TI-Android-GingerBread-2.3-DevKit-1.0 UserGuide

TI Android GingerBread 2.3 DevKit 1.0 User Guide

User Guide - March 31, 2011

About this manual

This document describes how to install and work with Texas Instruments' Android GingerBread DevKit release for OMAP35x, AM37x, AM35x platforms running Android. This release package provides a stable Android distribution with integrated SGX (3D graphics) drivers, TI hardware abstraction for video overlay and standard applications from Android. The package also includes Linux Android kernel, tools and documentation to ease development, deployment and execution of Android based systems. The product forms the basis for all Android application development on OMAP35x, AM37x, AM35x platforms. In this context, the document contains instructions to:

- Install the release
- Setting up the hardware
- Steps to use pre-built binaries in the package
- Running Android on the supported platforms
- Setting up the Android debugger “adb” with the hardware platform
- Installing and executing Android (out of market) applications hardware platforms

Installation

This section describes the list of Software and Hardware requirements to evaluate the DevKit release.

Hardware Requirements

This release of TI Android GingerBread 2.3 DevKit 1.0 is evaluated on the below given list of platforms. This package should be easily portable on other platforms on similar TI devices.

TI Device	Platform Supported	Version	Other Accessories
OMAP35x			
	OMAP35x EVM [1]	Rev G	DVI Monitor, USB HUB, USB Keyboard, USB Mouse, Ethernet, UART Cable, Audio Speakers, MMC/SD Card (2GB min)
	Beagleboard [2]	Rev Cx	DVI Monitor, USB HUB, USB Keyboard, USB Mouse, Ethernet, UART Cable, Audio Speakers, MMC/SD Card (2GB min)
AM35x			
	AM3517 Evaluation Module [3]	Rev C	DVI Monitor, USB HUB, USB Keyboard, USB Mouse, Ethernet, UART Cable, Audio Speakers, MMC/SD Card (2GB min)
AM37x			

	AM37x Evaluation Module [4]	Rev C	DVI Monitor, USB HUB, USB Keyboard, USB Mouse, Ethernet, UART Cable, Audio Speakers, MMC/SD Card (2GB min)
	BeagleBoard [2]	XM	DVI Monitor, USB HUB, USB Keyboard, USB Mouse, Ethernet, UART Cable, Audio Speakers, MMC/SD Card (2GB min)

Software Host Requirements

If you are a Android application developer or would like to use Android SDK Tools then refer to http://developer.android.com/sdk/requirements.html for Host PC requirements.

To evaluate this release we recommend you to have a Linux "Ubuntu 8.04 or above" Host machine, See Ubuntu Linux installation notes [5]

Package Content

```
TI_Android_GingerBread_2_3_DevKit_1_0
    |-- Android_Source_Manifest
    |   `-- TI-Android-GingerBread-2.3-DevKit-1.0.xml
    |-- Documents
    |   |-- RowboPERF_User_Guide.pdf
    |   |-- Software_Manifests
    |   |-- Test_Performance_Results
    |   |   |-- CTS_Report.tar.gz
    |   |   |-- Performance_Results.pdf
    |   |   `-- Test_Results.pdf
    |   |-- TI-Android-GingerBread-2.3-DevKit-1.0_ReleaseNotes.pdf
    |   |-- TI-Android-GingerBread-2.3-DevKit-1.0_UserGuide.pdf
    |   `-- TI-Android-GingerBread-2.3-DevKit-1.0_DeveloperGuide.pdf
    |-- Performance_Apps
    |   |-- 0xbench
    |   |-- 3D
    |   |-- Launcher2
    |   |-- rowboatBench
    |   |-- RowboPERF
    |   `-- StorageIO
    |-- Prebuilt_Images
    |   |-- AM35X
    |   |-- AM37X
    |   |-- beagleboard-rev-c4
    |   |-- OMAP35X
    |-- Tools
    |   |-- flash-utility.tar.gz
    |   |-- mk-bootscr
    |   |-- mk-mmc
    |   |-- pinmux-utility.tar.gz
    |   |-- signGP
    `-- Wireless
        `-- WL1271Android-patches.tar.gz
```

Out of the Box Demo

This section gives the instructions to quickly prepare an SD Card image and get an experience of TI Android GingerBread 2.3 DevKit 1.0 on TI platforms/devices.

- Download the pre-built Image from http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/TI_Android_GingerBread_2_3_DevKit_1_0/index_FDS.html for the platform you own, can be AM35x EVM, AM37x EVM, Beagleboard Rev C4/XM, OMAP35x EVM.
- Get an SD Card of minimum size 2GBytes (Class4 minimum) and a USB Card reader
- Insert the USB SD Card reader (with SD Card) in your host Linux PC
- Prepare the MMC/SD card Image

```
tar -xzvf <Board name>.tar.gz
cd <Board name>
sudo ./mkmmc-android /dev/sd<device>
```

- The above step prepares the SD Card.
- Setup the board/platform
  - Do the DIP switch settings to boot from SD Card, see the section on DIP switch setting below.
  - Insert the SD Card into the Board
  - Switch on the platform
  - Wait for 35sec to get Android up on the UI screen

```
  NOTE: For the first time boot the System might take few minutes to
boot.

  NOTE: If your NAND flash is not empty the system might not boot with
MMC,
        in that case do the following with Serial Console / Terminal
prompt in u-boot

  #> mmc init
  #> fatload mmc 0 0x82000000 boot.scr
  #> source 0x82000000
```

  - Click on the TI logo and run the apps you are interested in, look at the RowboPerf user guide for more details about this app http://processors.wiki.ti.com/index.php/RowboPERF_User_Guide.

