kernel into NAND.

5. CHOOSE OPERATION PARAMETERS (Radio Buttons)

Select the operation to be performed:

- DOWNLOAD

Choose this to download a binary image to either NAND or SDRAM.

Before downloading an image to NAND, you must first erase the region where the image will be downloaded (or Erase All NAND).

Set OFFSET (hexadecimal value) to the offset from the beginning of NAND or SDRAM where the image will be downloaded.

- DOWNLOAD AND EXECUTE

Choose this to download a binary image and execute it. Applies to SDRAM as destination only.

Supports Thumb Mode ARM code only.

Set OFFSET (hexadecimal value) to the offset from the beginning SDRAM where the image will be downloaded.

- ERASE REGION

Choose this to erase a region in NAND flash.

Set OFFSET (hexadecimal value) to offset from beginning of NAND device where erasing begins.

Set SIZE (hexadecimal value) to the number of bytes which will be erased. Use a multiple of the NAND BLOCK size.

- ERASE ALL

Choose this to erase the entire NAND flash.

6. IMAGE SELECTION (Edit Box)

For download operations, you must specify the binary image to be downloaded. Click the BROWSE button to select an XLOADER, U-Boot, Linux Kernel. File systems cannot be programmed to NAND flash using this Flash Tool. U-Boot should be used to program file systems to NAND flash.

7. GO (Button)

Once you have successfully made your selections, the GO button will be enabled. Click the GO start

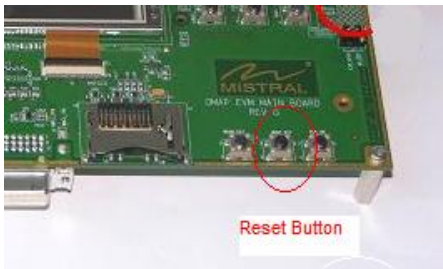
execution of the specified operation.

8. ABORT (Button)

Once the operation is in progress, you will have the option of clicking the ABORT button to cancel it. This would be used if communication with the target device fails or if the target board fails to complete the peripheral boot process.

9. OUTPUT (Text Window)

While the operation is in progress, status information is output to the OUTPUT text window. First a message will be displayed to "Please turn off device" (if powered on) and then "Please turn on device". At this point you can cycle power to the target EVM or (if already powered on) just press the RESET button on the EVM (For Mistral OMAP3 EVM, use the button labelled "OMAP_RST", as shown below.)



The Flash Tool will then automatically perform the following operations, as related status information is displayed in the OUTPUT text window:

- Read the ASIC ID from the target device
- Download the "2nd Loader" program (for communication with Flash Tool)
- Wait for "2nd Loader" program to start executing on the target board
- Perform register initializations as specified in the Target Configuration file.
- Download a software driver for the specific destination memory device
- Perform the specified Erase or Download operation

If any errors occur, related information will be displayed in the OUTPUT text window. Please cut and paste the information from the OUTPUT text window if support is needed to diagnose the problem.

10. CLEAR OUTPUT (Button)

At any time, you may clear the OUTPUT text window by clicking the CLEAR OUTPUT button.

Example 1: Flashing the AM/DM37x Evaluation Module

This example is compatible with the Sitara SDK 5.0.3.x and its u-boot-2011.09-bsp04.06.00.03.

and PSP software 04.02.00.07 which is based on the 2.6.37 Linux kernel. Flash Tool settings are shown for erasing and programming MLO, u-boot and the Linux kernel to NAND flash. It is assumed here that the image files shown were copied from the AM37x Sitara SDK from the "<SDK-Install-Dir>/board-support/preloaded-images"

folder of the Linux Host. For the AM37x Sitara SDK v5.0.3, the pre-built MLO, u-boot and kernel images are configured to use the HWECC 1-Bit ECC algorithm. (MLO must always be programmed into NAND using HWECC 1-Bit ECC as required by the ROM boot loader.)

As an alternative, Erase All could be used instead of the three Erase Region operations below. However, this would also erase the u-boot parameters partition in NAND. For the erase operations, the ECC STYLE and ERROR CORRECTION settings are don't cares. They can be left in the default state:

HWECC and 1-bit ECC.

For all of the operations listed below, also use these settings:

CHOOSE TARGET: AM37xx (Hynix)

SELECT TARGET CONNECTION TYPE: USB

CHOOSE THE DESIRED MEMORY TYPE: NAND

Perform NAND ERASE and NAND DOWNLOAD operations using the parameters shown here:

OPERATION	MEMORY OPTIONS	OFFSET	SIZE	IMAGE FILE
Erase Region for MLO	NON-ONFI	0	80000	
Download MLO	NON-ONFI,HWECC,1-Bit ECC, X-loader/u-boot ECC Layout	0	N/A	MLO-am37x-evm
Erase Region for u-boot	NON-ONFI	80000	80000	
Download u-boot	NON-ONFI,HWECC,1-Bit ECC, X-Loader/u-boot ECC Layout	80000	N/A	u-boot-am37x-evm.img
Erase Region for Linux kernel	NON-ONFI	280000	400000	
Download Linux kernel	NON-ONFI,HWECC,1-Bit ECC, Kernel ECC Layout	280000	N/A	uImage-am37x-evm.bin

The u-boot-2011.09-psp04.06.00.03 is a mainline SPL/u-boot. MLO and u-boot are built from the same sources. The ability to rebuild the MLO

to load u-boot using 4-bit BCH and 8-bit BCH is coming soon.

Example 2: Flashing the AM3517A Evaluation Module

This example is compatible with the Sitara SDK 5.0.3.x and its u-boot-2011.09-psp04.06.00.03.

and PSP software 04.02.00.07 which is based on the 2.6.37 Linux kernel. Flash Tool settings are shown

for erasing and programming MLO, u-boot and the Linux kernel to NAND flash. It is assumed here that the

image files shown were copied from the AM3517 Sitara SDK from the "`<SDK-Install-Dir>/board-support/preloaded-images`"

folder of the Linux Host. For the AM3517 Sitara SDK v5.0.3, the pre-built MLO, u-boot and kernel images are configured to use the HWECC 1-Bit ECC algorithm. (MLO must always be programmed into NAND using HWECC 1-Bit ECC as required by the ROM boot loader.)

As an alternative, Erase All could be used instead of the three Erase Region operations below.

However this would also erase the U-Boot parameters partition in NAND. For the erase

operations, the ECC STYLE and ERROR CORRECTION settings are don't cares. They can be left in the default state: HWECC and 1-bit ECC.

For all of the operations listed below, also use these settings:

CHOOSE TARGET: AM35xx (Micron)

SELECT TARGET CONNECTION TYPE: USB

CHOOSE THE DESIRED MEMORY TYPE: NAND

Perform NAND ERASE and NAND DOWNLOAD operations using the parameters shown here:

OPERATION	MEMORY OPTIONS	OFFSET	SIZE	IMAGE FILE
Erase Region for MLO	ONFI	0	80000	
Download MLO	ONFI,HWECC,1-Bit ECC, X-loader/U-Boot ECC Layout	0	N/A	MLO-am3517-evm
Erase Region for u-boot	ONFI	80000	80000	
Download u-boot	ONFI,HWECC,1-Bit ECC, X-Loader/U-Boot ECC Layout	80000	N/A	u-boot-am3517-evm.img
Erase Region for Linux kernel	ONFI	280000	400000	
Download Linux kernel	ONFI,HWECC,1-Bit ECC, Kernel ECC Layout	280000	N/A	uImage-am3517-evm.bin

The u-boot-2011.09-psp04.06.00.03 is a mainline SPL/u-boot. MLO and u-boot are built from the same sources. The ability to rebuild the MLO

to load u-boot using 4-bit BCH and 8-bit BCH is coming soon.

Rebuilding to Change from the Default ECC Scheme

For Sitara SDK 5.0.3.x this feature is coming soon.

After Programming Flash: Set EVM for NAND Boot Mode

After flash programming has been completed, you need to change the EVM to NAND boot mode

to execute the x-loader and u-boot binaries at boot up from NAND flash.

For Mistral OMAP EVM (AM37x, DM37x or OMAP35xx devices)

- Set SW4 switches #2 and #4 to the ON position and all others OFF. (UART3, NAND boot order)
- OR set SW4 switches #2, #4 and #6 to the ON position and all others OFF. (NAND, UART3 boot order)
- Power cycle the EVM

For LogicPD AM3517EVM:

- Set SW7 switches ALL to the OFF position. (NAND, EMAC, USB, MMC1 boot order)
- Power cycle the EVM

Flash v1.6 (Information for Developers)

- Source Code SVN repository is located on GForge in project Flash (<https://gforge.ti.com/gf/project/flash> ^[6])
- You can jump directly to this page from the Flash GUI by clicking the "Developers Welcome!" button.

Feedback

Several options exist for feedback:

- Leave comments using the link at the bottom of the page. For any error conditions, please copy and paste the text in the output box into the body of the email. Also, identify your target board, processor, and memory types. Provide a synopsis of what you are trying to do.
- Join the mailing list (arm_mpu_flash_tool@list.ti.com)
- Join the development effort via the GForge page, and provide feedback there.

Porting Guide

[Remainder of document intended for advanced users]

This is a small porting guide for Flash. It contains an introduction to the modifications that may be necessary in order to get Flash working on a new board or with a new memory device.

Porting to a new platform

In general, porting to a new platform should not require the modification or recompilation of the code base. The starting point will be to create a board configuration file for the new platform.

Existing board configuration files are found in `.\Targets\Configurations`. In order to add support for a new platform it is recommended to modify `configuration_custom_am37xx.txt` (used when selecting target "Custom AM37XX Board") or `configuration_custom_omap35xx.txt` (used when selecting target "Custom OMAP35xx Board").

Once the board configuration file has been created, it will need to be added to the content of `omapflash2nd.txt`, present in the application installation folder. (NOTE: in the case of the prebuilt custom files, this is already done for you). This file allows Flash to find the board configuration file. When modifying the file, simply add a new line to it:

```
<platform> <omap id> <omap version> <omap type> <second loader file> -peripheralboot_reopen -board_config <board configuration file>
```

where `<platform>` is the platform name (use a new name for the new platform), `<omap id>` is an id number provided by the OMAP device during peripheral boot over UART or USB along with the `<omap version>` number, `<omap type>` specifies whether the OMAP detected should be a High Security 'HS' or General Purpose 'GP' type, `<second loader file>` specifies the second loader to use with the combination of `<platform>`, `<omap id>`, `<omap version>` and `<omap type>` and the `<board configuration file>` is the newly added board configuration.

An example could be:

```
SDP_MDDR_HYNIX_4G 363007 07 GP Targets\2nd-Downloaders\dnld_startup_omap3_gp_4g.2nd -periphboot_reopen
-board_config Targets\Configurations\configuration_sdp3630_hynix_4g.txt
```

Note that the installation currently comes with second loaders supporting memory sizes of 2 Gb, 4 Gb and 8 Gb for GP and HS devices:

```
dnld_startup_omap3_gp_2g.2nd - OMAP3 GP w 2 Gb SDRAM
dnld_startup_omap3_gp_4g.2nd - OMAP3 GP w 4 Gb SDRAM
dnld_startup_omap3_gp_8g.2nd - OMAP3 GP w 8 Gb SDRAM
dnld_startup_omap3_hs_2g.2nd - OMAP3 HS w 2 Gb SDRAM
dnld_startup_omap3_hs_4g.2nd - OMAP3 HS w 4 Gb SDRAM
dnld_startup_omap3_hs_8g.2nd - OMAP3 HS w 8 Gb SDRAM
dnld_startup_omap4_gp_2g.2nd - OMAP4 GP w 2 Gb SDRAM
dnld_startup_omap4_gp_4g.2nd - OMAP4 GP w 4 Gb SDRAM
dnld_startup_omap4_gp_8g.2nd - OMAP4 GP w 8 Gb SDRAM
dnld_startup_omap4_hs_2g.2nd - OMAP4 HS w 2 Gb SDRAM
dnld_startup_omap4_hs_4g.2nd - OMAP4 HS w 4 Gb SDRAM
dnld_startup_omap4_hs_8g.2nd - OMAP4 HS w 8 Gb SDRAM
```

The reason for this is that there is a link-time dependency on the placement of the heap and external memory for the memory device drivers in SDRAM and on the location of the second loader components in internal memory between HS and GP devices. Pick the right one - e.g. if you have a 4 Gbit memory on an OMAP3 GP based board, use `'dnld_startup_omap3_gp_4g.2nd'`.

Board configuration

In order to create the new file it is often useful to start with a copy of one of the existing files. The file has three main sections:

1. 'use' directive, pointing to a definition file listing a number of OMAP registers and their addresses
2. 'memory' directives, specifying the memories on the platform
3. initialization commands, modifying registers to control the configuration of the OMAP to match the platform

Definitions

A 'use' directive can be used to point to a device definition file, e.g.:

```
use definitions_omap3.txt
```

Only one definition file can be used and it must be indicated before the first element using its definitions occurs in the configuration file. The device definition file basically holds a set of paired of labels and values. The labels can be used in the device configuration file in place of the values in order to make the device configuration file more readable. The syntax is as follows:

```
PRM_CLKSRC_CTRL          0x48307270
CM_CLKEN_PLL             0x48004D00
PRM_CLKSEL               0x48306D40
CM_CLKSEL1_PLL           0x48004D40
CM_CLKSEL2_PLL           0x48004D44
CM_CLKSEL3_PLL           0x48004D48
```

A maximum of 1000 definition pairs can be present in a definition file.

Memories

Memories are specified using the 'memory' directive:

```
memory NAME [driver DRIVER] [parameters PARAMETER1 VALUE1 PARAMETER2 VALUE2 ... PARAMETERN VALUEN]
```

An example of a memory specification could be:

```
memory NAND driver Targets\Flash-Drivers\nand_onfi_16bit_8bit.bin parameters gpmc 0x6E000000 cs 1 address 0x28000000 bberase 0
```

where the device name is NAND and the driver required to access it is present in the binary file `nand_onfi_16bit_8bit.bin` (part of this distribution). The driver needs a number of configuration parameters for correct operation. These are passed directly to driver as written on the line following the 'parameters' keyword. This distribution contains a number of driver binaries for various memory types. At present these are:

```
File      : nand_onfi_16bit_8bit.bin
Type      : NAND

Parameters: gpmc      (mandatory)      Base address of the GPMC in the OMAP
           cs         (mandatory)      Chip select where the device is present (or GPMC-config index)
           address    (mandatory)      Address of the device as mapped in the GPMC
           bberase    (mandatory)      Erase bad blocks in the device (0 for no, 1 for yes).
                                           Caution: erasing bad blocks may cause an irreversible loss of manufacturing information.
           onfi       (optional)       Read and use ONFI device description from the device (0 for no, 1 for yes).
           bpp        (mandatory if onfi = 0) Bytes per page if ONFI information is not used.
           sbpp       (mandatory if onfi = 0) Spare bytes per page if ONFI information is not used.
           ppb        (mandatory if onfi = 0) Pages per block if ONFI information is not used
```

```

bpl      (mandatory if onfi = 0)  Blocks per logical unit if ONFI information is not used

l        (mandatory if onfi = 0)  Logical unit count (only 1 supported by the driver)

acv      (mandatory if onfi = 0)  Address cycle values - 8 bit value with lower 4 bits for row and upper 4 bits for
                                     column.

f        (mandatory if onfi = 0)  Features - 16 bit value with bit 0 = 16 bit data operation, rest are don't-care

```

```

Examples : memory NAND driver Targets\Flash-Drivers\nand_onfi_16bit_8bit.bin parameters gpmc 0x6E000000 cs 1 address 0x28000000
          bberase 0

```

```

memory NAND driver Targets\Flash-Drivers\nand_onfi_16bit_8bit.bin parameters gpmc 0x6E000000 cs 1 address 0x28000000
          bberase 0 onfi 0 bpp 2048 sbpp 64 ppb 64 bpl 4096 l 1 acv 0x23 f 0x0019

```

For SDRAM, no driver is required, but the memory type must be specified with one parameter stating the base address in the memory map, e.g.:

```
memory SDRAM parameters address 0x80000000
```

Initialization of Target Device

A number of register operation commands can be used to configure the target device:

- WRITE - Write a value to a register
- MODIFY - Modify the value of a register
- POLL_ZERO - Poll a register value until zero
- POLL_NZERO - Poll a register value until not zero
- POLL_VALUE - Poll a register until value
- WAIT_N - Loop n times in a simple while-loop (where n is a value from 0x0000 to 0xFFFF)
- SPIN - Loop forever. This may be used for debugging.
- MODE_16 - Use 16 bit register access mode
- MODE_32 - Use 32 bit register access mode

The command structures are:

- WRITE : WRITE REGISTER VALUE
- MODIFY : MODIFY REGISTER MASK VALUE
- POLL_ZERO : POLL_ZERO REGISTER MASK
- POLL_NZERO : POLL_NZERO REGISTER MASK
- POLL_VALUE : POLL_VAL REGISTER MASK VALUE
- WAIT_N : WAIT_N N
- SPIN : SPIN
- MODE_16 : MODE_16
- MODE_32 : MODE_32

The definitions included from a definition file specified in a 'use' directive can be used with the commands. Definitions can be used for registers, values and masks, e.g.:

MODIFY	CM_CLKEN_PLL_MPU	EN_XXX_DPLL_MODE_MASK	EN_XXX_DPLL_LOCK_MODE
WRITE	CM_CLKSEL3_PLL	0x00000009	
POLL_ZERO	CM_IDLEST_CKGEN	ST_PERIPH_CLK_DPLL4_LOCKED	

Archived Versions

AM35x/AM37x Flash Tool User's Guide for Sitara SDK 5.0.2 (Archived) ^[7]

References

- [1] http://processors.wiki.ti.com/index.php/AM35x-OMAP35x-PSP_04.02.00.07_UserGuide#NAND
- [2] http://software-dl.ti.com/dsps/dsps_public_sw/am_bu/amflashutil/latest/index_FDS.html
- [3] http://processors.wiki.ti.com/index.php/How_to_Build_a_Ubuntu_Linux_host_under_VMware#How_to_Set_up_Shared_Folder_in_VMWare
- [4] http://processors.wiki.ti.com/index.php/How_to_Build_a_Ubuntu_Linux_host_under_VirtualBox#Sharing_Files_Between_Ubuntu_and_Windows
- [5] http://processors.wiki.ti.com/index.php/Flash_v1.4_User_Guide#Porting%20Guide
- [6] <https://gforge.ti.com/gf/project/flash>
- [7] http://processors.wiki.ti.com/index.php?title=Flash_v1.6_User_Guide&oldid=89405

AM18x Flash Tool User's Guide

Return to the [Sitara Linux Software Developer's Guide](#)



The following details the procedures required to Flash the AM18x EVM.