Android Booting Procedure

Booting Android on any TI platform requires following software components

- Kernel Image (uImage)
- Bootloader (u-boot.bin)
- Bootstrapping (x-load.bin.ift for NAND or MLO for MMC)
- Filesystem (rootfs)

The above listed software components or images can be populated by

- Building sources from this package
- Using the pre-built images in this package

Software Integration

This section describes the procedure to compile and integrate all the required software components to boot Android on TI platforms.

Toolchain

Download the tool chain and export it in the default Linux Path. The ARM tool chain can be downloaded from Android pre-built repository. Tool Chain Link [6]

```
  Example:

 #> export PATH=<tool chain install
path>/linux-x86/toolchain/arm-eabi-4.4.0/bin/:$PATH
```

Kernel

Untar the kernel source located in the sources directory

```
  #> tar -xzvf Android_Linux_Kernel_2_6_32.tar.gz
```

Execute the following commands to the kernel sources

```
  #> make CROSS_COMPILE=arm-eabi- distclean
```

```
  #> make CROSS_COMPILE=arm-eabi- <default config>

 Where default config is

    omap3_evm_android_defconfig         : For OMAP35x, AM37x
    am3517_evm_android_defconfig        : For AM35x EVM
    omap3_beagle_android_defconfig      : For Beagleboard Rev Cx, XM
```

```
  #> make CROSS_COMPILE=arm-eabi- uImage

 This command will build the Linux Kernel Image in arch/arm/boot
"uImage"
```

u-boot

Untar the u-boot sources located in the sources directory

```
  #> tar -xzvf u-boot.tar.gz
```

Execute the following commands to the kernel sources

```
  #> make CROSS_COMPILE=arm-eabi- distclean
```

```
  #> make CROSS_COMPILE=arm-eabi- <default config>

 Where default config is

    omap3_evm_config         : For OMAP35x, AM37x EVM
    am3517_evm_config        : For AM35x EVM
    omap3_beagle_config      : For Beagleboard Rev Cx, XM
```

```
 #> make CROSS_COMPILE=arm-eabi-

This command will build the u-boot Image "u-boot.bin"
```

x-loader

Untar the x-loader sources located in the sources directory

```
 #> tar -xzvf x-loader-03.00.02.07.tar.gz
```

Execute the following commands to the kernel sources

```
 #> make CROSS_COMPILE=arm-eabi- distclean
```

```
 #> make CROSS_COMPILE=arm-eabi- <default config>

Where default config is

   omap3evm_config          : For OMAP35x, AM37x EVM
   am3517evm_config         : For AM35x EVM
   omap3beagle_config       : For Beagleboard Rev Cx, XM
```

```
 #> make CROSS_COMPILE=arm-eabi-

This command will build the x-loader Image "x-load.bin"
```

To create the MLO file used for booting from a MMC/SD card, sign the x-loader image using the signGP tool found in the Tools directory of the Devkit.

```
 #> ./signGP ./x-load.bin
```

The signGP tool will create a .ift file, that can be renamed to MLO.

```
 NOTE:

- The Pre-built images are provided in this package to help users boot
android without building the sources
```

Setting up Hardware

This DevKit release supports six different platforms, OMAP35x EVM, AM37x EVM, AM35x EVM, Beagleboard Rev Cx, Beagleboard XM. While they are different devices the hardware setup will almost remain the same.

- Connect the UART port of the platform to the Host PC and have a Terminal software like TeraTerm, Minicom or Hyperterminal.
- Connect the Ethernet (on Beagle Rev C4 we don't have an Ethernet port)
- Connect Audio Speakers
- For Beagle board you need to connect DVI Monitor through HDMI connector.
- Use self powered USB HUB and connect it to USB Host port of the platform, mainly for Beagle and AM35x EVM. For AM37x, and OMAP35x EVM the onboard keypad can be used
  - Connect USB keyboard and USB Mouse to the USB HUB for use with Beagle or EVM

```
  NOTE:

  - Beagleboard have no keypad mappings, user is recommended to use USB
```

```
 Keyboard over a self powered USB HUB connected to the Host port of
Beagleboard.
```

- Select Appropriate DIP Switch settings on EVM(s) to boot over MMC/SD

For MMC/SD boot - On OMAP35x and AM37x EVM the DIP switch SW4 should be set as shown below

Switch	1	2	3	4	5	6	7	8
State	OFF	ON	ON	ON	OFF	OFF	OFF	OFF

For MMC/SD boot - On AM35x EVM the DIP switch S7 should be set as shown below

Switch	1	2	3	4	5	6	7	8
State	ON	OFF	OFF	ON	OFF	OFF	OFF	ON

Booting Android

TI platforms (Beagle or EVM) can be booted over MMC or NAND or UART. We follow and prefer MMC based booting of platforms.

Procedure to populate MMC/SD Card

Use the mk-mmc utility provided in the tools folder of this package to populate the SD Card. This utility helps users create a MMC/SD Card with required Images to boot Android on any given TI platform.

This will partition the SD card to three partitions namely boot, rootfs and data. 1) The boot partition will get populated with the images required for booting. 2) The rootfs partition will be used as android root filesystem partition. 3) The Media inside the folder Media_Clips will get copied to the data partition. The data partition will get mounted as EXTERNAL storage when Android boots up.

Execute the following command

```
   Example:

   #>./mkmmc-android /dev/sdc MLO u-boot.bin uImage boot.scr
rootfs.tar.bz2 Media_Clips

This populates the SD/MMC card with all the images.
```

```
  NOTE:

    To create the boot.scr boot script use the mkbootscr tool found in
the Tools directory provided in the DevKit.
```

If you want to use the pre built images in the DevKit you have to adjust the above mentioned command to take them into account, as a more direct example the commands below will generate a SD card for an OMAP3 EVM. Ensure you have your SD card connected to the Linux machine you are using and that it is in /dev/sdb for this command otherwise adjust the command accordingly (**WARNING**: if you get this wrong it can wipe your HDD). Note that this assumes you installed the SDK in your home (~) directory and that the command is run with sudo (or your preferred way of getting super user privileges) to allow for the reformatting of the SD card.

```
  HOST $ cd ~/TI_Android_GingerBread_2_3_DevKit_1_0/OMAP35X
  HOST $ sudo ../../Tools/mk-mmc/mkmmc-android.sh /dev/sdb
```

Procedure to add Video, Audio and other media

To play media after booting Android on any platform, the content must be included in the MMC/SD card's FAT32 partition. If you use the mk-mmc script included in the release package then it creates 3 partitions. The media content should be placed into the 3rd (FAT32) partition.

```
   Example:

  #> sudo mount /dev/sdd3 /mnt
  #> sudo cp <all media files> /mnt
  #> sudo umount /mnt
```

```
   NOTE:

  - This release supports all the standard Android media formats,
listed here http://developer.android.com/guide/appendix/
media-formats.html
```

Booting the platform

Booting over MMC using boot.scr

```
   NOTE:

  - If the board has bootargs configured already, then the board will
not boot for Android automatically,
  - It is suggested to either delete the bootargs or use the following
commands on u-boot prompt through UART console.
```

```
   #> mmc init
   #> fatload mmc 0 0x82000000 boot.scr
   #> source 0x82000000
```

If the board is not configured for bootargs, then it automatically boots.

Boot arguments

Boot arguments for various boards are as follows.

```
  AM35X:
  setenv bootargs 'console=ttyS2,115200n8 androidboot.console=ttyS2
mem=256M root=/dev/mmcblk0p2 rw rootfstype=ext3 rootdelay=1 init=/init
ip=off mpurate=600 omap_vout.vid1_static_vrfb_alloc=y'
```

```
  AM37X:
  setenv bootargs 'console=ttyS0,115200n8 androidboot.console=ttyS0
mem=256M root=/dev/mmcblk0p2 rw rootfstype=ext3 rootdelay=1 init=/init
ip=off mpurate=1000 omap_vout.vid1_static_vrfb_alloc=y'
```

```
  Beagleboard-rev-c4:
  setenv bootargs 'console=ttyS2,115200n8 androidboot.console=ttyS2
mem=256M root=/dev/mmcblk0p2 rw rootfstype=ext3 rootdelay=1 init=/init
ip=off mpurate=720 omap_vout.vid1_static_vrfb_alloc=y'
```

```
  Beagleboard-xm:
  setenv bootargs 'console=ttyS2,115200n8 androidboot.console=ttyS2
mem=256M root=/dev/mmcblk0p2 rw rootfstype=ext3 rootdelay=1 init=/init
ip=off mpurate=1000 omap_vout.vid1_static_vrfb_alloc=y'
```

```
  OMAP35XEVM:
  setenv bootargs 'console=ttyS0,115200n8 androidboot.console=ttyS0
mem=128M root=/dev/mmcblk0p2 rw rootfstype=ext3 rootdelay=1 init=/init
ip=off mpurate=720 omap_vout.vid1_static_vrfb_alloc=y'
```

Keypad mappings

The below table lists the keypad and USB Keyboard mappings to Android UI functionality.

Functionality	USB Keyboard/Mouse	Keypad on OMAP35x/AM37x EVM	Keypad on AM35x EVM
Home Screen	Home	R3C2	S3
Left	Left Arrow	R2C1	S6
Right	Right Arrow	R0C2	S5
Up	Up Arrow	R1C3	S7
Down	Down Arrow	R2C0	S9
Volume Up	Volume Up	R1C2	S4
Volume Down	Volume Down	R0C1	S1
Contacts	F3		
Power		R0C0	S2
Select	Enter	R3C1	
Back	Mouse right	R2C3	S8
Menu	F1	R3C3	S10

Using Network Filesystem

Android filesystem can be mounted over network, the bootargs for doing the same should include below text instead of MMC

```
   ip=dhcp rw root=/dev/nfs nfsroot=<your NFS server
ipaddr>:/home/USER/FILESYSTEM_DIR,nolock noinitrd
```

```
   Example: Complete bootargs for OMAP35x board using NFS and LCD
output

   setenv bootargs init=/init console=ttyS0,115200n8 ip=dhcp rw
root=/dev/nfs nfsroot=192.168.133.01:/home/user/targetfs,nolock
mem=256M noinitrd androidboot.console=ttyS0
```

Using Flashing Utility

Android boot images x-loader and u-boot etc can be flashed to NAND from windows host, using flashing utility provided in tools. Flashing utility is tested on USB connection with AM37X Pre-built images. For instructions on flashing utility please refer the steps at http://processors.wiki.ti.com/index.php/Flash_v1.5_User_Guide.

In some cases the flashing utility's recommended settings of HWECC, 4-bit for flashing Uboot and the kernel image may not work and the board displays a bad CRC message. If this happens re-flash the components again using the SWECC setting of the utility for Uboot and kernel.

Display Support

Using DVI Monitor

On OMAP35x, AM37x and AM35x EVMs the on board LCD is used as output device by default. User is allowed to configure DVI port as output device, by changing the boot arguments as shown below.