USB-based Flash Tool support is not planned for ARM9 based processors.

1. Download serial flashing tool here : <http://sourceforge.net/projects/dvflashutils/files/OMAP-L138/> ^[1]. Get the latest version, 2.35.

Version 2.35 is newer than the version included with the Sitara SDK 5.0.3.x. Version 2.35 has CCS v5.1 projects for the

CCS/JTAG based flash writer tools.

2. Extract the serial flashing tool onto your Windows machine. It creates a file structure that begins with the folder **OMAP-L138_FlashAndBootUtils_2_35**.

3. The OMAP-L138_FlashAndBootUtils program requires Microsoft .NET framework v4.0 or later. To see if it is necessary to update your Microsoft .NET Framework

to v4.0

see How to determine Microsoft .NET Framework version ^[2]. If v4.0 has NOT been installed, download it from here: Microsoft .NET 4.0 framework ^[3] and install it.

Reboot machine if it asks you.

- **The OMAP Serial Flash Tool has had some issues running on a vmware and virtual box environment, if you are using these environments it is recommended to use the Windows version of the tool.**

4. Copy your u-boot binary file into the OMAP-L138_FlashAndBootUtils_2_35\OMAP-L138\GNU folder.

The pre-built UBL and u-boot binary file can be found in the `./board-support/prebuilt-images/` folder in the SDK installation.

- **Note: The UBL binary found in the prebuilts section of the SDK is will run the processor at 456MHz. If your part will not run at that speed please use the UBL binary that ships with the Serial OMAP Flash Tool. This UBL will expect to find U-boot at a pre-defined sector on the SD Card. Please see**

this article on how to write u-boot for a bootable SD Card on AM180x processors. ^[4]

See How to Set up a Shared Folder in VMWare ^[3] or How to Setup a Shared Folder in Virtual Box ^[4] for help with moving files between a Linux virtual machine and the Windows OS.

5. Open a command prompt window and navigate to OMAP-L138_FlashAndBootUtils_2_35\OMAP-L138\GNU for a Windows platform and reversing the slashes for a Linux platform OMAP-L138_FlashAndBootUtils_2_35/OMAP-L138/GNU

6. Set the boot pins to UART2 boot mode. This is done by setting switch S7 on the AM18x EVM according to the following table:

- **Note: As a reminder please remember your existing switch settings before changing so you can switch back, this affects boot modes etc.**

AM180x EVM (Logic)

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	OFF	OFF	ON	ON

AM1810 EVM (Spectrum Digital)

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	OFF	ON	OFF	ON

7. Flash uboot using one of the following commands: (Please note, if you are using Windows command prompt, do not copy and past the command, you must type it at the command prompt)

For AM180x EVM (Logic): Execute the command “sfh_OMAP-L138.exe –flash ublubl_OMAPL138_SPI_MEM.bin u-boot.bin”.

For AM1810 EVM (Spectrum Digital): Execute the command “sfh_OMAP-L138.exe –targetType AM1810 –flash ublubl_INTDEV0_SPI_MEM.bin u-boot.bin”.

8. Press S5 to reset the target board when the serial flasher program requests it.

9. Set the boot pins to SPI1 boot mode. This is done by setting switch S7 on the AM18x EVM according to the following table:

AM180x EVM (Logic)

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

AM1810 EVM (Spectrum Digital)

Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	OFF	ON	ON	OFF

10. Reset the platform to obtain the uboot prompt

References

- [1] <http://sourceforge.net/projects/dvflashutils/files/OMAP-L138/>
- [2] <http://msdn.microsoft.com/en-us/kb/kbarticle.aspx?id=318785>
- [3] <http://www.microsoft.com/downloads/en/details.aspx?FamilyID=9cfb2d51-5ff4-4491-b0e5-b386f32c0992&displaylang=en>
- [4] http://processors.wiki.ti.com/index.php/GSG:_OMAP-L138_DVEVM_Additional_Procedures#Creating_bootable_SD_card_for_OMAP-L138_EVM_board

AMSDK File System Optimization/Customization



Return to the [Sitara Linux Software Developer's Guide](#)

Purpose

The purpose of this article is to explain the file systems that are delivered as part of the AMSDK and how those file systems can be modified to customize them for your use case. The tools discussed in this article are installed by default in the AMSDK file system for your convenience.

Pre-built File Systems

The AMSDK ships with two default file systems. They are:

1. `base-rootfs-<machine>.tar.gz` - This file system is the simple file system that forms the base of the AMSDK file system. It has some basic utilities installed but is intended to be rather small and light weight.
2. `tisdk-rootfs-<machine>.tar.gz` - This file system is created by taking the base file system and adding all the additional SDK components such as 3D graphics, matrix, profiling tools, etc. It is a larger file system but is meant to have most of the tools developers would need when working with TI processors.

Both of these file systems contain the **opkg** package manager and can be used as a starting point for system optimization as discussed in the next sections.

The Basics of OPKG

opkg - list commands

opkg is the package manager used in the SDK file systems to install and remove packages. You can get a list of the full commands supported by **opkg** by running the following command on the target device:

```
target# opkg
```

opkg - list installed packages

To the list of packages are part of the file system and the version of each package you can run the following command on the target device:

```
target# opkg list-installed
```

NOTE: This list only contains information about packages that were installed using the package manager. Applications that were built and copied to the file system and not installed as a `.ipk` package are not tracked by **opkg**

opkg - list package contents

Sometimes it is useful to know what files a package has installed. To do this you can use the following command:

```
target# opkg files <pkgname>
```

Where <pkgname> is the name of the package as given in the *opkg list-installed* output. This command will produce a list of all the files on the file system that belong to the given package.

opkg - find a file owner

Sometimes you may find a file on the target file system that you want to remove, or just know where it came from. In this case you can use the following command:

```
target# opkg search <file>
```

This command will find which package installed the given file. This may be useful later when you want to remove an particular file because this command can help you find the package to remove.

opkg - finding dependencies

Sometimes when you want to remove a package it is useful to find out what other packages depend on the package you are removing. While the *opkg remove* command will tell you the immediate dependencies you can find the longer list of dependencies using:

```
target# opkg whatdepends <pkgname>
```

This command will print the list of packages the depend on the package you entered, as well as the packages that depend on those packages, and so forth.

Removing Packages

One of the simplest ways to modify the contents of the file system is to use the **opkg** utility to remove packages (or install if you have pre-built packages). Removing a package is often as simple as:

```
target# opkg remove <pkgname>
```

However, sometimes a package is a **DEPENDENCY** of another package. In this case you have the following options:

- Use the `--force-depends` option
 - This option will force the removal of the package but will leave any packages that depend on this package installed. This can be useful in the case that you want to remove a particular package, but that package is depended on by some other package that you do not want removed.
- Use the `--force-removal-of-dependent-packages` option
 - This option will go up the dependency list and remove all packages in the dependency chain. You should check all the packages being removed to make sure they are indeed packages you want to remove and do not contain other files you want.
- Remove the individual packages that depend on this package first
 - This way you can control exactly which dependent packages are removed and which are left in place.

```
target# opkg remove --force-removal-of-dependent-packages <pkgname>
```

NOTE: *opkg* will print the packages that depend on the package being removed. It is usually a good idea to use the *opkg files* command for the packages that depend on the one being removed so that you can make sure that no files

you really want to keep are going to be removed.

NOTE: You can also use the `whatdepends` option discussed above to see the dependency list for a package

By using the `opkg remove` command in conjunction with the commands for listing packages, finding file owners, and listing package contents, you can quickly strip down a full file system into something smaller and more optimized for your use case. However, once a package is removed it cannot currently be re-installed without generating the `.ipk` file to install it.

Adding Applications

In most cases installing additional applications can be as simple as copying the the binary executable to the file system. However, if you have built your own `.ipk` packages you can use `opkg` to install those `ipks` into the target file system. One major advantage of using the package manager is the ability to track the package and it's content with the `opkg` package manager. More details will be coming about how to build your own packages, but for now please refer to [this link](#) ^[1] to learn more about building custom file systems with Arago.

References

[1] http://arago-project.org/wiki/index.php/Building_with_Arago

Sitara Linux Training

Return to the [Sitara Linux Software Developer's Guide](#)



Training Modules

We have a number of modules that are complete and others we are actively working on. All of our Linux training is provided on this wiki.

If you have comment on the training or a request for Linux training that is not be captured here, please let us know using the `sdk_feedback@list.ti.com` ^[14] mailing list

NOTE

The presentations below are loaded to a Google Drive share site. To download the presentations please follow the link to the Google Drive site and then select **File -> Download** to save the presentation

NOTE

If you are unable to access Google Drive documents you can also find this training material at www.ti.com/sitarabootcamp

Lecture	Lab	Description
Linux Host Configuration	Sitara Linux Training: Linux Host Configuration	This page details how the Linux Host is configured for Sitara Linux Training. These are the same methods to prepare laptops used in live TI training.
Introduction to Linux	-	Introduces the community-based Linux ecosystem on TI platforms. What will be covered are the components that make up the ecosystem such as the boot loader, Linux kernel, device drivers, user application layer and the relationship between them.
Linux Boot Process ^[15]	-	Looks at all aspects of the boot process from power up to running user a application beginning with ROM boot loader progressing through secondary program loader, u-boot, kernel and finishing with user-level initialization.
Sitara Linux Training: Hands-on with the SDK ^[16]	Sitara Linux Training: Hands on with the SDK	Learn about the various components that make up the ARM MPU Linux software development kit including the out-of-box application launcher, the CCS IDE, example applications. In addition, host tools such as the pin-mux utility and the flash tool will be introduced. All these components are packaged into a single easy to use installer from TI.com
Code Composer Studio v5	Code Composer Studio v5	Covers what the Eclipse-based Code Composer Studio is, how to use it for embedded Linux application development, debugging and additional plug-ins that are provided
Sitara Linux Training: Power Management ^[17]	Sitara Linux Training: Power Management	Discusses how to improve product power performance by minimizing power consumption and guaranteeing system performance. In addition, power management techniques enabled via the Linux SDK will be discussed
Sitara Linux Training: Cryptography ^[18]	Sitara Linux Training: Cryptography	Covers cryptography basics and explore cryptographic functions enabled via open source projects. In addition, cryptographic hardware acceleration access and Linux SDK example applications will be discussed.
Sitara Linux Training: Linux Board Port ^[19]	Sitara Linux Training: Linux Board Port	Discusses the fundamentals necessary to port a TI Linux-based EVM platform to a custom target platform. Introduces the necessary steps needed to port the following components: Linux kernel.
Sitara Linux Training: U-Boot Board Port ^[19]	Sitara Linux Training: U-Boot Board Port	Discusses the fundamentals necessary to port a TI Linux-based EVM platform to a custom target platform. Introduces the necessary steps needed to port the following components: secondary program loader, u-boot.
Sitara Linux Training: U-Boot/Kernel Debug with CCSv5 ^[20]	Sitara Linux Training: U-Boot Linux Debug with CCSv5	Learn about how U-Boot and Kernel Debug can be done using CCSv5 using JTAG. This presentation and accompanying lab will discuss what debug information is necessary to be built into U-Boot and the Kernel to allow source code level debug with a JTAG interface.
ARM Multimedia	ARM Multimedia	Introduces open-source based multimedia codecs for the ARM Cortex-A8. In addition, look at the capability of the NEON coprocessor to accelerate multimedia. Plus, introduces GStreamer, an open-source pipeline-based framework that enables access for multimedia through FFMPEG/libav support on the ARM. GStreamer will be illustrated with Linux SDK examples.
Sitara Linux Training: Hands on with QT ^[21]	Sitara Linux Training: Hands on with QT	Learn how to develop a GUI quickly with the Linux SDK. Learn background information on QT. Learn how to use the SDK to get started developing a GUI. Learn about QT Creator and all the QT toolset.
Oprofile	Oprofile	Introduces the Opensource tool Oprofile. When is it useful during the development cycle. Introduce some of the more popular features. Cover both modes of operation, internal HW counters or timer interrupts. Cover internal operation details. Also point out use cases where Oprofile may not be useful.
Init Scripts ^[22]	Sitara Linux Training: Init Scripts	Learn how the Linux init scripts work with the sysvinit system as well as how the profile scripts can be used to affect the login process.
Optimizing Linux Boot Time ^[23]	Sitara Linux Training: Optimizing Linux Boot Time	Learn how to identify the portions of the Linux boot taking the most time and remove or defer those operations until later. The goal of this lab is to have a system booting to a display on the LCD and reading a touchscreen event in less than 3 seconds.

How to use a Mouse instead of the Touchscreen with Matrix



Return to the [Sitara Linux Software Developer's Guide](#)

Introduction

A mouse can be used instead of touchscreen input with the Matrix GUI by following these steps. It is also possible to use touchscreen and mouse input simultaneously.

Restrictions: For AM37x and AM35x: The mouse must be connected through a USB 2.0 hub to the EVM. It must not be connected directly to the EVM.

Enable Mouse

On your Target File system, edit the file: `/etc/init.d/matrix-gui-2.0` This script file will set Mouse support by default, but if it detects a touchscreen, then it will setup touchscreen support. If you comment out the touchscreen support then the default will remain mouse support. Use the '#' to comment out the line below:

For Mouse support, Replace:

```
if [ -e /dev/input/touchscreen0 ]
then
    export QWS_MOUSE_PROTO=Tslib:/dev/input/touchscreen0
```

with

```
if [ -e /dev/input/touchscreen0 ]
then
#    export QWS_MOUSE_PROTO=Tslib:/dev/input/touchscreen0
```

To Enable both Mouse and touchscreen simultaneously: Replace

```
export QWS_MOUSE_PROTO=Tslib:/dev/input/touchscreen0
```

with

```
export QWS_MOUSE_PROTO="Tslib:/dev/input/touchscreen0 Auto"
```

Warning: If you enable both, the touchscreen now requires a double touch instead of a single touch.

Kernel

The Kernel has been configured to enable mouse support in your SDK, so this step is not necessary if you are using the SDK.

If you have built your own kernel: The kernel must be configured to include the PS/2 mouse support. To verify this, run the menuconfig utility.

```
From you Linux root directory:  
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- menuconfig
```

Look under:

Device Drivers ---> Input device support ---> <*> Mouse interface

Device Drivers ---> Input device support ---> <*> Mice --->

If mouse support is not already included in the kernel, enable the above two configuration by typing 'y' and rebuild the kernel. You can find instructions for rebuilding kernel here:

For AM335x: AM335x PSP User's Guide#Compiling_Linux_Kernel

For AM35x, AM37x, Beagleboard: AM35x-OMAP35x-PSP 04.02.00.07 UserGuide#Compiling_Linux_Kernel

For AM180x: GSG:_Building_Software_Components_for_OMAP-L1#Rebuilding_the_Linux_kernel

How to Recalibrate the Touchscreen



Return to the [Sitara Linux Software Developer's Guide](#)

Where is the pointercal file?

In the latest Sitara SDK (versions 05.03.00.00 and greater) the location of the calibration file for the touchscreen has been relocated from the default location of /etc/pointercal to instead be placed at /media/mmcblk0p1/pointercal if the SD device is present. This was done to allow user's to easily delete this file and force a recalibration without requiring a Linux PC to mount the file system or a serial console to access the device. In this case the pointercal file is now written to a FAT partition of the SD card, which can be mounted on both Windows or Linux.

If the pointercal file is created at /media/mmcblk0p1/pointercal, and there is not already a file in /etc/pointercal then the file is also copied to the old default location. This is to prevent requiring the export of the TSLIB_CALIBFILE variable as described in later sections of this document.

In the case that there is no SD card available then the pointercal file will be written to the default /etc/pointercal location.

Recalibrating by Deleting pointercal

Because the pointercal file is now written to the FAT partition of the SD card recalibrating the touchscreen can be done by simply inserting the SD card into a PC and deleting the pointercal file found on the first partition (assuming the first partition is a FAT partition). When the system is then rebooted you will be asked to calibrate the touchscreen.

Recalibrating over Serial Console

If you do have access to the serial console you can always recalibrate the touchscreen using the `ts_calibrate` command as normal. However, in this case if you already have a pointercal file on the SD card you will need to export the `TSLIB_CALIBFILE` variable to point to that location before you run `ts_calibrate`. i.e.

```
target# export TSLIB_CALIBFILE=/media/mmcblk0p1/pointercal
target# ts_calibrate
```

NOTE: If you have Matrix or any other GUI application already running you should stop that application (How to Stop Matrix) before running `ts_calibrate` so that you can see the calibration screen.

Exporting TSLIB_CALIBFILE

The environment variable `TSLIB_CALIBFILE` is used by the `tslib` package to indicate the location of the pointercal file. The default value for `TSLIB_CALIBFILE` is `/etc/pointercal`. If you are writing a program that uses `tslib` you should make sure that you export `TSLIB_CALIBFILE` to point to the location of your pointercal file (as indicated in the previous section), or that you place a copy of the pointercal file at the default `/etc/pointercal` location.

AM335x U-Boot User's Guide



AM335x U-Boot User's Guide

Linux PSP

U-Boot

In AM335x the ROM code serves as the 1st stage bootloader. The 2nd and the 3rd stage bootloaders are based on U-Boot^[1]. *In the rest of this document when referring to the binaries, the binary for the 2nd stage is referred to as SPL and the binary for the 3rd stage as simply U-Boot.* SPL is a non-interactive loader and is a specially built version of U-Boot. It is built concurrently when building U-Boot.