Append the boot arguments with following text

```
   omapfb.mode=dvi:1280x720MR-16 omapdss.def_disp="dvi"
```

```
   Example:

  To boot over MMC and use DVI at resolution 1024x768 on OMAP35x EVM,
the complete bootargs would be,

  setenv bootargs init=/init console=ttyS0,115200n8 ip=dhcp rw
root=/dev/mmcblk0p2 rw init=/init rootwait mem=256M
androidboot.console=ttyS0 omapfb.mode=dvi:1024x768MR-16
omapdss.def_disp="dvi"
```

We have noticed on few monitors the below bootargs works for DVI.

```
  setenv bootargs console=ttyS0,115200n8 androidboot.console=ttyS0
mem=256M root=/dev/mmcblk0p2 rw rootfstype=ext3 rootdelay=1 init=/init
ip=off mpurate=1000 omap_vout.vid1_static_vrfb_alloc=y vram=16M
omapfb.vram=0:8M,1:4M,2:4M omapfb.mode=dvi:hd720 omapdss.def_disp=dvi
```

```
  NOTE:

  - On beagleboard the DVI port is configured as default output device.
```

Using S-VIDEO Monitor

By Setting Boot parameters

On OMAP35x, AM37x and AM35x EVMs the on board LCD is used as output device by default. User is allowed to configure S-video port as output device, by changing the boot arguments as shown below.

Append the boot arguments with following text

```
   omapdss.def_disp="tv"
```

```
  Example:

  To boot over MMC and use S-video port as output device on OMAP35x
EVM, the complete bootargs would be,

  setenv bootargs 'console=ttyS0,115200n8 androidboot.console=ttyS0
mem=256M root=/dev/mmcblk0p2 rw rootfstype=ext3 rootdelay=1 init=/init
ip=off mpurate=720 omap_vout.vid1_static_vrfb_alloc=y
omapdss.def_disp="tv"'
```

By Sysfs configuration

Boot the device with LCD as default output device.

```
  To boot over MMC and use lcd as output device on OMAP35x EVM, the
complete bootargs would be,
  setenv bootargs 'console=ttyS0,115200n8 androidboot.console=ttyS0
mem=256M root=/dev/mmcblk0p2 rw rootfstype=ext3 rootdelay=1 init=/init
ip=off mpurate=720 omap_vout.vid1_static_vrfb_alloc=y'
```

Switch video on S-video device

Use the following procedure to display video on s-video device.

Disable overlay1. Before changing any overlay attributes, it should first be disabled.

```
  #echo 0 > /sys/devices/platform/omapdss/overlay1/enabled;
```

Reset overlay1 manager which is set to lcd by default

```
  #echo "" > /sys/devices/platform/omapdss/overlay1/manager;
```

Disable display1/tv

```
  #echo 0 > /sys/devices/platform/omapdss/display1/enabled;
```

Set overlay manager to tv

```
  #echo "tv" > /sys/devices/platform/omapdss/overlay1/manager;
```

Enable display1/tv

```
  #echo 1 > /sys/devices/platform/omapdss/display1/enabled
```

Open gallery application and play video. Video will be played on s-video device and graphics will be displayed on the lcd.

Switch video on lcd device

Use the following procedure to display video on lcd device.

Disable overlay1. Before changing any overlay attributes, it should first be disabled.

```
  #echo 0 > /sys/devices/platform/omapdss/overlay1/enabled;
```

Disable display1/tv

```
  #echo 0 > /sys/devices/platform/omapdss/display1/enabled;
```

Disable display0/lcd

```
  #echo 0 > /sys/devices/platform/omapdss/display0/enabled;
```

Reset overlay1 manager which is previously set to tv

```
  #echo "" > /sys/devices/platform/omapdss/overlay1/manager;
```

Set overlay manager to lcd

```
  #echo "lcd" > /sys/devices/platform/omapdss/overlay1/manager;
```

Enable display0/lcd

```
  #echo 1 > /sys/devices/platform/omapdss/display0/enabled
```

Open gallery application and play video. Video and graphics will be displayed on the lcd.

Rotation Support

Graphics Rotation

Graphics rotation is managed by software. Use standard android API to rotate screen in portrait or landscape mode. Currently portrait and landscape graphics rotation mode is supported.

Video Rotation

Video rotation is implemented with use of Rotation Engine module in Virtual Rotation Frame Buffer(VRFB) module in OMAP3x. Rotation engine supports rotation of an image with degree 0, 90, 180 and 270.

V4L2 driver supports rotation by using rotation engine in the VRFB module. Driver provides ioctl interface for enabling/disabling and changing the rotation angle. These ioctls are VIDIOC_S_CTRL/VIDIOC_G_CTRL.ioctl.

Following code shows how to set rotation angle to 90 degree:

```
  struct v4l2_control control;
  int degree = 90;
  control.id = V4L2_CID_ROTATE;
  control.value = degree;
  ret = ioctl(fd, VIDIOC_S_CTRL, &control);
  if (ret < 0) {
  perror("VIDIOC_S_CTRL\n");
  close(fd);
  exit(0);
  }
  /* Rotation angle is now set to 90 degree. Application can now do
streaming to see rotated image*/
```

Camera Support

Ti Android GingerBread 2.3 Devkit 1.0 supports camera sensor (mt9v113) for beagleboard-xm.

Mt9v113.jpeg

Feature supported:

- Image Capture

1. Go to application/activity view luncher > open camera
2. Click on camera capture.By default images will get stored at - */sdcard/DCIM*

- Video recording

1. Go to application/activity view luncher > open camera
2. Press 'F1' to switch to video recording view
3. Press 'Enter' to start/stop video recording

```
Note : All recording will be done using default stagefright encoders. Format :
.3gpp Video Encoder : H.263 Audio Encoder : AMR
```

Wireless

AM37x evm revG with the wireless module supports WLAN and Bluetooth on Android. For steps on installing the wireless module see this page.

WLAN

- Menu->Settings->Wireless & networks->Wi-Fi settings
- click Wi-Fi and wait for few seconds. The following logs appear on serial console output

```
TIWLAN: driver init
TIWLAN: 977.211973: wlanDrvIf_Open()
TIWLAN: 977.331480: pInitParams->RoamingScanning_2_4G_enable 0
SDIO clock Configuration is now set to 24Mhz
TIWLAN: 977.543120: CHIP VERSION... set 1273 chip top registers
TIWLAN: 977.549559: Working on a 1273 PG 2.0 board.
TIWLAN: 977.554228: Starting to process NVS...
TIWLAN: 977.558470: No Nvs, Setting default MAC address
TIWLAN: 977.563444: pHwInit->uEEPROMCurLen: 1c
TIWLAN: 977.567656: ERROR: If you are not calibating the device, you
will soon get errors !!!
TIWLAN: 977.576109: Chip ID is 0x4030111.
TIWLAN: 977.580198: FEM Type 1
```

```
TIWLAN: 977.583342: Starting to download firmware...
TIWLAN: 977.665770: Starting to download firmware...
TIWLAN: 977.693602: Starting to download firmware...
TIWLAN: 977.698454: Starting to download firmware...
TIWLAN: 977.710875: Starting to download firmware...
TIWLAN: 977.719420: Starting to download firmware...
TIWLAN: 977.725706: Finished downloading firmware.
TIWLAN: 977.730314: Firmware running.
TIWLAN: 977.749785:
TIWLAN: 977.751707:
-----------------------------------------------------------------
TIWLAN: 977.759611: Driver Version  : WiLink_Driver_6.1.0.0.144
TIWLAN: 977.765318: Firmware Version: Rev 6.1.0.0.335
TIWLAN: 977.770170: Station ID      : 08-00-28-12-34-56
TIWLAN: 977.775145:
-----------------------------------------------------------------
TIWLAN: 977.783018:
TIWLAN: 977.799376: Interrogate TX/RX parameters
```

- On the UI, Green check-mark appears and status shows scanning
- After scan completes, available APs are listed
- connect to desired AP by clicking on its name and enter required details (username/key etc) and click connect
- The following appears on console

```
TIWLAN: 1248.835661: ************ NEW CONNECTION ************
TIWLAN: 1248.841246: -- SSID  = APNAME
TIWLAN: 1248.845091: -- BSSID = xx-xx-xx-xx-xx-xx
TIWLAN: 1248.849516: ************************************
```

- when successfully connected, you will see status as connected to APNAME
- To turn off Wi-Fi, click Wi-Fi in Menu->Settings->Wireless & networks
- Following message appears on console and the green check mark is not visible on UI

```
TIWLAN: 1706.198423: wlanDrvIf_Release()
TI WLAN: driver unloaded
```

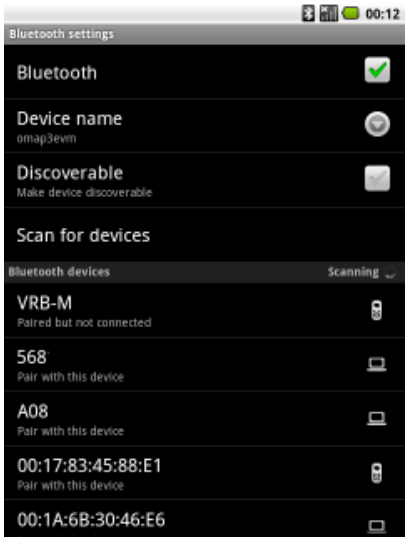
NOTE: For information on WLAN calibration refer to WLAN Calibration section of Android wireless porting guide.

Bluetooth

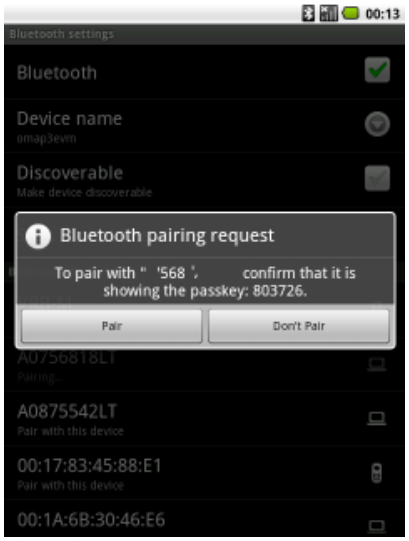
- Menu->Settings->Wireless & networks->Bluetooth settings
- Click Bluetooth and wait for few seconds
- LED LD3 turns ON on the wireless module and the Bluetooth icon appears on taskbar. The following appears on the debug console

```
Set BT_EN of WL1271
WL1271: Powering on
```

- If BT is enabled green check-mark appears and status shows scanning
- available bt devices are listed



- click on desired device to pair with
- popup with pin will appear



- click Pair button to confirm pairing
- verify that the desired device shows the same pin. click ok. devices paired

NOTE: When pairing with Bluetooth headset, pin may not be displayed. Android attempts to pair automatically with Bluetooth headsets. Pin dialog will be shown only if auto-pairing fails.

- To turn off Bluetooth, click Bluetooth in Menu->Settings->Wireless & networks
- LED LD3 turns OFF on the wireless module, Bluetooth icon is not displayed on taskbar and the following messages appear on console:

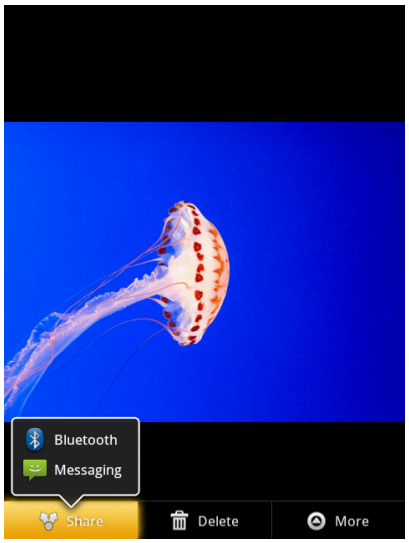
```
Set BT_EN of WL1271
WL1271: Powering off
```

Bluetooth Object Push profile

Using Bluetooth, it is possible to send receive files (pictures, media files etc).