The ROM code can load the SPL image from any of the following devices

- **Memory devices non XIP (NAND/SDMMC)**

The image should have the Image header. The image header is of length 8 byte which has the load address(Entry point) and the size of the image to be copied. RBL would copy the image, whose size is given by the length field in the image header, from the device and loads into the internal memory address specified in the load address field of Image header.

- **Peripheral devices (UART)**

RBL loads the image to the internal memory address 0x402f0400 and executes it. No Image Header present.

Two stage U-Boot design

This section gives an overview of the two stage U-Boot approach adopted for AM335X.

The size of the internal RAM in AM335X is 128KB out of which 18KB at the end is used by the ROM code. Also, 1 KB at the start (0x402f0000 - 0x402f0400) is secure and it cannot be accessed. This places a limit of 109KB on the size of the U-Boot binary which the ROM code can transfer to the internal RAM and use as an initial stack before initialization of DRAM.

Since it is not possible to squeeze in all the functionality that is normally expected from U-Boot in < 110KB (after setting aside some space for stack, heap etc) a two stage approach has been adopted. Initial stage initialize only the required boot devices (NAND, MMC, I2C etc); 2nd full stage initialize all other devices (ethernet, timers, clocks etc). The 1st binary is generated MLO and the 2nd stage is generated as u-boot.img.

NOTE

*When using memory boot (NAND) a header needs to be attached to the SPL binary indicating the load address and the size of the image. SPI boot additionally requires endian conversion before flashing the image.

- When using peripheral boot (UART) there can be no header as the load address is fixed.

Building U-Boot

Prerequisite

GNU toolchain for ARM processors from Arago is recommended. Arago Toolchain can be found in the linux-devkit directory of the SDK here ^[2]

NOTE

Below steps assumes that the release package is extracted inside directory represented as \$AM335x-PSP-DIR

Change to the base of the U-Boot directory.

```
$ cd ./AM335x-LINUX-PSP-MM.mm.pp.bb/src/u-boot/u-boot-MM.mm.pp.bb
```

Building into a separate object directory with the "O=" parameter to make is strongly recommended.

Commands

```
$ [ -d ./am335x ] && rm -rf ./am335x
$ make O=am335x CROSS_COMPILE=arm-arago-linux-gnueabi- ARCH=arm am335x_evm
```

This will generate two binaries in the am335x directory, MLO and u-boot.img along with other intermediate binaries that may be needed in some cases (see below).

Host configuration

Serial port configuration

Connect a serial cable from the serial port of the EVM (serial port is next to the power switch) to the COM port on either the Windows machine or Linux host depending on where you'll be running the serial terminal software.

For correct operation the serial terminal software should be configured with the following settings:

```
*Baud rate: 115,200
*Data bits: 8
*Parity: None
```



```
*Stop bits: 1
*Flow control: None
```

NOTE

If Teraterm is being used, ensure that the latest version (4.67 is the latest version as of writing this user guide) of Teraterm is installed. The implementation of the Kermit protocol in Teraterm is not reliable in older versions. The latest version of Teraterm 4.67 can be downloaded from here ^[3]. Recent Teraterm updates causes slow Binary transfer over UART. In such cases, use Windows in-built HyperTerminal application.

Target configuration

Boot Switch Settings

This option is only available on Am335x EVM. Switch SW3 is for selecting the boot modes. Also, separate DIP switch (SW8) is provided to select various profiles on EVMs.

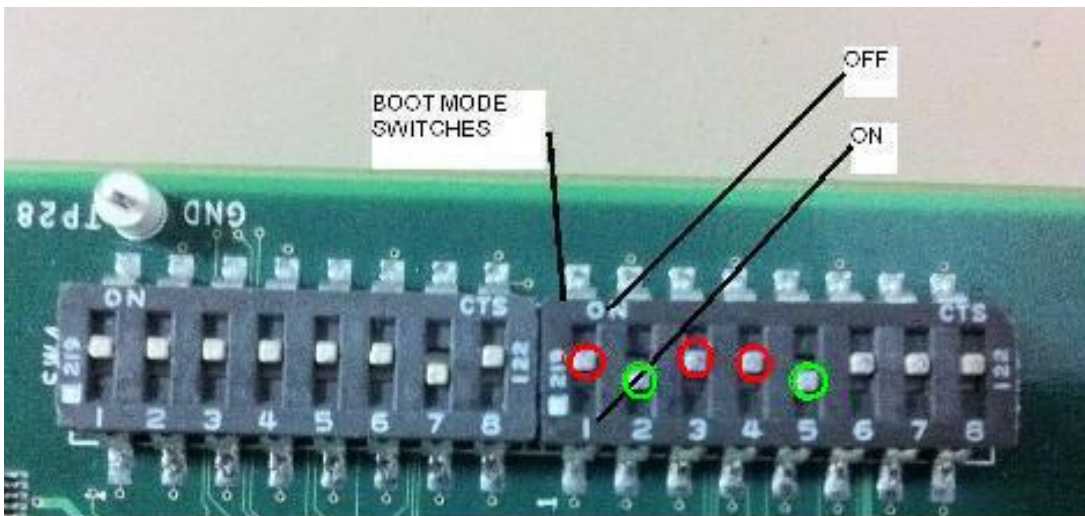
The picture below shows the boot mode configuration switch SW3 on the AM335X EVM. **RED**: circle shows **OFF** and **GREEN** circles shows **ON** switches.

IMPORTANT

ON is labeled on the wrong side of SW3 boot mode switch.

NOTE

The bootmode setting in this picture is for NAND boot.NAND boot corresponds to (SW3 5:1) 10010.



- Make sure that the EVM boot switch settings are set to required boot mode and then power on the board.

NAND

In order to boot from the NAND flash, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	OFF	ON	OFF	OFF	ON

SPI

In order to boot from the SPI flash, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	OFF	ON	ON	OFF	ON

UART

In order to boot from the UART mode, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	ON	OFF	OFF	OFF	OFF

SD

In order to boot from the SD card, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	ON	ON	ON	OFF	ON

CPSW Ethernet

In order to boot from the CPSW ethernet mode, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5	6	7	8
Position	ON	ON	ON	ON	OFF	INDEPENDENT	OFF	ON

NOTE

The setting of switch SW3:[7:6] is because the EVM uses RGMII mode. For more details please refer to the TRM.

NOTE

Due to heavy pin-muxing, boot device is selectively available on selected AM335x EVMs & profiles. Details about the availability of the peripherals on different Profiles can be found from the EVM reference manual ^[3].

Flashing U-Boot with CCS

NOTE

Both the stages of U-Boot need to be flashed on the same media.

The tools are provided in the PSP release to write SPL & U-Boot on to the NAND flash(for NAND boot) .

Refer to AM335x Flashing Tools Guide wiki page for instructions on how to flash the pre-built (or compiled) binary to NAND flash (or the recompiled one) with the help of the NAND flash writer.

After flashing the 2 stages, make sure boot mode is set to NAND and power on the board.

Boot Modes

NAND

NOTE

*The following sub-sections illustrate the usage of NAND specific commands on AM335X EVM.

- Refer to EVM Switch Settings section for more info on enabling/disabling different boot devices.

This section gives an *overview* of the NAND support in U-Boot. It also describe how to store the kernel image, RAMDISK or the UBIFS filesystem to NAND so as to have a network-free boot right from powering on the board to getting the kernel up and running.

Writing to NAND

To write len bytes of data from a memory buffer located at addr to the NAND block offset:

```
U-Boot# nand write <addr> <offset> <len>
```

NOTE

*Offset & len fields should be in align with 0x800 (2048) bytes. On writing 3000 (0xbb8) bytes, len field can be aligned to 0x1000 ie 4096 bytes. Offset field should be aligned to page start address, multiple of 2048 bytes.

If a bad block is encountered during the write operation, it is skipped and the write operation continues from next 'good' block.

For example, to write 0x40000 bytes from memory buffer at address 0x80000000 to NAND - starting at block 32 (offset 0x400000):

```
U-Boot# nand write 0x80000000 0x400000 0x40000
```

Reading from NAND

To read len bytes of data from NAND block at a particular offset to the memory buffer in DDR located at addr:

```
U-Boot# nand read <addr> <offset> <len>
```

If a bad block is encountered during the read operation, it is skipped and the read operation continues from next 'good' block.

For example, to read 0x40000 bytes from NAND - starting at block 32 (offset 0x400000) to memory buffer at address 0x80000000:

```
U-Boot# nand read 0x80000000 0x400000 0x40000
```

Marking a bad block

Some of the blocks in the NAND may get corrupted over a period of time. In such cases you should explicitly mark such blocks as bad so that the image that you are writing to NAND does not end up getting corrupted.

To forcefully mark a block as bad:

```
U-Boot# nand markbad <offset>
```

For example, to mark block 32 (assuming erase block size of 128Kbytes) as bad block - offset = blocknum * 128 * 1024:

```
U-Boot# nand markbad 0x400000
```

Viewing bad blocks

To view the list of bad blocks:

```
U-Boot# nand bad
```

NOTE

*The user marked bad blocks can be viewed by using this command only after a reset.

Erasing NAND

To erase NAND blocks in a particular the address range or using block numbers:

```
U-Boot# nand erase <start offset addr> <len>
```

NOTE

*start offset addr & len fields should align to 0x20000 (64*2048) bytes, i.e.to block size 128KB.

This commands skips bad blocks (both factory and user marked) encountered within the specified range.

For example, to erase blocks 32 through 34:

```
U-Boot# nand erase 0x00400000 0x40000
```

NAND ECC algorithm selection

NAND flash memory, although cheap, suffers from problems like bit flipping which lead to data corruption. However by making use of some error correction coding (ECC) techniques it is possible to work around this problem.

Prior to the AM335xPSP_04.06.00.09-rc2 release, for the data stored in NAND flash, U-Boot supports following NAND ECC schemes

1. S/W ECC (Hamming code)
2. H/W ECC (Hamming code, BCH8)

Starting with the 04.06.00.09-rc2 release only BCH8 is supported, and is used by default in all cases. No user interaction is required to select the algorithm and locations where the **nandecc** command are required can be seen by looking at the history of this page.

BCH Flash OOB Layout

For any ECC scheme we need to add some extra data while writing so as to detect and correct (if possible) the errors introduced by the NAND part. In case of BCH scheme some bytes are needed to store the ECC related info.

The section of NAND memory where addition info like ECC data is stored is referred to as Out Of Band or OOB section.

The first 2 bytes are used for Bad block marker – 0xFFFF => Good block

The next 'N' bytes is used for BCH bytes

$$N = B * \text{<Number of 512-byte sectors in a page>}$$

1. B = 8 bytes per 512 byte sector in BCH4
2. B = 14 bytes per 512 byte sector in BCH8
3. B = 26 bytes per 512 byte sector in BCH16

So for a 2k page-size NAND flash with 64-byte OOB size, we will use BCH8. This will consume 2 + (14*4) = 58 bytes out of 64 bytes available.

The NAND flash part used in EVM does not have enough spare area to support BCH16.

ECC Schemes and their context of usage

ECC type	Usage
S/W ECC	Not used
H/W ECC - Hamming Code	Should use this scheme only for flashing the U-Boot ENV variables.
H/W ECC – BCH8	Should use this scheme while flashing any image/binary other than the U-Boot ENV variables.

To select ECC algorithm for NAND:

```
U-Boot# nandecc [sw | hw <hw_type>]
```

Usage:

sw - Set software ECC for NAND hw <hw_type> - Set hardware ECC for NAND <hw_type> - 0 for Hamming code 1 for bch4 2 for bch8 3 for bch16 Currently we support only Software, Hamming Code and BCH8. We do not support BCH4 and BCH16

ECC schemes usage table

ECC schemes usage table

Component	Default ECC scheme used by the component	ECC scheme to be used to flash the component	ECC schemes supported by the component
SPL	BCH8	BCH8	BCH8
U-boot	Hamming	BCH8	Hamming/BCH8
Linux	BCH8	BCH8	BCH8
File System	NA	BCH8	NA
Environment variables	NA	Hamming	NA

Flashing Kernel

TFTP the kernel uImage to DDR.

```
U-Boot# tftp 0x82000000 <kernel_image>
```

Now flash the kernel image to NAND at the appropriate offset (refer to NAND layout section for the offsets)

```
U-Boot# nand erase 0x00280000 0x00500000
U-Boot# nand write 0x82000000 0x00280000 0x500000
```

NOTE

*Image_size should be aligned to page size of 2048 (0x800) bytes

UBIFS file system flashing

In AM335X, UBIFS file system is used in NAND flash as it is next generation flash file system.

1. Creating and Flashing of UBIFS file system image is described over here

NOTE

In case of AM335x, file system partition is starting from 0x780000. So flashing offset for file system in U-Boot is 0x780000 and from Linux MTD partition number 7 should used for flashing file file system.

UART

This section describes how to use UART boot mode using TeraTerm.

Boot Over UART

NOTE

*The release package does not contain the binary for UART boot. Please follow the steps mentioned here for compiling u-boot and use the *spl/u-boot-spl.bin* file that is produced.

1. Switch ON EVM with switch settings for UART boot. When “CCCC” characters appear on TeraTerm window, from the File Menu select Transfer --> XMODEM --> Send (1K mode)
2. Select “u-boot-spl.bin” for the transfer
3. After image is successfully downloaded, the ROM code will boot it.
4. When “CCCC” characters appear on TeraTerm window, from the File Menu select Transfer --> YMODEM --> Send (1K mode)
5. Select “u-boot.img” for the transfer
6. After image is successfully downloaded, U-Boot will boot it.
7. Hit enter and get to u-boot prompt “U-Boot# ”

Flashing images to NAND in UART boot mode

Boot using UART boot mode as here

After the U-Boot prompt **U-Boot#** comes up, the images for the 1st stage and 2nd stage can be flashed to NAND for persistent storage.

Flashing SPL to NAND in UART boot mode

Flash SPL (**MLO**) to NAND by executing the following commands:

```
U-Boot# loadb 0x82000000
```

- From TeraTerm Menu click “**File -> Transfer -> Kermit -> Send**”.
- Select the 1st stage u-boot image “**MLO**” and click “**OPEN**” button
- Wait for download to complete and then run following commands in u-boot prompt

```
U-Boot# nand erase 0x0 0x20000
U-Boot# nand write 0x82000000 0x0 0x20000
```

If no error messages are displayed the SPL of NAND boot has been successfully transferred to NAND.

Flashing U-Boot to NAND in UART boot mode

Flash the 2nd stage U-Boot (**u-boot.img**) to NAND by executing the following commands:

```
U-Boot# loadb 0x82000000
```

- From TeraTerm Menu click “**File -> Transfer -> Kermit -> Send**”.
- Select the 2nd stage u-boot image “**u-boot.img**” and click “**OPEN**” button
- Wait for download to complete and then run following commands in U-Boot prompt

```
U-Boot# nand erase 0x80000 0x40000
U-Boot# nand write 0x82000000 0x80000 0x40000
```

If no error messages are displayed the U-boot of NAND boot has been successfully transferred to NAND.

SD (Secured Digital card)

This section gives an overview of the SD support in U-Boot

Read and execute uImage from SD card

Please note that the following commands are being executed from the 2nd stage. List files on a FAT32 formatted SD card

```
U-Boot# mmc rescan
U-Boot# fatls mmc 0
```

Booting kernel image from the SD card

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 uImage
U-Boot# bootm 0x82000000
```

Read and execute u-boot from SD card

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 u-boot.bin
U-Boot# go 0x82000000
```

Setting Up Boot Environment on SD Card

This section describes steps to be followed to create a standalone power-on bootable system on SD card.

Prerequisites Ensure that following is available:

- A Linux host with fdisk, sfdisk, mkfs.ext3 and mkfs.vfat utilities is available
- Copy images MLO, u-boot.img, uImage, nfs.tar.gz and mkspd-am335x.sh from release package to a directory on this Linux machine. For subsequent description, we will assume these files are copied to /home/am335x. Please refer "Package Contents" section in AM335x PSP User's Guide for location of these files.
- Empty SD card (at least 256MB, preferably 4GB SDHC)
- A SD memory card reader/programmer to copy files from Linux Host

NOTE

*The SD Boot has some specific restriction about the format of the 1st partition and copying the MLO image.

- So it is recommended that the supplied script mkspd-am335x.sh is used for creating partitions and copying files.

Steps

- Connect the SD memory card using Memory Card reader to the Linux Host
- Note the name allotted for this device. Type "dmesg". The SD/MMC card name should show up near the end, usually something like "SDC" (/dev/sdc) or "SDD" (/dev/sdd).
- Navigate to the /home/am335x directory where all the mentioned files are copied
- Ensure that the script mkspd-am335x.sh has executable permissions
- This scripts expects arguments in the following format

```
./mkspd-am335x.sh <sd-device-name> <sd-1st-stage-bootloader> <sd-2nd-stage-bootloader> <kernel-uImage>
<filesystem>
```

NOTE

that the user will require root/sudo permissions

- In our example, we run the following command

```
sudo ./mkspd-am335x.sh /dev/sdd MLO u-boot.img uImage nfs.tar.gz
```


- You will be asked about data getting overwritten, confirm it and the files along with filesystem will be copied to SD card
- Note that this script will create two primary partitions:
 - 1st partition is formatted as FAT32 containing MLO, u-boot.img, uImage files
 - 2nd partition is formatted as ext3 where the filesystem is extracted in root

Boot using SD card

Once the SD card has been setup as described in the previous section make sure the switch settings are set for SD boot mode and then plug in the SD card in the MMC/SD card slot on the EVM.

Flashing images to NAND in SD boot

Boot using SD boot mode as here.

After the 2nd stage prompt **U-Boot#** comes up, the images for the 1st stage and 2nd stage can be flashed to NAND for persistent storage.

Flashing SPL to NAND in SD boot

Flash SPL (**MLO**) to NAND by executing the following commands:

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 MLO

U-Boot# nand erase 0x0 0x20000
U-Boot# nand write 0x82000000 0x0 0x20000
```

If no error messages are displayed the SPL of NAND boot has been successfully transferred to NAND.

Flashing U-Boot to NAND in SD boot

Flash the 2nd stage U-Boot (**u-boot.img**) to NAND by executing the following commands:

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 u-boot.img
U-Boot# nand erase 0x80000 0x40000
U-Boot# nand write 0x82000000 0x80000 0x40000
```

If no error messages are displayed the U-boot of NAND boot has been successfully transferred to NAND.

Setting U-Boot environment using uEnv.txt

U-Boot environment variables can be modified using a plain text file named uEnv.txt. The scripts can be used to modify and even over-ride the various parameters like bootargs, TFTP serverip etc. If a command named uenvcmd is defined in the file it will be executed. uEnv.txt can be loaded from SD card and tftp server.

Example text file named *uEnv.txt*

IMPORTANT

*The uEnv.txt file should be in **unix** format. Also make sure that there is **an empty line** at the end of the file.

```
bootargs=console=ttyO0,115200n8 root=/dev/mmcblk0p2 mem=128M rootwait
bootcmd=mmc rescan; fatload mmc 0 0x82000000 uImage; bootm 0x82000000
uenvcmd=boot
```

The file *uEnv.txt* is automatically loaded from SD if the bootcmd is run. It can be loaded and put into the environment manually.

Making use pre-existing uEnv on SD card

uEnv.txt on an SD card can be used to override the env settings saved on a persistent storage like NAND by making use of the following commands

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x81000000 uEnv.txt
U-Boot# env import -t 0x81000000 $filesize
U-Boot# boot
```

SPI

Note

- This feature is not supported prior to PSP 04.06.00.08 (and AMSDK 05.05.00.00).
- This feature changed kernel location and filenames with PSP 04.06.00.09 (and AMSDK 05.06.00.00).
- The release package does not contain the binary for SPI boot. Please follow the steps mentioned here for compiling u-boot and use the *MLO.spi* and *u-boot.bin* files that are produced.
- Following those same instructions but building for **am335x_evm_spiboot** rather than **am335x_evm** will result in binaries that will use the SPI flash for environment rather than NAND.

In this example we initially boot from an SD card and use that to transfer the files to write to SPI flash. If loading via other methods, modify the commands below.

1. Switch ON EVM with switch settings such that SPI is present and boot via non-SPI. See SPI boot for more information.
2. Write it to SPI memory by executing the following commands:

```
U-Boot# sf probe 0
U-Boot# sf erase 0 +E0000
U-Boot# mmc rescan
U-Boot# fatload mmc 0 ${loadaddr} MLO.byteswap
U-Boot# sf write ${loadaddr} 0 ${filesize}
U-Boot# fatload mmc 0 ${loadaddr} u-boot.img
U-Boot# sf write ${loadaddr} 0x80000 ${filesize}
```

CPSW Ethernet

Note

- This feature is not supported prior to PSP 04.06.00.08 (and AMSDK 05.05.00.00).
- The release package does not contain the binary for CPSW ethernet boot. Please follow the steps mentioned here for compiling u-boot and use the *spl/u-boot-spl.bin* and *u-boot.img* files that are produced.

Booting Over CPSW Ethernet

- Configure DHCPd and tftpd on your host.
- Within the dhcpd configuration, add entries to send the *u-boot-spl.bin* or *u-boot.img* files based on the vendor-class-identifier field. For ISC dhcpd an example host entry looks like this:

```
host am335x_evm {
    hardware ethernet de:ad:be:ee:ee:ef;
```

```

if substring (option vendor-class-identifier, 0, 10) = "DM814x ROM" {
    filename "u-boot-spl.bin";
} elsif substring (option vendor-class-identifier, 0, 17) = "AM335x U-Boot SPL" {
    filename "u-boot.img";
} else {
    filename "uImage-am335x";
}
}
}

```

- Copy the *u-boot-spl.bin* and *u-boot.img* files you have build to the directory tftpd serves files from.
- Switch ON EVM with switch settings for CPSW ethernet boot.
- Hit enter and get to u-boot prompt "U-Boot# "

Flashing in CPSW Ethernet boot mode

It is possible to build a version of U-Boot that boots over the cpsw ethernet interface and will automatically load and execute a script file. This can be used in initial programming or restoring of boards. It follows the general principles outlined above. In this case first you must build for **am335x_evm_restore_flash** rather than **am335x_evm** when building U-Boot. Next, you will need to create the script file that is run. There are examples provided in the source tree at **doc/am335x.net-spl/debrick-nand.txt** and **doc/am335x.net-spl/debrick-spi.txt**. You should copy one of these files and modify for your exact situation. Once you have modified the script you will need to turn it into a scr file using the following command:

```
./tools/mkimage -A arm -O U-Boot -C none -T script -d <your script> debrick.scr
```

and copy the resulting debrick.scr file to the location tftpd serves files out of. For more information please see the **doc/am335x.net-spl/README** file in the source tree.

U-Boot Network configuration

In order to download the Linux kernel image from the TFTP server and for mounting NFS the network settings in U-Boot need to be configured.

When booting for the first time, U-Boot tries to fetch the MAC address from the env space. If it returns empty, it will look for MAC address from the eFuse registers in the Control module space and set the "ethaddr" variable in the env appropriately. The ethaddr can also be set using the setenv/saveenv commands. In such cases the user-set MAC address will take effect on subsequent reboot only.

To set a different MAC address use the following command

```
U-Boot# set ethaddr <random MAC address eg- 08:11:23:32:12:77>
```

NOTE

*When setting a MAC address please ensure that the LSB of the 1st byte if not 1 i.e. when setting the MAC address: **y** in **xy:ab:cd:ef:gh:jk** has to be an even number. For more info this refer to the wiki page http://en.wikipedia.org/wiki/MAC_address

In case a static ip is not available run the dhcp command to obtain the ip address from the DHCP server on the network to which the EVM is connected.

```

U-Boot# setenv serverip <tftp server in your network>
U-Boot# netmask 255.255.255.0
U-Boot# dhcp
U-Boot# saveenv

```

In case a static ip is available run the following commands

```
U-Boot# setenv ipaddr <your static ip>
U-Boot# saveenv
```

This completes the network configuration in U-Boot.

U-Boot Environment Variables

After completing the network configuration and flashing the kernel image and filesystems to flash you need to set some other parameters which are essential for booting the kernel. We make use of a number of existing helper variables that exist in the environment at present. To see what they are set to use the *printenv* command. After setting **bootargs** along with any other variables for your needs you will need to do:

```
U-Boot# saveenv
```

In all cases we make use of **optargs** to control passing in of additional arguments and **ip_method** to determine how the kernel will deal with networking PRIOR to userspace spawning init. This does not control for example if a dhcp client will be started as part of the userspace init sequence.

Environment Settings for Ramdisk

In case you are using a RAMDISK as the Linux filesystem:

```
U-Boot# setenv bootargs ${console} ${optargs} root=/dev/ram rw initrd=${loadaddr},32MB ip=${ip_method}
```

For booting from NAND:

```
U-Boot# setenv nand_src_addr 0x00280000
U-Boot# setenv nand_img_siz 0x170000
U-Boot# setenv initrd_src_addr 0x00780000
U-Boot# setenv initrd_img_siz 0x320000
U-Boot# setenv bootcmd 'nand read ${kloadaddr} ${nand_src_addr}
${nand_img_siz};nand read ${loadaddr} ${initrd_src_addr}
${initrd_img_siz};bootm ${kloadaddr}'
```

For booting from SD card:

```
U-Boot# setenv bootcmd 'mmc rescan;run mmc_load_uimage;fatload mmc ${mmc_dev} ${loadaddr} initrd.ext3.gz;bootm ${kloadaddr}'
```

NOTE

*The sizes of images mentioned in the above commands have to be modified based on the actual image size. Also, it should be aligned to sector size of the flash device used.

Environment Settings for UBIFS Filesystem

- The U-Boot environment variable **bootargs** contains arguments to be passed to the Linux kernel. The **bootargs** variable here makes use of the **nand_root** variable. The typical format of this variable for a UBIFS root filesystem is:

```
root=ubi0:<VOLUME NAME> ubi.mtd=<PARTITION_ID>,YYYY rw
```

The value of **PARTITION_ID** depends on MTD device which holds the rootfs, **YYYY** depends on the page size of the partition and **VOLUME NAME** depends on the volume name given in ubinize.cfg file while creating UBIFS image as described here . In cases where you have multiple UBI volumes, ubi0 would change to the volume with the root partition.

Once **nand_root** is set:

```
U-Boot# setenv bootcmd run nand_boot
```

Environment Settings for jffs2 Filesystem

The **bootargs** variable here makes use of the **nand_root** variable. The root file system format is jffs2:

Enabling and using the jffs2 as a root file system is describe here ^[4]

Environment Settings for NFS Filesystem

NOTE

*When setting a MAC address please ensure that the LSB of the 1st byte if not 1 i.e. when setting the MAC address: **y** in **xy:ab:cd:ef:gh:jk** has to be an even number. For more info this refer to the wiki page http://en.wikipedia.org/wiki/MAC_address

- If you need to use a separate server for TFTP and NFS please see how to use the **next-server** option in your DHCP server.

Modify the required variables:

```
U-Boot# print ethaddr                <-- Check if MAC address is assigned and is unique
U-Boot# setenv ethaddr <unique-MAC-address>  <-- Set only if not present already, format uv:yy:zz:aa:bb:cc
U-Boot# setenv serverip <NFS and TFTP server-ip>
U-Boot# setenv rootpath /location/of/nfsroot/export
U-Boot# setenv bootcmd net_boot
```

Booting the kernel

In case everything went well, boot the kernel using the following command.

```
U-Boot# boot
```

References

- [1] <http://www.denx.de/wiki/U-Boot/WebHome>
- [2] http://software-dl.ti.com/dsps/dsps_public_sw/am_bu/sdk/AM335xSDK/latest/index_FDS.html
- [3] <http://logmett.com/index.php?/products/teraterm.html>
- [4] http://processors.wiki.ti.com/index.php/AM335x_JFFS2_Support_Guide

Sitara Linux SDK Top-Level Makefile

Return to the [Sitara Linux Software Developer's Guide](#)



Overview

Inside of the Sitara Linux SDK there is a top-level Makefile that can be used to build some of the sub-components found within the SDK. This Makefile uses the Rules.make file and gives an example of how the various components can be built and the parameters to use.

NOTE: You should not call this makefile with the [environment-setup script](#) sourced. The sub-component Makefiles will handle sourcing this script where appropriate, but some make targets such as the Linux kernel make target do not work properly when this script is already sourced.

Rules.make

The following sections cover the Rules.make file found in the top-level of the Sitara Linux SDK.

Purpose

The **Rules.make** file in the top-level of the Sitara Linux SDK is used not only by the top-level Makefile, but also by many of the sub-component Makefiles to gain access to common shared variables and settings. The next section covers the main variables defined in the Rules.make file.

Variables Defined

- **PLATFORM** - This represents the machine name of the device supported by the SDK. This machine name has a direct correlation to the machine definition in the Arago project ^[1] build system. The PLATFORM variable can be used by component Makefiles to make decisions on a per-machine basis.
- **ARCH** - This represents the architecture family of the machine. This can be used by component Makefiles to change settings such as mtune values in CFLAGS based on the architecture of the PLATFORM.
- **UBOOT_MACHINE** - This is used when building u-boot to configure the u-boot sources for the correct device.
- **TI_SDK_PATH** - This points to the top-level of the SDK. This is the same directory where the Rules.make file itself is located.
- **DESTDIR** - This points to the base installation directory that applications/drivers should be installed to. This is usually the root of a target file system but can be changed to point anywhere. By default the initial value is a unique key value of `__DESTDIR__` which is replaced with the location of the target NFS file system when the [setup.sh](#) script is run.
- **LINUX_DEVKIT_PATH** - This points to the linux-devkit directory. This directory is the base directory containing the cross-compiler and cross-libraries as well as the [environment-setup](#) script used by many component Makefiles to source additional variable settings.
- **CROSS_COMPILE** - This setting represents the CROSS_COMPILE prefix to be used when invoking the cross-compiler. Many components such as the Linux kernel use the variable CROSS_COMPILE to prepend the proper prefix to commands such as `gcc` to invoke the ARM cross-compiler.

- **ENV_SETUP** - This points to the **environment-setup** script in the linux-devkit directory used by many components to configure for a cross-compilation build.
- **LINUXKERNEL_INSTALL_DIR** - This points to the location of the Linux kernel sources, which is used by components such as out-of-tree kernel drivers to find the Linux kernel Makefiles and headers.

Makefile

The following sections cover the Makefile found in the top-level of the Sitara Linux SDK

Target Types

For each of the targets discussed below the following target type are defined

- **<target>** - This is the build target which will compile the release version of the component
- **<target>_install** - This target will install the component to the location pointed to by DESTDIR
- **<target>_clean** - This target will clean the component

Top-Level Targets

The Sitara Linux SDK provides the following targets by default which will invoke the corresponding component targets:

- **all** - This will call the build target for each component defined in the Makefile
- **install** - This will call the install target for each component defined in the Makefile
- **clean** - This will call the clean target for each component defined in the Makefile

Common Targets

The following targets are common to all Sitara Linux SDKs

- **linux** - Compiles the Linux kernel using the default tisdk_<PLATFORM>_defconfig configuration
- **matrix-gui** - Builds the matrix-gui sources
- **am-benchmarks** - Builds the ARM Benchmarks for the ARCH defined in Rules.make
- **am-sysinfo** - Build the helper applications used by the system settings demos in Matrix
- **matrix-gui-browser** - Builds the matrix GUI browser Qt project
- **refresh-screen** - Builds the refresh screen Qt project

Additional Targets

Depending on the capabilities and software available for a given device additional targets may also be defined. You can find the list of all the targets by looking at the **all** target as described in the **Top-Level Targets** section above. Add devices will have one or the other of the following targets depending on the u-boot version used:

- **u-boot-spl** - This target will build both u-boot and the u-boot SPL (MLO) binaries used in newer versions of u-boot. This actually provides a u-boot and u-boot-spl target in the Makefile.
- **u-boot-legacy** - This target will build the u-boot binary for older versions of u-boot which do not support the SPL build.
- **wireless** - Starting with Sitara Linux SDK 05.04 there is now a wireless top-level build target that can be used to rebuild the wireless drivers against the kernel sources in the board-support directory.

Usage Examples

The following examples demonstrate how to use the top-level Makefile for some common tasks. All of the examples below assume that you are calling the Makefile from the top-level of the SDK.

- Build Everything

```
host# make
```

- Clean Everything

```
host# make clean
```

- Install Everything

```
host# make install
```

- Build the Linux kernel

```
host# make linux
```

- Install the Linux kernel modules

```
host# make linux_install
```

- Build the ARM Benchmarks

```
host# make am-benchmarks
```

- Clean the ARM Benchmarks

```
host# make am-benchmarks_clean
```

- Install the ARM Benchmarks

```
host# make am-benchmarks_install
```

A Note about Out-of-tree Kernel Modules

Some drivers like the SGX drivers are delivered as modules outside of the kernel tree. If you rebuild the kernel and install it using the "make linux_install" target you will also need to rebuild the out of tree modules and install them as well. The modules_install command used by the linux_install target will remove any existing drivers before installing the new ones. This means those drivers are no longer available until they have been rebuilt against the kernel and re-installed.

References

- [1] <http://arago-project.org>

Sitara Linux SDK GCC Toolchain

Return to the [Sitara Linux Software Developer's Guide](#)



Overview

Starting with SDK 05.02 the Sitara Linux SDK contains a cross-compile toolchain and pre-built cross-compile libraries within the SDK. This removes the need to download and install a separate toolchain. These toolchains have also been optimized for the processor family supported by the SDK, rather than being a generic ARM toolchain. This allows for devices based on the ARM Cortex-A8 family to have a toolchain that takes advantage of accelerators such as the Neon co-processor.

Location in SDK

The toolchain is located in the Sitara Linux SDK in the `<SDK INSTALL DIR>/linux-devkit` directory. The following sections will cover the key components of the toolchain.

Cross-Compilers/Tools

The cross-compilers and tools such as `qmake2` can be found in the `<SDK INSTALL DIR>/linux-devkit/bin` directory. Adding this directory to your `PATH` will allow using these tools. For example:

```
host# export PATH="<SDK INSTALL DIR>/linux-devkit/bin:$PATH"
```

the cross-compile tools are prefixed with the `arm-arago-linux-gnueabi-` prefix. i.e. the GCC cross compiler is called `arm-arago-linux-gnueabi-gcc`. Additional tools are also located here such as the `qmake2`, `rcc`, `uic` tools used by Qt. In addition there is a `qt.conf` file that can be used by tools such as Qt creator to use the pre-built libraries found in the Sitara Linux SDK.

Cross-Compiled Libraries

The toolchain within the Sitara Linux SDK contains more than just the cross-compiler, it also contains pre-built libraries that can be used in your applications without requiring you to cross-compile them yourself. These libraries include packages from `alsa` to `zlib`. The libraries are located in the `<SDK INSTALL DIR>/linux-devkit/arm-arago-linux-gnueabi` directory. For a list of the libraries you can refer to the software manifest found in the `<SDK INSTALL DIR>/docs` directory or look at the list of libraries available in the `<SDK INSTALL DIR>/linux-devkit/arm-arago-linux-gnueabi/usr/lib` directory. You will also find the header files corresponding to these libraries in the `<SDK INSTALL DIR>/linux-devkit/arm-arago-linux-gnueabi/usr/include` directory. Usage of these libraries will be covered in more detail in the next sections, but as an example if your application wants access to the `alsa` sound library then you can now do the following command (assuming you have added the cross compiler to your `PATH`):

```
host# arm-arago-linux-gnueabi-gcc -lasound app.c -o app.out
```

environment-setup script

When cross-compiling packages that use configuration tools and autotools there are many settings that are required to make sure that the proper cross-compile libraries are used. The **environment-setup** script located in the **<SDK INSTALL DIR>/linux-devkit** directory handles this for you. This script exports variables to perform actions such as:

- Adding the toolchain to the PATH
- Setting up CPATH
- Setting up LIBTOOL_SYSROOT_PATH
- Setting up PKG_CONFIG_* paths
- Setting standard variable such as CC, CPP, AR to the cross-compile values

To *use* the environment-setup script you only need to source it. This is as simple as:

```
host# source linux-devkit/environment-setup
```

To know if the environment setup script has been sourced in your current shell the shell prompt will be changed to contain the **[linux-devkit]:** prefix in the command prompt.

The **Usage** section below will cover some cases where using the environment-setup script is useful.

When Compiling the Linux Kernel

Because environment-setup changes standard variables such as CC you should not use this script when compiler projects that build host-side as well as target-side tools. A prime example of this is the Linux kernel, which builds some host side tools to help during the kernel build. If the environment-setup script has been sourced then the CC value will specify the cross-compiler for the host-side tool build. This means that the tools compiled and used during the kernel build will be compiled for the ARM platform while the kernel build tries to run these tools on an Intel platform. This will cause the tools to fail to run and the kernel to fail to compile.

Usage

The following sections give some examples of how to use the included toolchain to compile simple applications such as *HelloWorld* to more complex examples such as configuring and compiler GStreamer plugins.

Simple Cross-Compile

In the simplest case the cross-compiler can be used to compile simple applications that just need access to standard libraries. The two examples below cover an application that uses only the standard libgcc libraries and another example that uses the pthreads threading library.

HelloWorld

Simple applications like **HelloWorld** can be compiled using just a call to the cross-compiler since the cross-compiler can find the libraries it was built with without any issues. The following steps will show how to make a simple helloworld application and cross-compile that application.

1. Create a **helloworld.c** file

```
#include <stdio.h>

int main() {
    printf ("Hello World from Sitara!!!\n");
    return 0;
}
```

2. Cross-compile the **helloworld.c** file using the cross-compile toolchain. In this example we will invoke the toolchain without it having been added to our PATH.