Sending files

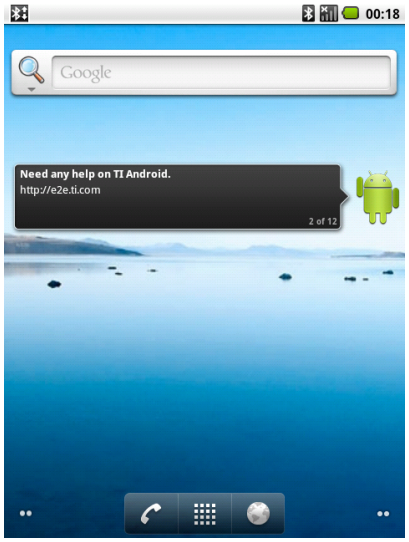
- go to Menu ->Gallery
- Select a picture to share
- click Menu (bottom right)
- click share, select bluetooth from the options



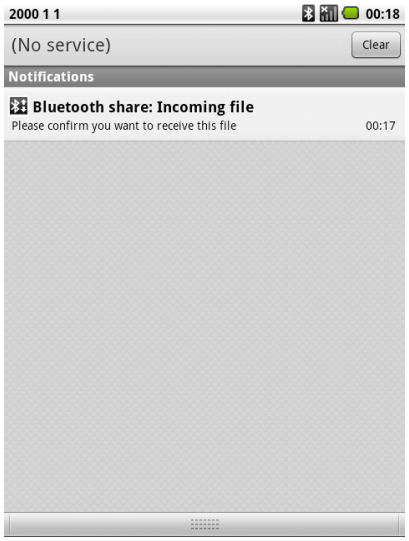
- select paired bt device to send to
- the bt device will prompt to accept. accept incoming file
- once download finished. check file

Receiving files

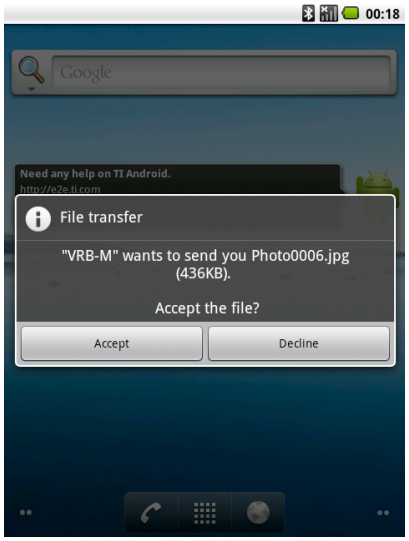
- on paired device (e.g. phone), select send to bt, click on omap3evm
- on evm, notification appears about incoming connection (top left)



- open task bar. click on note



- in popup click accept

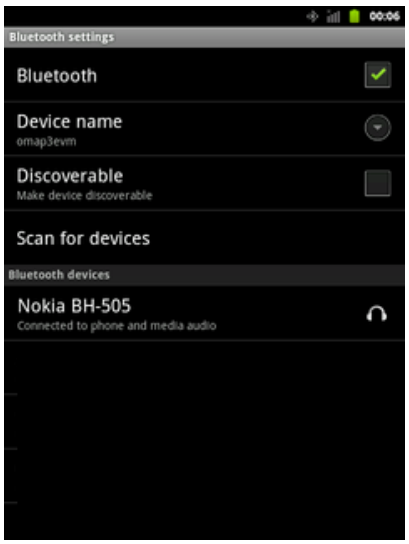


- once download complete. check file

Bluetooth A2DP

You can listen to Media audio on Bluetooth A2DP headset.

- Pair A2DP capable bluetooth headset with device. Android uses the stereo headset icon to denote A2DP headset
- After pairing succeeds, the status is updated to 'Connected to phone and media audio'.



- Open Music player and play any audio clip
- Audio will be heard on the Bluetooth headset

Bluetooth AVRCP

You can control Media playback with Media player keys on Bluetooth headset with AVRCP capabilities.

NOTE: The following steps assume Bluetooth A2DP headset with AVRCP.

- Pair the BT headset
- The following text appears on the debug serial when the pairing is successfully completed. This confirms that AVRCP feature is registered with android

```
input: AVRCP as /devices/virtual/input/input3
```

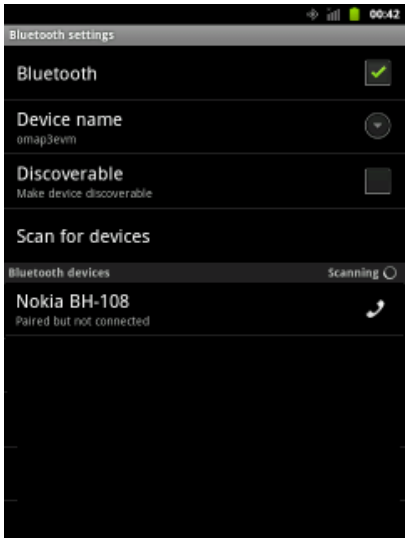
- Open Music Player and go to playlist view. Check that there are multiple clips in the playlist.
- Press the Play/Pause button on the BT headset - The currently queued clip begins playing on the headset
- Press the Play/Pause button again on the headset - The currently playing clip is paused
- Press the Next button on the headset; the next clip in the playlist begins to play on the headset
- Press the Prev button on the headset; the currently playing clip restarts from the begining

Bluetooth SCO Audio

Android supports the Bluetooth Headset Profile/HandsFree Profile Audio Gateway (HSP/HFP AG). Bluetooth SCO connection is used to play/capture audio in these profiles.

It is possible to record audio from Bluetooth headset over SCO.