```
host# <SDK INSTALL DIR>/linux-devkit/bin/arm-arago-linux-gnueabi-gcc helloworld.c -o helloworld
```

After the above steps are run you should now have a **helloworld** executable in your directory that has been compiled for the ARM. A simple way to check this is to run the following command:

```
host# file helloworld
```

This should yield output like:

```
helloworld: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.16, not stripped
```

NOTE: the ARM entry above was made bold for emphasis

Using PThreads

In many cases your simple application probably wants to use additional libraries that the standard libgcc and glibc libraries. In this case you will need to include the header files for those libraries as well as add the library to the compile line. In this example we will look at how to build a simple threading application and use the pthread library. This example was derived from the example code at <http://www.amparo.net/ce155/thread-ex.html>" [1]

- Create a file **thread-ex.c** with the following contents

```
#include <unistd.h>
#include <sys/types.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>

int print_message_function(void *ptr);

/* struct to hold data to be passed to a thread
   this shows how multiple data items can be passed to a thread */
typedef struct str_thdata
{
    int thread_no;
    char message[100];
} thdata;

int main(int argc, void **argv)
{
    pthread_t thread1, thread2;
    thdata data1, data2;

    data1.thread_no = 1;
    strcpy(data1.message, "Hello!");

    data2.thread_no = 2;
    strcpy(data2.message, "Hi!");
```

```

pthread_create (&thread1, NULL, (void *) &print_message_function, (void *) &data1);
pthread_create (&thread2, NULL, (void *) &print_message_function, (void *) &data2);

pthread_join(thread1, NULL);
pthread_join(thread2, NULL);

exit(0);
}

int print_message_function ( void *ptr )
{
    thdata *data;
    data = (thdata *) ptr; /* type cast to a pointer to thdata */

    /* do the work */
    printf("Thread %d says %s \n", data->thread_no, data->message);

    return 0;
}

```

- Cross-compile the **thread-ex.c** file using the cross-compile toolchain. In this example we will first add the toolchain to our PATH. This only needs to be done once. We will also add the pthread library to the compile line so that we will link with the library file that provides the pthread_* functions.

```

export PATH="<SDK INSTALL DIR>/linux-devkit/bin:$PATH"
arm-arago-linux-gnueabi-gcc -lpthread thread-ex.c -o thread-ex

```

NOTE: the **-lpthread** entry above was made bold for emphasis

Configure/Autotools

The last case to cover is one where the **environment-setup** script is useful. In this case we will download the *gst-plugins-bad* package and configure and build it using the environment-setup script to configure the system for the *autotools* to properly detect the libraries available as pre-built libraries.

1. First download the *gst-plugins-bad-0.10.11.tar.gz* ^[2] package

```
wget http://gstreamer.freedesktop.org/src/gst-plugins-bad/gst-plugins-bad-0.10.11.tar.gz
```

IMPORTANT

In order to build the *gst-plugins-bad* package you will need *libglib2.0-dev* installed on your system. You can install this using **sudo apt-get install libglib2.0-dev**

2. Extract the plugins tarball

```
tar xzf gst-plugins-bad-0.10.11.tar.gz
```
3. Change directory into the extracted sources

```
cd gst-plugins-bad-0.10.11
```
4. Source the **<SDK INSTALL DIR>/environment-setup** script to prepare to configure and build the plugins.

```
source <SDK INSTALL DIR>/environment-setup
```
5. Now configure the package. We need to define the *host* setting to tell the configuration utility what our host system is, and we will also disable some plugins that are known to be bad.

```
./configure --host=i686 --disable-deinterlace2 --disable-x264
```

6. When the configuration is done the last sections will show which plugins will be build based on the libraries available. This is the key point behind what the environment-setup script provides. By setting up the `PKG_CONFIG_*` paths and other variables the configure script was able to check for required libraries being available to know which plugins to enable. Now that the sources have been configured you can compile them with a simple make command.

```
make
```

References

[1] <http://www.amparo.net/ce155/thread-ex.html>

[2] <http://gststreamer.freedesktop.org/src/gst-plugins-bad/gst-plugins-bad-0.10.11.tar.gz>

Sitara Linux SDK create SD card script

Return to the [Sitara Linux Software Developer's Guide](#)



Overview

Starting with the Sitara Linux SDK version 05.04.00.00 there is now a script in the `<SDK INSTALL DIR>/bin` directory named `create-sdcard.sh`. The purpose of this script is to create SD cards for the following high-level use cases:

1. Create the **SD card using default images** from the Sitara Linux SDK
2. Create the **SD card using custom images**
3. Create the **SD card using partition tarballs** (This is not common and is used most often by board vendors)

The script will give you information about each step, but the following sections will go over the details for the use cases above and walk you through how to use the script as well.

Common Steps

No matter which use case above that you are creating an SD card for the following steps are the same.

Invoking the Script

The `create-sdcard.sh` script can be run from any location but must be run with **root** permissions. This usually means using the `sudo` command to start execution of the script. For example:

```
host# sudo <SDK INSTALL DIR>/bin/create-sdcard.sh
```

If you fail to execute the script with root permissions you will receive a message that root permissions are required and the script will exit.

Select the SD Card Device

The first step of the script will ask you to select the drive representing the SD card that you want to format. In most cases your host root file system drive has been masked off to prevent damage to the host system. When prompted enter the device number corresponding to the SD card. For example if the output looks like:

```
Available Drives to write images to:
```

```
# major  minor    size  name
1:    8        16    7761920 sdb
```

```
Enter Device Number:
```

You would enter **1** to select the *sdb* device

Re-Partitioning the SD Card

Any partitions of the device that are already mounted will be un-mounted so that the device is ready for partitioning. If the SD Card already has partition you will see a prompt like the following asking you if you would like to repartition the card. If the card was not already partitioned then this step will be skipped and you can move on to the next step.

```
Would you like to re-partition the drive anyways [y/n] :
```

- Options:
 - **y** - This will allow you to change the partitioning of the SD card. For example if you have a 3 partition card and want to create a 2 partition card to give additional storage space to the root file system you would select **y** here.
NOTE: This operation **WILL ERASE** the contents of your SD card
 - **n** - If the SD card already has the desired number of partitions then this will leave the partitioning alone. If you select **n** here skip on to the [Installing SD Card Content](#) section.

Select Number of Partitions

You should now see a prompt like the following which will ask you how many partitions you want to create for the SD card

```
Number of partitions needed [2/3] :
```

- Options:
 - **2** - This is the most common use case and will give the most space to the root file system.
 - **3** - This case should only be used by board manufacturers making SD cards to go in the box with the EVM. This requires access to the partition tarballs used for Out-Of-Box SD cards. This option should be selected if you are going to follow the [SD card using partition tarballs](#) steps below

After selecting the number of partitions move on to the next section.

Installing SD Card Content

After the SD card is partitioned you will be prompted whether you want to continue installing the file system or safely exit the script.

- Options:
 - **y** - Selecting yes here will begin the process of installing the SD card contents. This operation **WILL ERASE** any existing data on the SD card. Refer to one of the following sections for additional instructions depending on which use case you are creating an SD card for
 - Create the **SD card using default images**
 - Create the **SD card using custom images**
 - Create the **SD card using partition tarballs**
 - **n** - Selecting no here will allow you to have partitioned your card but will leave the partitions empty.

SD Card Using Default Images

The purpose of this section is to cover how to use the **create-sdcard.sh** script to populate an SD card that can be used to boot the device using the default images that ship with the Sitara Linux SDK.

NOTE: For devices like AM180x see the **AM180x SD card boot** section below for information on booting that device from an SD card. This script will create a card suitable for the root file system but additional steps are required to load the boot loaders from the SD card.

Prerequisites

1. The Sitara Linux SDK is installed on your host system
2. The SD card you wish to create is inserted into the host system and has a size sufficiently large to hold at least the bootloaders, kernel, and root file system.
3. You have started running the script as detailed in the **Common Steps** section above.

Choose Install Pre-built Images

You should now see a prompt like:

```
#####
Choose file path to install from

1 ) Install pre-built images from SDK
2 ) Enter in custom boot and rootfs file paths

#####

Choose now [1/2] :
```

You should choose option **1** to create an SD card using the pre-built images from the SDK.

If you executed this script from within the SDK then the script can determine the SDK path automatically and will start copying the contents to the SD card. Once the files are copied the script will exit.

If you executed the script from outside of the SDK (i.e. you copied it to some other directory and executed it there) please see the next section.

Enter SDK Path

In the case that the script was invoked from a directory without the SDK installation in the path, i.e. the script was copied to your home directory and executed there you may see a prompt like

```
no SDK PATH found
Enter path to SDK :
```

Enter the path to the SDK installation directory here. For example if the SDK was installed into the home directory of the *sitara* user the path to enter would be `/home/sitara/ti-sdk-<machine>-<version>`. You will be prompted to confirm the installation directory. The SD card will then be created using the default images and the script will exit when finished.

SD Card Using Custom Images

Often times you will use TFTP and NFS during development to transfer you kernel images and boot your root file systems respectively. Once you are done with your development you may want place these images onto an SD card so that they can be used stand-alone without requiring a network connection to a server.

Prerequisites

1. The Sitara Linux SDK is installed on your host system
2. The SD card you wish to create is inserted into the host system and has a size sufficiently large to hold at least the bootloaders, kernel, and root file system.
3. You have started running the script as detailed in the [Common Steps](#) section above.

Choose Custom Images

You should now see a prompt like:

```
#####

Choose file path to install from

1 ) Install pre-built images from SDK
2 ) Enter in custom boot and rootfs file paths

#####

Choose now [1/2] :
```

Select option **2** to create an SD card with your custom images.

Select Boot Partition

You will now be prompted to provide a path to the location of the boot partition files. The prompt will explain the requirements of the files to be placed at the path, but the basic options are:

1. Point to a tarball containing all of the files you want placed on the boot partition. This would include the boot loaders and the kernel image as well as any optional files like uEnv.txt
2. Point to a directory containing the files for the boot partition like those in the first option.

The script is intelligent enough to recognize whether you provided a tarball or a directory path and will copy the files accordingly. You will be given a list of the files that are going to be copied and given the option to change the path if the list of files is not correct.

Select Root Partition

You will now be prompted to provide a path to the location of the root file system partition files. The prompt will explain the requirements of the files to be placed at the path, but the basic options are:

1. Point to a tarball of the root file system you want to use
2. Point to a directory containing the root file system such as an NFS share directory.

The script is intelligent enough to recognize whether you provided a tarball or a directory path and will copy the files accordingly. You will be given a list of the files that are going to be copied and given the option to change the path if the list of files is not correct.

SD Card Using Partition Tarballs

This option is meant for board vendors to create SD cards to go in the box with the EVM. It requires access to the three tarballs representing the the partitions of the SD card shipped with the EVM.

Prerequisites

1. The Sitara Linux SDK is installed on your host system
2. The SD card you wish to create is inserted into the host system and has a size sufficiently large to hold at least the bootloaders, kernel, and root file system.
3. You have started running the script as detailed in the [Common Steps](#) section above.

Provide Tarball Location

After the SD card has been partitioned you will be prompted to

```
Enter path where SD card tarballs were downloaded :
```

Point to the directory containing the following tarball files:

- **boot_partition.tar.gz**
- **rootfs_partition.tar.gz**
- **start_here_partition.tar.gz**

The script will show you the contents of the directory given and ask you to verify that the tarballs are present in that directory. The SD card will then be populated with the contents of the tarballs and be ready for inclusion in the box with the EVM.

AM180x SD Card Boot

For the AM180x device the boot ROM on most silicon versions does not support reading the boot loaders from the SD card vfat partition like other devices. For these devices you have the following options:

1. Flash the boot loaders into persistent storage on the device and use the SD card for the root file system by creating an SD card as detailed above. In this case the SD card will have a boot partition but the boot loaders on that partition will not be used. The kernel image can still be loaded from the SD card boot partition.
 - Please refer to the [AM18x Flash Tool User's Guide](#) for instructions on how to flash the boot loaders into SPI Flash
2. It is also possible to use the tool described in the [AM18x Flash Tool User's Guide](#) to flash only the UBL into the SPI Flash and then to place u-boot.bin in a special partition of the SD card.
 - To learn how to put u-boot.bin on the SD card please see [Creating bootable SD card for OMAP-L138 EVM board](#) section on the wiki.
 - **NOTE:** Use of this method is incompatible with the `create-sdcard.sh` script.

How to add a JVM



Return to the [Sitara Linux Software Developer's Guide](#)

Introduction

This article shows the steps necessary to add an Oracle Hotspot JVM to your Sitara SDK. Due to the licensing of this JVM, it can not be delivered with the Sitara SDK. You must download it and install it separately. Customers wishing to evaluate this JVM should also investigate and evaluate the Oracle Hotspot license.

Download the JVM

This can be done from the **Oracle** link: <http://www.oracle.com/technetwork/java/embedded/downloads/javase/index.html>

1. Select the version that matches your processor.
 - ARMv7 for Cortex-A8 and ARMv5 for ARM9
2. Select the headless version. To use the full capabilities of the headful version, you need to have some windowing support like X11 in your filesystem. Sitara SDK currently does not support X11. However if you download the headful version, you can run Caffeinemark, but you will not be able to use the display capabilities.
3. Before you can download, you have to agree to their license and click through a few required inputs.
4. Download the JVM

Install the JVM

1. Untar the JVM to your target filesystem. It can go anyplace you want. For this wiki we are assuming you install at /home/root

See this article if you're not sure how to untar: <http://www.freebsdidiary.org/untar.php>

2. Setup your PATH environment variable on your target filesystem to point to the location of the "java" executable which can be found in the "bin" directory located in the root directory of the JVM.

to automate setting of the PATH, add the following line to the file: /etc/profile (NOTE: for this example version 1.7.0_04 was used, your version may be different)

```
PATH=$PATH:/home/root/ejrel.7.0_04/bin
```

add the above line before the following line which is found near the bottom of /etc/profile

```
export PATH PS1 OPIEDIR QPEDIR QTDIR EDITOR TERM
```

3. Now each time you login, your path will be automatically set up to include the "java" executable from your installed JVM. By typing "java -version", you can verify you are setup correctly.

- you should see the JVM version displayed similar to the results below. Your version may be different depending on which one you downloaded:

```
root@am335x-evm:~# java -version
java version "1.7.0_04-ea"
Java(TM) SE Runtime Environment for Embedded (build 1.7.0_04-ea-b20, headless)
Java HotSpot(TM) Embedded Client VM (build 23.0-b21, mixed mode)
```

Running a benchmark - Pendragon Software's CaffeineMark(tm) ver. 3.0

CaffeineMark is a Opensource benchmark which can be downloaded here: <http://www.benchmarkhq.ru/cm30/info.html>

1. Download the embedded version to your target file system.
2. To run:

```
java CaffeineMarkEmbeddedApp
```

3. Benchmark information:

- There are 6 tests associated with the Embedded version. Each tests runs for about the same amount of time and reports back how many iterations of the test could be completed in allocated time period. The higher the score the better. There is also an overall result which takes in to account all the test results.

Results for CaffeineMark running on Sitara

The following results were performed without independent verification by Pendragon Software and Pendragon Software makes no representations or warranties as to the result of the test

AM37x EVM running with a MPU clock rate of 1GHz

AM37x EVM achieved a CaffeineMark(tm) 3.0 score of 16046 running on linux-2.6.37

```
root@am37x-evm:~/ttt# java CaffeineMarkEmbeddedApp
Sieve score = 13991 (98)
Loop score = 29054 (2017)
Logic score = 24167 (0)
```

```
String score = 8910 (708)
Float score = 10483 (185)
Method score = 18603 (166650)
Overall score = 16046
```

AM335x EVM running with a MPU clock rate of 720MHz

AM335x EVM achieved a CaffeineMark(tm) 3.0 score of 11172 running on linux-3.2

```
root@am335x-evm:~/ttt# java CaffeineMarkEmbeddedApp
Sieve score = 9747 (98)
Loop score = 20384 (2017)
Logic score = 17383 (0)
String score = 6047 (708)
Float score = 7091 (185)
Method score = 13136 (166650)
```

Overall score = 11172

AM335x U-Boot User's Guide



AM335x U-Boot User's Guide

Linux PSP

U-Boot

In AM335x the ROM code serves as the 1st stage bootloader. The 2nd and the 3rd stage bootloaders are based on U-Boot ^[1]. *In the rest of this document when referring to the binaries, the binary for the 2nd stage is referred to as SPL and the binary for the 3rd stage as simply U-Boot.* SPL is a non-interactive loader and is a specially built version of U-Boot. It is built concurrently when building U-Boot.

The ROM code can load the SPL image from any of the following devices

- **Memory devices non XIP (NAND/SDMMC)**

The image should have the Image header. The image header is of length 8 byte which has the load address(Entry point) and the size of the image to be copied. RBL would copy the image, whose size is given by the length field in the image header, from the device and loads into the internal memory address specified in the load address field of Image header.

- **Peripheral devices (UART)**

RBL loads the image to the internal memory address 0x402f0400 and executes it. No Image Header present.

Two stage U-Boot design

This section gives an overview of the two stage U-Boot approach adopted for AM335X.

The size of the internal RAM in AM335X is 128KB out of which 18KB at the end is used by the ROM code. Also, 1 KB at the start (0x402f0000 - 0x402f0400) is secure and it cannot be accessed. This places a limit of 109KB on the size of the U-Boot binary which the ROM code can transfer to the internal RAM and use as an initial stack before initialization of DRAM.

Since it is not possible to squeeze in all the functionality that is normally expected from U-Boot in < 110KB (after setting aside some space for stack, heap etc) a two stage approach has been adopted. Initial stage initialize only the required boot devices (NAND, MMC, I2C etc); 2nd full stage initialize all other devices (ethernet, timers, clocks etc). The 1st binary is generated MLO and the 2nd stage is generated as u-boot.img.

NOTE

*When using memory boot (NAND) a header needs to be attached to the SPL binary indicating the load address and the size of the image. SPI boot additionally requires endian conversion before flashing the image.

- When using peripheral boot (UART) there can be no header as the load address is fixed.

Building U-Boot

Prerequisite

GNU toolchain for ARM processors from Arago is recommended. Arago Toolchain can be found in the linux-devkit directory of the SDK here ^[2]

NOTE

Below steps assumes that the release package is extracted inside directory represented as \$AM335x-PSP-DIR

Change to the base of the U-Boot directory.