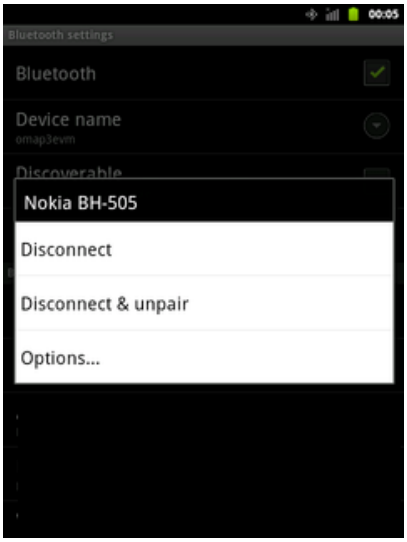
- Pair Bluetooth headset with device
- After pairing succeeds, the status is updated to 'Connected to phone audio'. If using A2DP headset, the status is 'Connected to media and phone audio'



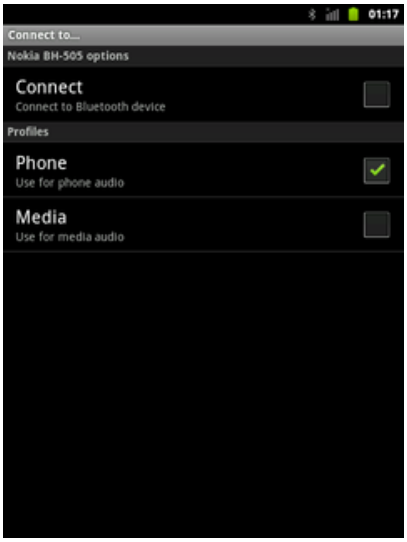
- Menu -> BluetoothSCOApp. BluetoothSCOApp is a sample app to establish SCO connectivity with a Bluetooth headset
- Tap once so that check mark appears against 'Enable Music via Media Player'

NOTE: If using A2DP headset, you need to disable the A2DP profile before BluetoothSCOApp can be used. To disable A2DP follow the steps below

- Menu -> Settings -> Wireless & networks -> Bluetooth settings
- Long tap on the entry for A2DP headset, a menu appears



- Tap Options
- Uncheck Media. This stops android from using A2DP profile of the headset



- Menu -> Sound Recorder
- Tap record button
- Speak into the bluetooth headset mic, the VU meter display moves as per the sound. This confirms that audio is being recorded from bluetooth headset
- Stop recording and save the clip.

The audio is recorded in 3gpp format and can be played back by the Music player.

NOTE: The audio from Music player can be heard only over audio jack or A2DP. Headsets supporting only HSP/HFP cannot be used as audio playback device with Music Player or Gallery app. Please see Bluetooth SCO Audio section in the porting guide for more information.

Power Management

Below are the features supported in TI-Android-GingerBread-2.3-DevKit-1.0

AM37X, OMAP35X

- Change of LCD backlights based on Wake Locks and Screen Timeouts (DIM,OFF)
- LCD back light brightness control from Settings Application
- Suspending the device to Memory.The device can be suspended in two ways.

1. Pressing the POWER key on the keypad
2. Allowing the system go to suspend after a screen timeout

- Prevent Suspend based on Wake Locks
- System Resume on Key Press
- CPU Dynamic Voltage and Frequency Scaling (ondemand, performance, powersave and userspace governors)
- CPU Idle States
- Enabling system for hitting retention during idle
- Enabling system for hitting OFF

AM35X

- Suspending the device to Memory.The device can be suspended in two ways.

1. Pressing the POWER key on the keypad
2. Allowing the system go to suspend after a screen timeout

- Prevent Suspend based on Wake Locks
- System Resume on Key Press
- Android Early Suspend Feature for frame buffer driver

Basic Settings

To go in suspend mode

- Press POWER key on the keypad

To resume from suspend mode

- Press any button on the key pad

To set the Screen timeout to go suspend

- Select Settings->Display-> Screen Timeout
- Select one of the options from the list

To set set the screen always on preventing suspend

- Select Settings-> Applications-> Development-> Stay awake

To set Screen brightness

- Select Settings->Display-> Brightness

Advanced Settings

To Disable Power Management

- Edit init.rc file on the root directory.
- Set the property hw.nopm to true
- This will prevent POWER key suspend
- Select Settings-> Applications-> Development-> Stay awake
- The above will prevent screen timeout suspend

Enabling system for hitting retention during idle

```
   #echo 1 > /debug/pm_debug/sleep_while_idle
```

Enabling system for hitting OFF

```
   #echo 1 > /debug/pm_debug/enable_off_mode
```

By default sleep_while_idle is set to false and enable_off_mode is set to true

CPU Dynamic Voltage Frequency Scaling settings

Enabling ondemand frequency governor

The ondemand governor enables DVFS(frequency/OPP) transitions based on CPU load.

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

Enabling performance frequency governor

The performance governor keeps the CPU always at the highest frequency.

```
  #echo performance >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

Enabling powersave frequency governor

The powersave governor keeps the CPU always at the lowest frequency.

```
  #echo powersave >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

Enabling userspace frequency governor

Once this governor is enabled, DVFS(frequency) transitions will be manually triggered by a userspace application by using the CPUfreq sysfs interface

```
#echo userspace >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

See all the available operating points

```
  #cat
/sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
```

Application can select any of the available frequency from the above

```
  #echo  <Desired Frequancy> >
/sys/devices/system/cpu/cpu0/cpufreq/ scaling_setspeed
```

Checking CPU IDLE states usage

There are seven power states introduced by CPU Idle

The usage and time count for these different states can be checked via