```
$ cd ./AM335x-LINUX-PSP-MM.mm.pp.bb/src/u-boot/u-boot-MM.mm.pp.bb
```

Building into a separate object directory with the "O=" parameter to make is strongly recommended.

Commands

```
$ [ -d ./am335x ] && rm -rf ./am335x
$ make O=am335x CROSS_COMPILE=arm-arago-linux-gnueabi- ARCH=arm am335x_evm
```

This will generate two binaries in the am335x directory, MLO and u-boot.img along with other intermediate binaries that may be needed in some cases (see below).

Host configuration

Serial port configuration

Connect a serial cable from the serial port of the EVM (serial port is next to the power switch) to the COM port on either the Windows machine or Linux host depending on where you'll be running the serial terminal software.

For correct operation the serial terminal software should be configured with the following settings:

```
*Baud rate: 115,200
*Data bits: 8
*Parity: None
*Stop bits: 1
*Flow control: None
```

NOTE

If Teraterm is being used, ensure that the latest version (4.67 is the latest version as of writing this user guide) of Teraterm is installed. The implementation of the Kermit protocol in Teraterm is not reliable in older versions. The latest version of Teraterm 4.67 can be downloaded from here ^[3]. Recent Teraterm updates causes slow Binary transfer over UART. In such cases, use Windows in-built HyperTerminal application.

Target configuration

Boot Switch Settings

This option is only available on Am335x EVM. Switch SW3 is for selecting the boot modes. Also, separate DIP switch (SW8) is provided to select various profiles on EVMs.

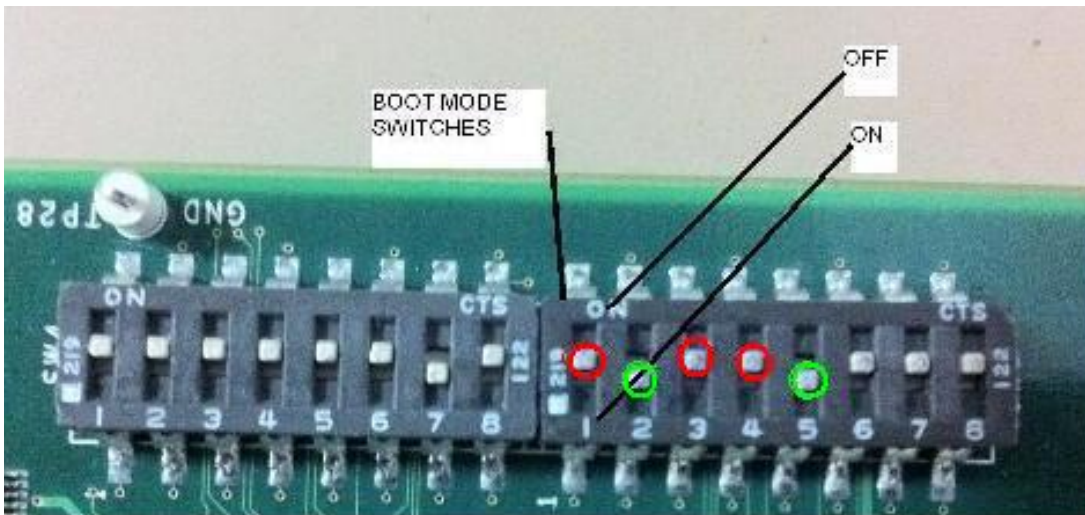
The picture below shows the boot mode configuration switch SW3 on the AM335X EVM. **RED** circle shows **OFF** and **GREEN** circles shows **ON** switches.

IMPORTANT

ON is labeled on the wrong side of SW3 boot mode switch.

NOTE

The bootmode setting in this picture is for NAND boot.NAND boot corresponds to (SW3 5:1) 10010.



- Make sure that the EVM boot switch settings are set to required boot mode and then power on the board.

NAND

In order to boot from the NAND flash, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	OFF	ON	OFF	OFF	ON

SPI

In order to boot from the SPI flash, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	OFF	ON	ON	OFF	ON

UART

In order to boot from the UART mode, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	ON	OFF	OFF	OFF	OFF

SD

In order to boot from the SD card, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5
Position	ON	ON	ON	OFF	ON

CPSW Ethernet

In order to boot from the CPSW ethernet mode, set the SW3 switch as follows:

Dip switch #	1	2	3	4	5	6	7	8
Position	ON	ON	ON	ON	OFF	INDEPENDENT	OFF	ON

NOTE

The setting of switch SW3:[7:6] is because the EVM uses RGMII mode. For more details please refer to the TRM.

NOTE

Due to heavy pin-muxing, boot device is selectively available on selected AM335x EVMs & profiles. Details about the availability of the peripherals on different Profiles can be found from the EVM reference manual ^[3].

Flashing U-Boot with CCS

NOTE

Both the stages of U-Boot need to be flashed on the same media.

The tools are provided in the PSP release to write SPL & U-Boot on to the NAND flash(for NAND boot) .

Refer to AM335x Flashing Tools Guide wiki page for instructions on how to flash the pre-built (or compiled) binary to NAND flash (or the recompiled one) with the help of the NAND flash writer.

After flashing the 2 stages, make sure boot mode is set to NAND and power on the board.

Boot Modes

NAND

NOTE

*The following sub-sections illustrate the usage of NAND specific commands on AM335X EVM.

- Refer to EVM Switch Settings section for more info on enabling/disabling different boot devices.

This section gives an *overview* of the NAND support in U-Boot. It also describe how to store the kernel image, RAMDISK or the UBIFS filesystem to NAND so as to have a network-free boot right from powering on the board to getting the kernel up and running.

Writing to NAND

To write len bytes of data from a memory buffer located at addr to the NAND block offset:

```
U-Boot# nand write <addr> <offset> <len>
```

NOTE

*Offset & len fields should be in align with 0x800 (2048) bytes. On writing 3000 (0xbb8) bytes, len field can be aligned to 0x1000 ie 4096 bytes. Offset field should be aligned to page start address, multiple of 2048 bytes.

If a bad block is encountered during the write operation, it is skipped and the write operation continues from next 'good' block.

For example, to write 0x40000 bytes from memory buffer at address 0x80000000 to NAND - starting at block 32 (offset 0x400000):

```
U-Boot# nand write 0x80000000 0x400000 0x40000
```

Reading from NAND

To read len bytes of data from NAND block at a particular offset to the memory buffer in DDR located at addr:

```
U-Boot# nand read <addr> <offset> <len>
```

If a bad block is encountered during the read operation, it is skipped and the read operation continues from next 'good' block.

For example, to read 0x40000 bytes from NAND - starting at block 32 (offset 0x400000) to memory buffer at address 0x80000000:

```
U-Boot# nand read 0x80000000 0x400000 0x40000
```

Marking a bad block

Some of the blocks in the NAND may get corrupted over a period of time. In such cases you should explicitly mark such blocks as bad so that the image that you are writing to NAND does not end up getting corrupted.

To forcefully mark a block as bad:

```
U-Boot# nand markbad <offset>
```

For example, to mark block 32 (assuming erase block size of 128Kbytes) as bad block - offset = blocknum * 128 * 1024:

```
U-Boot# nand markbad 0x400000
```

Viewing bad blocks

To view the list of bad blocks:

```
U-Boot# nand bad
```

NOTE

*The user marked bad blocks can be viewed by using this command only after a reset.

Erasing NAND

To erase NAND blocks in a particular the address range or using block numbers:

```
U-Boot# nand erase <start offset addr> <len>
```

NOTE

*start offset addr & len fields should align to 0x20000 (64*2048) bytes, i.e.to block size 128KB.

This commands skips bad blocks (both factory and user marked) encountered within the specified range.

For example, to erase blocks 32 through 34:

```
U-Boot# nand erase 0x00400000 0x40000
```

NAND ECC algorithm selection

NAND flash memory, although cheap, suffers from problems like bit flipping which lead to data corruption. However by making use of some error correction coding (ECC) techniques it is possible to work around this problem.

Prior to the AM335xPSP_04.06.00.09-rc2 release, for the data stored in NAND flash, U-Boot supports following NAND ECC schemes

1. S/W ECC (Hamming code)
2. H/W ECC (Hamming code, BCH8)

Starting with the 04.06.00.09-rc2 release only BCH8 is supported, and is used by default in all cases. No user interaction is required to select the algorithm and locations where the **nandecc** command are required can be seen by looking at the history of this page.

BCH Flash OOB Layout

For any ECC scheme we need to add some extra data while writing so as to detect and correct (if possible) the errors introduced by the NAND part. In case of BCH scheme some bytes are needed to store the ECC related info.

The section of NAND memory where addition info like ECC data is stored is referred to as Out Of Band or OOB section.

The first 2 bytes are used for Bad block marker – 0xFFFF => Good block

The next 'N' bytes is used for BCH bytes

$$N = B * \text{<Number of 512-byte sectors in a page>}$$

1. B = 8 bytes per 512 byte sector in BCH4
2. B = 14 bytes per 512 byte sector in BCH8
3. B = 26 bytes per 512 byte sector in BCH16

So for a 2k page-size NAND flash with 64-byte OOB size, we will use BCH8. This will consume 2 + (14*4) = 58 bytes out of 64 bytes available.

The NAND flash part used in EVM does not have enough spare area to support BCH16.

ECC Schemes and their context of usage

ECC type	Usage
S/W ECC	Not used
H/W ECC - Hamming Code	Should use this scheme only for flashing the U-Boot ENV variables.
H/W ECC – BCH8	Should use this scheme while flashing any image/binary other than the U-Boot ENV variables.

To select ECC algorithm for NAND:

```
U-Boot# nandecc [sw | hw <hw_type>]
```

Usage:

sw - Set software ECC for NAND hw <hw_type> - Set hardware ECC for NAND <hw_type> - 0 for Hamming code 1 for bch4 2 for bch8 3 for bch16 Currently we support only Software, Hamming Code and BCH8. We do not support BCH4 and BCH16

ECC schemes usage table

ECC schemes usage table

Component	Default ECC scheme used by the component	ECC scheme to be used to flash the component	ECC schemes supported by the component
SPL	BCH8	BCH8	BCH8
U-boot	Hamming	BCH8	Hamming/BCH8
Linux	BCH8	BCH8	BCH8
File System	NA	BCH8	NA
Environment variables	NA	Hamming	NA

Flashing Kernel

TFTP the kernel uImage to DDR.

```
U-Boot# tftp 0x82000000 <kernel_image>
```

Now flash the kernel image to NAND at the appropriate offset (refer to NAND layout section for the offsets)

```
U-Boot# nand erase 0x00280000 0x00500000
U-Boot# nand write 0x82000000 0x00280000 0x500000
```

NOTE
*Image_size should be aligned to page size of 2048 (0x800) bytes

UBIFS file system flashing

In AM335X, UBIFS file system is used in NAND flash as it is next generation flash file system.

1. Creating and Flashing of UBIFS file system image is described over here

NOTE
In case of AM335x, file system partition is starting from 0x780000. So flashing offset for file system in U-Boot is 0x780000 and from Linux MTD partition number 7 should used for flashing file file system.

UART

This section describes how to use UART boot mode using TeraTerm.

Boot Over UART

NOTE

*The release package does not contain the binary for UART boot. Please follow the steps mentioned here for compiling u-boot and use the *spl/u-boot-spl.bin* file that is produced.

1. Switch ON EVM with switch settings for UART boot. When “CCCC” characters appear on TeraTerm window, from the File Menu select Transfer --> XMODEM --> Send (1K mode)
2. Select “u-boot-spl.bin” for the transfer
3. After image is successfully downloaded, the ROM code will boot it.
4. When “CCCC” characters appear on TeraTerm window, from the File Menu select Transfer --> YMODEM --> Send (1K mode)
5. Select “u-boot.img” for the transfer
6. After image is successfully downloaded, U-Boot will boot it.
7. Hit enter and get to u-boot prompt “U-Boot# ”

Flashing images to NAND in UART boot mode

Boot using UART boot mode as here

After the U-Boot prompt **U-Boot#** comes up, the images for the 1st stage and 2nd stage can be flashed to NAND for persistent storage.

Flashing SPL to NAND in UART boot mode

Flash SPL (**MLO**) to NAND by executing the following commands:

```
U-Boot# loadb 0x82000000
```

- From TeraTerm Menu click “**File -> Transfer -> Kermit -> Send**”.
- Select the 1st stage u-boot image “**MLO**” and click “**OPEN**” button
- Wait for download to complete and then run following commands in u-boot prompt

```
U-Boot# nand erase 0x0 0x20000
U-Boot# nand write 0x82000000 0x0 0x20000
```

If no error messages are displayed the SPL of NAND boot has been successfully transferred to NAND.

Flashing U-Boot to NAND in UART boot mode

Flash the 2nd stage U-Boot (**u-boot.img**) to NAND by executing the following commands:

```
U-Boot# loadb 0x82000000
```

- From TeraTerm Menu click “**File -> Transfer -> Kermit -> Send**”.
- Select the 2nd stage u-boot image “**u-boot.img**” and click “**OPEN**” button
- Wait for download to complete and then run following commands in U-Boot prompt

```
U-Boot# nand erase 0x80000 0x40000
U-Boot# nand write 0x82000000 0x80000 0x40000
```

If no error messages are displayed the U-boot of NAND boot has been successfully transferred to NAND.

SD (Secured Digital card)

This section gives an overview of the SD support in U-Boot

Read and execute uImage from SD card

Please note that the following commands are being executed from the 2nd stage. List files on a FAT32 formatted SD card

```
U-Boot# mmc rescan
U-Boot# fatls mmc 0
```

Booting kernel image from the SD card

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 uImage
U-Boot# bootm 0x82000000
```

Read and execute u-boot from SD card

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 u-boot.bin
U-Boot# go 0x82000000
```

Setting Up Boot Environment on SD Card

This section describes steps to be followed to create a standalone power-on bootable system on SD card.

Prerequisites Ensure that following is available:

- A Linux host with fdisk, sfdisk, mkfs.ext3 and mkfs.vfat utilities is available
- Copy images MLO, u-boot.img, uImage, nfs.tar.gz and mkspd-am335x.sh from release package to a directory on this Linux machine. For subsequent description, we will assume these files are copied to /home/am335x. Please refer "Package Contents" section in AM335x PSP User's Guide for location of these files.
- Empty SD card (at least 256MB, preferably 4GB SDHC)
- A SD memory card reader/programmer to copy files from Linux Host

NOTE

*The SD Boot has some specific restriction about the format of the 1st partition and copying the MLO image.

- So it is recommended that the supplied script mkspd-am335x.sh is used for creating partitions and copying files.

Steps

- Connect the SD memory card using Memory Card reader to the Linux Host
- Note the name allotted for this device. Type "dmesg". The SD/MMC card name should show up near the end, usually something like "SDC" (/dev/sdc) or "SDD" (/dev/sdd).
- Navigate to the /home/am335x directory where all the mentioned files are copied
- Ensure that the script mkspd-am335x.sh has executable permissions
- This scripts expects arguments in the following format

```
./mkspd-am335x.sh <sd-device-name> <sd-1st-stage-bootloader> <sd-2nd-stage-bootloader> <kernel-uImage>
<filesystem>
```

NOTE

that the user will require root/sudo permissions

- In our example, we run the following command

```
sudo ./mkspd-am335x.sh /dev/sdd MLO u-boot.img uImage nfs.tar.gz
```

- You will be asked about data getting overwritten, confirm it and the files along with filesystem will be copied to SD card
- Note that this script will create two primary partitions:
 - 1st partition is formatted as FAT32 containing MLO, u-boot.img, uImage files
 - 2nd partition is formatted as ext3 where the filesystem is extracted in root

Boot using SD card

Once the SD card has been setup as described in the previous section make sure the switch settings are set for SD boot mode and then plug in the SD card in the MMC/SD card slot on the EVM.

Flashing images to NAND in SD boot

Boot using SD boot mode as here.

After the 2nd stage prompt **U-Boot#** comes up, the images for the 1st stage and 2nd stage can be flashed to NAND for persistent storage.

Flashing SPL to NAND in SD boot

Flash SPL (**MLO**) to NAND by executing the following commands:

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 MLO

U-Boot# nand erase 0x0 0x20000
U-Boot# nand write 0x82000000 0x0 0x20000
```

If no error messages are displayed the SPL of NAND boot has been successfully transferred to NAND.

Flashing U-Boot to NAND in SD boot

Flash the 2nd stage U-Boot (**u-boot.img**) to NAND by executing the following commands:

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 u-boot.img
U-Boot# nand erase 0x80000 0x40000
U-Boot# nand write 0x82000000 0x80000 0x40000
```

If no error messages are displayed the U-boot of NAND boot has been successfully transferred to NAND.

Setting U-Boot environment using uEnv.txt

U-Boot environment variables can be modified using a plain text file named uEnv.txt. The scripts can be used to modify and even over-ride the various parameters like bootargs, TFTP serverip etc. If a command named uenvcmd is defined in the file it will be executed. uEnv.txt can be loaded from SD card and tftp server.

Example text file named *uEnv.txt*

IMPORTANT

*The uEnv.txt file should be in **unix** format. Also make sure that there is **an empty line** at the end of the file.

```
bootargs=console=ttyO0,115200n8 root=/dev/mmcblk0p2 mem=128M rootwait
bootcmd=mmc rescan; fatload mmc 0 0x82000000 uImage; bootm 0x82000000
uenvcmd=boot
```

The file *uEnv.txt* is automatically loaded from SD if the bootcmd is run. It can be loaded and put into the environment manually.

Making use pre-existing uEnv on SD card

uEnv.txt on an SD card can be used to override the env settings saved on a persistent storage like NAND by making use of the following commands

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x81000000 uEnv.txt
U-Boot# env import -t 0x81000000 $filesize
U-Boot# boot
```

SPI

Note

- This feature is not supported prior to PSP 04.06.00.08 (and AMSDK 05.05.00.00).
- This feature changed kernel location and filenames with PSP 04.06.00.09 (and AMSDK 05.06.00.00).
- The release package does not contain the binary for SPI boot. Please follow the steps mentioned here for compiling u-boot and use the *MLO.spi* and *u-boot.bin* files that are produced.
- Following those same instructions but building for **am335x_evm_spiboot** rather than **am335x_evm** will result in binaries that will use the SPI flash for environment rather than NAND.

In this example we initially boot from an SD card and use that to transfer the files to write to SPI flash. If loading via other methods, modify the commands below.

1. Switch ON EVM with switch settings such that SPI is present and boot via non-SPI. See SPI boot for more information.
2. Write it to SPI memory by executing the following commands:

```
U-Boot# sf probe 0
U-Boot# sf erase 0 +E0000
U-Boot# mmc rescan
U-Boot# fatload mmc 0 ${loadaddr} MLO.byteswap
U-Boot# sf write ${loadaddr} 0 ${filesize}
U-Boot# fatload mmc 0 ${loadaddr} u-boot.img
U-Boot# sf write ${loadaddr} 0x80000 ${filesize}
```

CPSW Ethernet

Note

- This feature is not supported prior to PSP 04.06.00.08 (and AMSDK 05.05.00.00).
- The release package does not contain the binary for CPSW ethernet boot. Please follow the steps mentioned here for compiling u-boot and use the *spl/u-boot-spl.bin* and *u-boot.img* files that are produced.

Booting Over CPSW Ethernet

- Configure DHCPd and tftpd on your host.
- Within the dhcpd configuration, add entries to send the *u-boot-spl.bin* or *u-boot.img* files based on the vendor-class-identifier field. For ISC dhcpd an example host entry looks like this:

```
host am335x_evm {
    hardware ethernet de:ad:be:ee:ee:ef;
```

```

if substring (option vendor-class-identifier, 0, 10) = "DM814x ROM" {
    filename "u-boot-spl.bin";
} elsif substring (option vendor-class-identifier, 0, 17) = "AM335x U-Boot SPL" {
    filename "u-boot.img";
} else {
    filename "uImage-am335x";
}
}
}

```

- Copy the *u-boot-spl.bin* and *u-boot.img* files you have build to the directory tftpd serves files from.
- Switch ON EVM with switch settings for CPSW ethernet boot.
- Hit enter and get to u-boot prompt "U-Boot# "

Flashing in CPSW Ethernet boot mode

It is possible to build a version of U-Boot that boots over the cpsw ethernet interface and will automatically load and execute a script file. This can be used in initial programming or restoring of boards. It follows the general principles outlined above. In this case first you must build for **am335x_evm_restore_flash** rather than **am335x_evm** when building U-Boot. Next, you will need to create the script file that is run. There are examples provided in the source tree at **doc/am335x.net-spl/debrick-nand.txt** and **doc/am335x.net-spl/debrick-spi.txt**. You should copy one of these files and modify for your exact situation. Once you have modified the script you will need to turn it into a scr file using the following command:

```
./tools/mkimage -A arm -O U-Boot -C none -T script -d <your script> debrick.scr
```

and copy the resulting debrick.scr file to the location tftpd serves files out of. For more information please see the **doc/am335x.net-spl/README** file in the source tree.

U-Boot Network configuration

In order to download the Linux kernel image from the TFTP server and for mounting NFS the network settings in U-Boot need to be configured.

When booting for the first time, U-Boot tries to fetch the MAC address from the env space. If it returns empty, it will look for MAC address from the eFuse registers in the Control module space and set the "ethaddr" variable in the env appropriately. The ethaddr can also be set using the setenv/saveenv commands. In such cases the user-set MAC address will take effect on subsequent reboot only.

To set a different MAC address use the following command

```
U-Boot# set ethaddr <random MAC address eg- 08:11:23:32:12:77>
```

NOTE

*When setting a MAC address please ensure that the LSB of the 1st byte if not 1 i.e. when setting the MAC address: **y** in **xy:ab:cd:ef:gh:jk** has to be an even number. For more info this refer to the wiki page http://en.wikipedia.org/wiki/MAC_address

In case a static ip is not available run the dhcp command to obtain the ip address from the DHCP server on the network to which the EVM is connected.

```

U-Boot# setenv serverip <tftp server in your network>
U-Boot# netmask 255.255.255.0
U-Boot# dhcp
U-Boot# saveenv

```


In case a static ip is available run the following commands

```
U-Boot# setenv ipaddr <your static ip>
U-Boot# saveenv
```

This completes the network configuration in U-Boot.

U-Boot Environment Variables

After completing the network configuration and flashing the kernel image and filesystems to flash you need to set some other parameters which are essential for booting the kernel. We make use of a number of existing helper variables that exist in the environment at present. To see what they are set to use the *printenv* command. After setting **bootargs** along with any other variables for your needs you will need to do:

```
U-Boot# saveenv
```

In all cases we make use of **optargs** to control passing in of additional arguments and **ip_method** to determine how the kernel will deal with networking PRIOR to userspace spawning init. This does not control for example if a dhcp client will be started as part of the userspace init sequence.

Environment Settings for Ramdisk

In case you are using a RAMDISK as the Linux filesystem:

```
U-Boot# setenv bootargs ${console} ${optargs} root=/dev/ram rw initrd=${loadaddr},32MB ip=${ip_method}
```

For booting from NAND:

```
U-Boot# setenv nand_src_addr 0x00280000
U-Boot# setenv nand_img_siz 0x170000
U-Boot# setenv initrd_src_addr 0x00780000
U-Boot# setenv initrd_img_siz 0x320000
U-Boot# setenv bootcmd 'nand read ${kloadaddr} ${nand_src_addr}
${nand_img_siz};nand read ${loadaddr} ${initrd_src_addr}
${initrd_img_siz};bootm ${kloadaddr}'
```

For booting from SD card:

```
U-Boot# setenv bootcmd 'mmc rescan;run mmc_load_uimage;fatload mmc ${mmc_dev} ${loadaddr} initrd.ext3.gz;bootm ${kloadaddr}'
```

NOTE

*The sizes of images mentioned in the above commands have to be modified based on the actual image size. Also, it should be aligned to sector size of the flash device used.

Environment Settings for UBIFS Filesystem

- The U-Boot environment variable **bootargs** contains arguments to be passed to the Linux kernel. The **bootargs** variable here makes use of the **nand_root** variable. The typical format of this variable for a UBIFS root filesystem is:

```
root=ubi0:<VOLUME NAME> ubi.mtd=<PARTITION_ID>,YYYY rw
```

The value of **PARTITION_ID** depends on MTD device which holds the rootfs, **YYYY** depends on the page size of the partition and **VOLUME NAME** depends on the volume name given in ubinize.cfg file while creating UBIFS image as described here . In cases where you have multiple UBI volumes, ubi0 would change to the volume with the root partition.

Once **nand_root** is set:

```
U-Boot# setenv bootcmd run nand_boot
```

Environment Settings for jffs2 Filesystem

The **bootargs** variable here makes use of the **nand_root** variable. The root file system format is jffs2:

Enabling and using the jffs2 as a root file system is describe here ^[4]

Environment Settings for NFS Filesystem

NOTE

*When setting a MAC address please ensure that the LSB of the 1st byte if not 1 i.e. when setting the MAC address: **y** in **xy:ab:cd:ef:gh:jk** has to be an even number. For more info this refer to the wiki page http://en.wikipedia.org/wiki/MAC_address

- If you need to use a separate server for TFTP and NFS please see how to use the **next-server** option in your DHCP server.

Modify the required variables:

```
U-Boot# print ethaddr                <-- Check if MAC address is assigned and is unique
U-Boot# setenv ethaddr <unique-MAC-address>  <-- Set only if not present already, format uv:yy:zz:aa:bb:cc
U-Boot# setenv serverip <NFS and TFTP server-ip>
U-Boot# setenv rootpath /location/of/nfsroot/export
U-Boot# setenv bootcmd net_boot
```

Booting the kernel

In case everything went well, boot the kernel using the following command.

```
U-Boot# boot
```

Article Sources and Contributors

Sitara Linux Software Developer's Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=137600> *Contributors:* Bshay, Cem8101, DaveC, Fcooper, Gguyotte, Gregturne, Hazeljojo, Jefflance01, Kevinsc, Mike Tadyshak, Orbarron, Pprakash, SchuylerPatton, Swai, TommyG

How to Build a Ubuntu Linux host under VMware *Source:* <http://processors.wiki.ti.com/index.php?oldid=126661> *Contributors:* Alanc, Cem8101, Fcooper, Gregturne, Jefflance01, Kevinsc, Pprakash, SiddharthHeroor

Sitara SDK Installer *Source:* <http://processors.wiki.ti.com/index.php?oldid=114236> *Contributors:* Cem8101, Kevinsc, Mike Tadyshak

Sitara Linux SDK Setup Script *Source:* <http://processors.wiki.ti.com/index.php?oldid=137370> *Contributors:* Fcooper, Kevinsc

AMSDK START HERE Script *Source:* <http://processors.wiki.ti.com/index.php?oldid=106916> *Contributors:* Cem8101, Jefflance01

Matrix Users Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=114205> *Contributors:* Alexander.stohr, BernieThompson, Cem8101, ChrisRing, ClayMontgomery, Fcooper, Gregturne, Jefflance01, Kevinsc, Mike Tadyshak, Stevek, Tpothast

Power Management Users Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=136308> *Contributors:* Cem8101, Gguyotte, Kevinsc

ARM Multimedia Users Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=128910> *Contributors:* Cem8101, D-gerlach, Kevinsc, Pprakash

Camera Users Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=114493> *Contributors:* Cem8101, Kevinsc, Pprakash

Cryptography Users Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=136822> *Contributors:* Cem8101, Gregturne, Joelagnel, Kevinsc, Lo

Oprofile User's Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=114234> *Contributors:* Jefflance01, Kevinsc

Open Source Wireless Connectivity WLAN Bluetooth User Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=114572> *Contributors:* A0794974, Cem8101, Jefflance01, Kevinsc

OMAP Wireless Connectivity OpenSource Compat Wireless Build *Source:* <http://processors.wiki.ti.com/index.php?oldid=94045> *Contributors:* A0389760, A0794974, Benburns, Eyalreizer, Moosa

OMAP Wireless Connectivity mac80211 compat wireless implementation Architecture *Source:* <http://processors.wiki.ti.com/index.php?oldid=118818> *Contributors:* A0794974, Eyalreizer, Moosa

OMAP Wireless Connectivity Battleship Game demo *Source:* <http://processors.wiki.ti.com/index.php?oldid=107186> *Contributors:* A0794974, Dmeixner, Moosa

AMSDK u-boot User's Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=136076> *Contributors:* Cem8101, Jefflance01, Mike Tadyshak, Trini

AMSDK Linux User's Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=106920> *Contributors:* Cem8101, Gregturne, Jefflance01

Code Composer Studio v5 Users Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=137210> *Contributors:* Cem8101, Fcooper, Jefflance01, Kevinsc, Mike Tadyshak

Linux Debug in CCSv5 *Source:* <http://processors.wiki.ti.com/index.php?oldid=127228> *Contributors:* Ali Bahar, AndyW, Arnier, John, Jsarao, Kevinsc, Rp, Rjsouza, Stater

How to setup Remote System Explorer plug-in *Source:* <http://processors.wiki.ti.com/index.php?oldid=136444> *Contributors:* Cem8101, ChrisRing, Mike Tadyshak

How to Run GDB on CCSv5 *Source:* <http://processors.wiki.ti.com/index.php?oldid=128743> *Contributors:* Alexander.stohr, Cem8101, Jefflance01, John, Kevinsc, Mike Tadyshak, SchuylerPatton

Pin Mux Utility for ARM MPU Processors *Source:* <http://processors.wiki.ti.com/index.php?oldid=136179> *Contributors:* Alexander.stohr, Cem8101, Erives, Kevinsc, Kevinvap, Mike Tadyshak, Nfatemi

Pin Setup Tool for AM18xx ARM Microprocessors *Source:* <http://processors.wiki.ti.com/index.php?oldid=114529> *Contributors:* Benburns, Jefflance01, Kevinsc, Mike Tadyshak

AM335x Flashing Tools Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=89304> *Contributors:* Cem8101, Gururaja, Kevinsc, Rachna, SekharNori, Sudhakar.raj, X0155329

Flash v1.6 User Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=114147> *Contributors:* Kevinsc, Mike Tadyshak

AM18x Flash Tool User's Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=106925> *Contributors:* Alanc, Gguyotte, Jeff Cobb, Jefflance01, Kevinsc, Mike Tadyshak, SchuylerPatton

AMSDK File System Optimization/Customization *Source:* <http://processors.wiki.ti.com/index.php?oldid=114466> *Contributors:* Cem8101, Jefflance01, Kevinsc

Sitara Linux Training *Source:* <http://processors.wiki.ti.com/index.php?oldid=117278> *Contributors:* Cem8101, Kevinsc, SchuylerPatton

How to use a Mouse instead of the Touchscreen with Matrix *Source:* <http://processors.wiki.ti.com/index.php?oldid=114469> *Contributors:* ClayMontgomery, Jefflance01, Kevinsc

How to Recalibrate the Touchscreen *Source:* <http://processors.wiki.ti.com/index.php?oldid=126848> *Contributors:* Cem8101, Gguyotte, Jefflance01, Kevinsc

AM335x U-Boot User's Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=136084> *Contributors:* Akshay.s, Cem8101, Gururaja, Jinleileiking, Rachna, Trini, VaibhavBedia, X0155329

Sitara Linux SDK Top-Level Makefile *Source:* <http://processors.wiki.ti.com/index.php?oldid=114508> *Contributors:* Cem8101, Jefflance01, Kevinsc

Sitara Linux SDK GCC Toolchain *Source:* <http://processors.wiki.ti.com/index.php?oldid=117497> *Contributors:* Cem8101, Jefflance01, Kevinsc

Sitara Linux SDK create SD card script *Source:* <http://processors.wiki.ti.com/index.php?oldid=114505> *Contributors:* Cem8101, Jefflance01, Kevinsc

How to add a JVM *Source:* <http://processors.wiki.ti.com/index.php?oldid=114524> *Contributors:* Jefflance01, Kevinsc

AM335x U-Boot User's Guide *Source:* <http://processors.wiki.ti.com/index.php?oldid=136084> *Contributors:* Akshay.s, Cem8101, Gururaja, Jinleileiking, Rachna, Trini, VaibhavBedia, X0155329

Image Sources, Licenses and Contributors

Image:TIBanner.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:TIBanner.png> *License:* unknown *Contributors:* Nsnehaprabha

Image:Sitara linux stack.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Sitara_linux_stack.png *License:* unknown *Contributors:* Hazeljojo, Kevinse

Image:Evm environ.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Evm_envIRON.png *License:* unknown *Contributors:* Kevinse

Image:Bone environ.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Bone_envIRON.png *License:* unknown *Contributors:* Kevinse

Image:Sitara-Linux-directory-structure.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-directory-structure.png> *License:* unknown *Contributors:* Cem8101

File:Ubuntu 10 04 4.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Ubuntu_10_04_4.png *License:* unknown *Contributors:* Fcooper

Image:VMwareNetwork.PNG *Source:* <http://processors.wiki.ti.com/index.php?title=File:VMwareNetwork.PNG> *License:* unknown *Contributors:* Gregturne

Image:Vmware sh1.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Vmware_sh1.jpg *License:* unknown *Contributors:* Kevinse

Image:Vmware sh2.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Vmware_sh2.jpg *License:* unknown *Contributors:* Kevinse

Image:Sitara-Linux-VMWare-removable-devices.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-VMWare-removable-devices.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-VMWare-removable-devices2.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-VMWare-removable-devices2.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-VMWare-connect-beaglebone.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-VMWare-connect-beaglebone.png> *License:* unknown *Contributors:* Cem8101

Image:Vmware sh4.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Vmware_sh4.jpg *License:* unknown *Contributors:* Kevinse

Image:Vmware sh6.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Vmware_sh6.jpg *License:* unknown *Contributors:* Kevinse

Image:Vmware sh7.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Vmware_sh7.jpg *License:* unknown *Contributors:* Kevinse

Image:Vmware sh8.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Vmware_sh8.jpg *License:* unknown *Contributors:* Kevinse

Image:Network proxy setup.JPG *Source:* http://processors.wiki.ti.com/index.php?title=File:Network_proxy_setup.JPG *License:* unknown *Contributors:* Kevinse

Image:Network proxy preferences.JPG *Source:* http://processors.wiki.ti.com/index.php?title=File:Network_proxy_preferences.JPG *License:* unknown *Contributors:* Kevinse

Image:Network proxy password.JPG *Source:* http://processors.wiki.ti.com/index.php?title=File:Network_proxy_password.JPG *License:* unknown *Contributors:* Kevinse

Image:Vmware kb1.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Vmware_kb1.jpg *License:* unknown *Contributors:* Kevinse

Image:Vmware kb2.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Vmware_kb2.jpg *License:* unknown *Contributors:* Kevinse

Image:Vmware kb3.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Vmware_kb3.jpg *License:* unknown *Contributors:* Kevinse

Image:VMWareDisk.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:VMWareDisk.png> *License:* unknown *Contributors:* Gregturne

Image:AM37x Env ov 1 KC.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:AM37x_Env_ov_1_KC.jpg *License:* unknown *Contributors:* SchuylerPatton

Image:AM37x Sdk dir selection.JPG *Source:* http://processors.wiki.ti.com/index.php?title=File:AM37x_Sdk_dir_selection.JPG *License:* unknown *Contributors:* SchuylerPatton

Image:Matrix 2 Main Menu.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Matrix_2_Main_Menu.png *License:* unknown *Contributors:* Fcooper

Image:Screenshot-2.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Screenshot-2.png> *License:* unknown *Contributors:* Fcooper

Image:main_screen.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Main_screen.png *License:* unknown *Contributors:* Gguyotte

Image:Power_screen.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Power_screen.png *License:* unknown *Contributors:* Gguyotte

Image:mpeg4aac.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Mpeg4aac.png> *License:* unknown *Contributors:* Pprakash

Image:mpeg4.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Mpeg4.png> *License:* unknown *Contributors:* Pprakash

Image:h264.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:H264.png> *License:* unknown *Contributors:* Pprakash

Image:aac.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Aac.png> *License:* unknown *Contributors:* Pprakash

File:SDKMMFwk.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:SDKMMFwk.png> *License:* unknown *Contributors:* Pprakash

File:MMFwk.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:MMFwk.png> *License:* unknown *Contributors:* Pprakash

File:gstBuildDepandancies.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:GstBuildDepandancies.png> *License:* unknown *Contributors:* Pprakash

File:NeonPerf.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:NeonPerf.png> *License:* unknown *Contributors:* Pprakash

File:AM37xx_L1-3M02_Single_Camera_Approach.png *Source:* http://processors.wiki.ti.com/index.php?title=File:AM37xx_L1-3M02_Single_Camera_Approach.png *License:* unknown *Contributors:* Pprakash

Image:MatrixMain.PNG *Source:* <http://processors.wiki.ti.com/index.php?title=File:MatrixMain.PNG> *License:* unknown *Contributors:* Gregturne

Image:MatrixCryptos.PNG *Source:* <http://processors.wiki.ti.com/index.php?title=File:MatrixCryptos.PNG> *License:* unknown *Contributors:* Gregturne

Image:SecureServer.PNG *Source:* <http://processors.wiki.ti.com/index.php?title=File:SecureServer.PNG> *License:* unknown *Contributors:* Gregturne

Image:Oprofile.PNG *Source:* <http://processors.wiki.ti.com/index.php?title=File:Oprofile.PNG> *License:* unknown *Contributors:* Jefflance01

Image:HomepageIcon.jpg *Source:* <http://processors.wiki.ti.com/index.php?title=File:HomepageIcon.jpg> *License:* unknown *Contributors:* A0794974

File: BattleshipIcon.jpg *Source:* <http://processors.wiki.ti.com/index.php?title=File:BattleshipIcon.jpg> *License:* unknown *Contributors:* Moosa

File:WFD_Main.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:WFD_Main.jpg *License:* unknown *Contributors:* Moosa

File:WDF_Settings.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:WDF_Settings.jpg *License:* unknown *Contributors:* Moosa

File: BattleshipMain.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:BattleshipMain.png> *License:* unknown *Contributors:* Moosa

File: BattleshipHost.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:BattleshipHost.png> *License:* unknown *Contributors:* Moosa

File: BattleshipConnectGame.jpg *Source:* <http://processors.wiki.ti.com/index.php?title=File:BattleshipConnectGame.jpg> *License:* unknown *Contributors:* Moosa

Image:Download-page.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Download-page.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Sitara-Linux-CCS-Install-SDK.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-Install-SDK.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-Install-License.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-Install-License.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-Install-Location.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-Install-Location.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-Install-Processor.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-Install-Processor.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-Install-Components.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-Install-Components.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-Install-Emulator.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-Install-Emulator.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-Install-Options.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-Install-Options.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-icon.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-icon.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-splash-screen.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-splash-screen.png> *License:* unknown *Contributors:* Cem8101

Image:Workspace-Launcher.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Workspace-Launcher.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Sitara-Linux-CCS-resource-explorer.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-resource-explorer.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-project-explorer.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-project-explorer.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-window-preferences.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-window-preferences.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-general-capabilities.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-general-capabilities.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-enable-rse.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-enable-rse.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-open-perspective.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-open-perspective.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-import-qt.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-import-qt.png> *License:* unknown *Contributors:* Cem8101

Image:Select-Qt-Project-File.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Select-Qt-Project-File.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Qt-Import-Wizard.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Qt-Import-Wizard.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Sitara-Linux-CCS-matrix-browser.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-matrix-browser.png> *License:* unknown *Contributors:* Cem8101

Image:Preferences-1.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Preferences-1.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:New-Qt-Version.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:New-Qt-Version.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Sitara-Linux-CCS-make-target.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-make-target.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-release-build.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-release-build.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-matrix-build.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-matrix-build.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-make-install.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-make-install.png> *License:* unknown *Contributors:* Cem8101

Image:Import C projects-1.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Import_C_projects-1.png *License:* unknown *Contributors:* Mike Tadyshak

Image:Sitara-Linux-CCS-import-c.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-import-c.png> *License:* unknown *Contributors:* Cem8101

Image:Example-applications.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Example-applications.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Import-2.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Import-2.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Import-Qt.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Import-Qt.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Projects-imported.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Projects-imported.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Sitara-Linux-CCS-new-c-project.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-new-c-project.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-cross-compile.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-cross-compile.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-command-setup.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-command-setup.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-select-configurations.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-select-configurations.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-empty-helloworld.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-empty-helloworld.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-helloworld-c-file.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-helloworld-c-file.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-helloworld.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-helloworld.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-helloworld-built.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-helloworld-built.png> *License:* unknown *Contributors:* Cem8101

Image:CCS-Sitara ConsoleTab.png *Source:* http://processors.wiki.ti.com/index.php?title=File:CCS-Sitara_ConsoleTab.png *License:* unknown *Contributors:* Fcooper

Image:CCS-Sitara ProblemsTag.png *Source:* http://processors.wiki.ti.com/index.php?title=File:CCS-Sitara_ProblemsTag.png *License:* unknown *Contributors:* Fcooper

Image:Linux debug v5 GDB config.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_GDB_config.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 tab main.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_tab_main.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 tab dbg main.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_tab_dbg_main.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 ifconfig.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_ifconfig.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 tab dbg connection.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_tab_dbg_connection.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 gdbserver.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_gdbserver.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 debugger.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_debugger.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 kernel pj1 wizard.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_kernel_pjt_wizard.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 kernel pj1 new.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_kernel_pjt_new.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 kernel pj1 tree.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_kernel_pjt_tree.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 kernel build set.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_kernel_build_set.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 jtag tab main.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_jtag_tab_main.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 jtag target assign.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_jtag_target_assign.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 jtag vmlinux.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_jtag_vmlinux.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 jtag debugger launching.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_jtag_debugger_launching.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 jtag debugger.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_jtag_debugger.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 mixed app startup.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_mixed_app_startup.png *License:* unknown *Contributors:* Rsjsouza

Image:Linux debug v5 mixed kernel halted.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Linux_debug_v5_mixed_kernel_halted.png *License:* unknown *Contributors:* Rsjsouza

Image:Sitara-Linux-CCS-rse-perspective.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-rse-perspective.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-rse-view.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-rse-view.png> *License:* unknown *Contributors:* Cem8101

Image:New-connection.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:New-connection.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Remote-system-type.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Remote-system-type.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:X-New Connection.png *Source:* http://processors.wiki.ti.com/index.php?title=File:X-New_Connection.png *License:* unknown *Contributors:* Mike Tadyshak

Image:Ssh-files.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Ssh-files.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Processes .png *Source:* http://processors.wiki.ti.com/index.php?title=File:Processes_.png *License:* unknown *Contributors:* Mike Tadyshak

Image:Shells.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Shells.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Terminals.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Terminals.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Sitara-Linux-CCS-target-view.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-target-view.png> *License:* unknown *Contributors:* Cem8101

Image:Sitara-Linux-CCS-c-view.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-CCS-c-view.png> *License:* unknown *Contributors:* Cem8101

Image:Show-view-remote-systems.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Show-view-remote-systems.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:X-change-user-id.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:X-change-user-id.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Setup-ssh-editted-1.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Setup-ssh-editted-1.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Nasty-Warning-2.PNG *Source:* <http://processors.wiki.ti.com/index.php?title=File:Nasty-Warning-2.PNG> *License:* unknown *Contributors:* Mike Tadyshak

Image:X-network-Connections-top.jpeg *Source:* <http://processors.wiki.ti.com/index.php?title=File:X-network-Connections-top.jpeg> *License:* unknown *Contributors:* Mike Tadyshak

Image:X-network-Connections-bottom.jpeg *Source:* <http://processors.wiki.ti.com/index.php?title=File:X-network-Connections-bottom.jpeg> *License:* unknown *Contributors:* Mike Tadyshak

Image:X-login.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:X-login.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Expand-root-small.jpeg *Source:* <http://processors.wiki.ti.com/index.php?title=File:Expand-root-small.jpeg> *License:* unknown *Contributors:* Mike Tadyshak

Image:MyTerminalView-small.jpeg *Source:* <http://processors.wiki.ti.com/index.php?title=File:MyTerminalView-small.jpeg> *License:* unknown *Contributors:* Mike Tadyshak

Image:MyShellView-small.jpeg *Source:* <http://processors.wiki.ti.com/index.php?title=File:MyShellView-small.jpeg> *License:* unknown *Contributors:* Mike Tadyshak

Image:MyProcessList-small.jpeg *Source:* <http://processors.wiki.ti.com/index.php?title=File:MyProcessList-small.jpeg> *License:* unknown *Contributors:* Mike Tadyshak

Image:Initial-debug-configurations.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Initial-debug-configurations.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Dhrystone-debug-config1.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Dhrystone-debug-config1.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Debug-config-2.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Debug-config-2.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Auto-debug-config-main-tab.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Auto-debug-config-main-tab.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Gdbinit.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Gdbinit.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Browse-to-dhrystone-with-rse.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Browse-to-dhrystone-with-rse.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Permissions.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Permissions.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Auto-debugging.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Auto-debugging.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Shell-processes.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Shell-processes.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Kill-gdbserver.png *Source:* <http://processors.wiki.ti.com/index.php?title=File:Kill-gdbserver.png> *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v1.0 Installer Start.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v1.0_Installer_Start.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v1.0 Installer Remove-2.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v1.0_Installer_Remove-2.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility Uninstall.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_Uninstall.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v1.0 Installer Complete.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v1.0_Installer_Complete.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v1.0 TI Icon.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v1.0_TI_Icon.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 Device Type Selection.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_Device_Type_Selection.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:PinMuxUtility_save_design.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:PinMuxUtility_save_design.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v1.0 AM37xx Open Design.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v1.0_AM37xx_Open_Design.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 Legend 2.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_Legend_2.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 Legend.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_Legend.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx Conflict PI Grid.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_Conflict_PI_Grid.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx I2C3 Conflict 2.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_I2C3_Conflict_2.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx UART4 Conflict.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_UART4_Conflict.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx I2C3 Fixed 2.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_I2C3_Fixed_2.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx Multi Muxed Conflict PI Grid.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_Multi_Muxed_Conflict_PI_Grid.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx I2C3 Multi Muxed Conflict.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_I2C3_Multi_Muxed_Conflict.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx SPI1 IO Set Match PI Grid.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_SPI1_IO_Set_Match_PI_Grid.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx SPI1 IO Set Subset PI Grid.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_SPI1_IO_Set_Subset_PI_Grid.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx SPI1 IO Set Violation PI Grid.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_SPI1_IO_Set_Violation_PI_Grid.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:PinMuxUtility Voltages.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:PinMuxUtility_Voltages.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx MMC0 IO Power.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_MMC0_IO_Power.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v2.1 AM35xx MMC0 IO Power Violation PI Grid.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v2.1_AM35xx_MMC0_IO_Power_Violation_PI_Grid.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:PinMuxUtility MMC0 IO Power Violation.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:PinMuxUtility_MMC0_IO_Power_Violation.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:PinMuxUtility Pad Config Selected.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:PinMuxUtility_Pad_Config_Selected.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:PinMuxUtility Pad Config Editor.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:PinMuxUtility_Pad_Config_Editor.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:PinMuxUtility Pad Config Changed.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:PinMuxUtility_Pad_Config_Changed.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v1.0 AM37xx Save Device Header 3.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v1.0_AM37xx_Save_Device_Header_3.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v1.0 AM37xx Save Board Header 4.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v1.0_AM37xx_Save_Board_Header_4.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v1.0 AM37xx Device Header File 4.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v1.0_AM37xx_Device_Header_File_4.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v1.0 AM37xx Board Header File 4.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v1.0_AM37xx_Board_Header_File_4.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Pin Mux Utility v1.0 AM37xx Modified Board Header File.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Pin_Mux_Utility_v1.0_AM37xx_Modified_Board_Header_File.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Bin folder.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Bin_folder.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Program Startup.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Program_Startup.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Collision Warning.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Collision_Warning.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:After Selecting UART0.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:After_Selecting_UART0.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:After Not Selecting UART0.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:After_Not_Selecting_UART0.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Saved Header File.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Saved_Header_File.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Find Pin.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Find_Pin.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Find Pin Results.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Find_Pin_Results.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Flash v1.6.0.0 Startup.PNG *Source:* http://processors.wiki.ti.com/index.php?title=File:Flash_v1.6.0.0_Startup.PNG *License:* unknown *Contributors:* Mike Tadyshak

Image:Evm revg reset.JPG *Source:* http://processors.wiki.ti.com/index.php?title=File:Evm_revg_reset.JPG *License:* unknown *Contributors:* Gguyotte

Image:Bootmodeswitches.jpg *Source:* <http://processors.wiki.ti.com/index.php?title=File:Bootmodeswitches.jpg> *License:* unknown *Contributors:* Mike Tadyshak

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED. BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

License

1. Definitions

- "Adaptation"** means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.
- "Collection"** means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined below) for the purposes of this License.
- "Creative Commons Compatible License"** means a license that is listed at <http://creativecommons.org/compatibility/licenses> that has been approved by Creative Commons as being essentially equivalent to this License, including, at a minimum, because that license: (i) contains terms that have the same purpose, meaning and effect as the License Elements of this License; and, (ii) explicitly permits the relicensing of adaptations of works made available under that license under this License or a Creative Commons jurisdiction license with the same License Elements as this License.
- "Distribute"** means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.
- "License Elements"** means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.
- "Licensor"** means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.
- "Original Author"** means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.
- "Work"** means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.
- "You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.
- "Publicly Perform"** means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.
- "Reproduce"** means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights

Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant

Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

- to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;
 - to create and to Reproduce Adaptations provided that for each Adaptation You must take reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";
 - to Distribute and Publicly Perform the Work including as incorporated in Collections; and,
 - to Distribute and Publicly Perform Adaptations.
- For the avoidance of doubt:
- Non-waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;
 - Waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,
 - Voluntary License Schemes.** The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. Restrictions

The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

- You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(c), as requested.
- You may Distribute or Publicly Perform an Adaptation only under the terms of: (i) this License; (ii) a later version of this License with the same License Elements as this License; (iii) a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g., Attribution-ShareAlike 3.0 US); (iv) a Creative Commons Compatible License. If you license the Adaptation under one of the licenses mentioned in (iv), you must comply with the terms of that license. If you license the Adaptation under the terms of any of the licenses mentioned in (i), (ii) or (iii) (the "Applicable License"), you must comply with the terms of the Applicable License generally and the following provisions: (I) You must include a copy of, or the URI for, the Applicable License with every copy of each Adaptation You Distribute or Publicly Perform; (II) You may not offer or impose any terms on the Adaptation that restrict the terms of the Applicable License or the ability of the recipient of the Adaptation to exercise the rights granted to that recipient under the terms of the Applicable License; (III) You must keep intact all notices that refer to the Applicable License and to the disclaimer of warranties with every copy of the Work as included in the Adaptation You Distribute or Publicly Perform; (IV) when You Distribute or Publicly Perform the Adaptation, You may not impose any effective technological measures on the Adaptation that restrict the ability of a recipient of the Adaptation from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Adaptation as incorporated in a Collection, but this does not require the Collection apart from the Adaptation itself to be made subject to the terms of the Applicable License.
- If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and (iv), consistent with Section 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of an Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.
- Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

- This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
- Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

- Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
- Each time You Distribute or Publicly Perform an Adaptation You must also offer to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
- If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
- No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
- This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
- The rights granted under, and the subject matter referred to in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.