```
  #cat /sys/devices/system/cpu/cpu0/cpuidle/state*/time
  #cat /sys/devices/system/cpu/cpu0/cpuidle/state*/usage
```

Limitations

- In beagle and am35x platforms there is no keypad connected to the wake up domain. So wake up is not possible by pressing keys. So power management is disabled by default in am35x and beagle platforms.
- To enable power management in am35x platform, set Settings -> Applications -> Development -> Stay Awake to true. In this case, you can wake up am35x platform like below
  Press ENTER on Serial Port console and then press any key on the keypad to wake up the device.
- If the usb device connected to the USB EHCI port when suspending the device, it might not work properly after resume.

So Disconnect any usb device connected to USB EHCI port when suspending the device.

- Display starts flashing sometimes, once sleep_while_idle and enable_off_mode both enabled in AM37X.

NAND Booting

Fastboot and UBI rootfs

Fastboot flashing utility is for updating the different software components of Android. Here is a guide to reflash the xloader, u-boot, kernel and root-filesystem (UBIFS image). This guide assume that Rowboat has been compiled before trying out these instructions.

Establishing Fastboot connectivity

- Connect serial port to host PC via null modem cable.
- Serial port settings: 115200 8N1, No flow control.
- Apply power to the board.
- Press any key in serial port utility during boot and get U-boot command prompt.
- Run "fastboot" on u-boot command prompt (u-boot will echo "fastboot initialized").
- Connect USB cable between USB OTG port of the board and host PC.

Setup on linux host

- On command prompt, run

```
$ export ANDROID_ROOT=<rowboat top level directory>
```

```
$ cd $ANDROID_ROOT/out/host/linux-x86/bin
```

```
$ sudo ./fastboot devices
```

if a device number is echoed, fastboot is working.

Setup on Windows host

- Refer ADB over USB on Windows Machine [7]
- Edit android_winusb.inf - under section [Google.NTx86], add line as below:

```
%SingleBootLoaderInterface% = USB_Install, USB\VID_0451
```

- Proceed installing, with the difference that device to be selected is "Android Bootloader Interface" instead of "Android ADB Interface".

Creating ubifs images

- Make sure you have all the necessary tools to build the UBI image.

```
$ sudo apt-get install mtd-utils
```

- Create a self explanatory ubinize.cfg configuration file. For example:

```
$ cat ubinize.cfg
[ubifs]
mode=ubi
image=ubifs.img
vol_id=0
vol_size=200MiB
vol_type=dynamic
vol_name=rootfs
vol_flags=autoresize
```

- Create a placeholder for the android filesystem. This will be the source for the UBI image

```
$ mkdir temp
```

The temp directory can be populated by extracting the tarball of the prebuilt image (*.tar.bz2).

```
$ tar xvjf <rootfs tarball> --numeric-owner -C temp/
```

To populate the temp directory with the filesystem created by building from sources

```
$ cp -r
<android_src_top>/out/target/platfrom/<target_product>/root/*
 temp
  $ cp -r
<android_src_top>/out/target/platfrom/<target_product>/system
 temp
```

- Add all the permissions to the filesystem. Android will eventually re-assign the correct/necessary permissions on first boot.

```
$ chmod -R 777 temp/
```

- Create ubifs.img. Note it will not be the one which will be used for flashing. On Am37x board it will be,

```
$ sudo mkfs.ubifs -r temp/ -m 2048 -e 129024 -c 1948 -o ubifs.img
```

- Create ubi.img for flashing.

```
$ ubinize -o ubi.img -m 2048 -p 128KiB -s 512 ubinize.cfg
```

Fastboot commands

```
$ export ANDROID_ROOT=<rowboat_top_level_build_directory>
```

```
$ cd $ANDROID_ROOT/out/host/linux-x86/bin
```

- List connected devices

```
$ sudo ./fastboot devices
```

- Update xloader.

```
$ sudo ./fastboot flash xloader <xloader_binay_path>/MLO
```

- Updating u-boot.

```
$ sudo ./fastboot flash bootloader
<uboot_binary_path>/u-boot.bin
sending 'bootloader' (203 KB)... OKAY [  0.541s]
          writing 'bootloader'... OKAY [  0.552s]
finished. total time: 1.093s
```

- Updating kernel.

```
$ sudo ./fastboot flash boot <kernel_image_path>/uImage
sending 'boot' (2537 KB)... OKAY [  6.466s]
                writing 'boot'... OKAY [  3.330s]
finished. total time: 9.796s
```

- Updating filesystem.

```
$ sudo ./fastboot flash system <rootfs_image_path>/ubi.img
sending 'system' (72320 KB)... OKAY [183.758s]
              writing 'system'... OKAY [ 84.040s]
finished. total time: 267.819s
```

- Erasing partition.

```
$ sudo ./fastboot erase <partition name> (eg. xloader)
```

- Reboot

```
$ sudo ./fastboot reboot
```

- Display fastboot variable

```
$ sudo ./fastboot getvar <variable>
```

- Exit fastboot mode in uboot

```
$ sudo ./fastboot continue
```

Booting with UBIFS rootfs

- Set the bootarguments from u-boot prompt.

```
# setenv nandboot 'echo Booting from nand ...; nand read ${loadaddr}
${boot_nand_offset} ${boot_nand_size}; bootm ${loadaddr}'
```

```
# setenv bootcmd 'run nandboot'
```

```
# setenv bootargs 'init=/init console=ttyS0,115200n8 noinitrd ip=off
androidboot.console=ttyS0 rootwait mem=256M \
    omap_vout.vid1_static_vrfb_alloc=y rw ubi.mtd=4 rootfstype=ubifs
root=ubi0:rootfs bootdelay=2'
```

Limitations

- AM3517evm u-boot doesn't support Full Speed USB mode(USB 1.1) and it expects HOST machine USB port to be High Speed USB (USB 2.0).

YAFFS2 rootfs

Following the official(?) procedure how to incorporate yaffs linux [8] to write/flash the YAFFS2 image i.e flash_erase <mtd_partition>, mount the <mtd_partition> and copy/untar the Android rootfs, from a running system (MMC/NFS), onto the mount_point to boot from NAND partition.

flashing the MTD partition

- Copy the rootfs.tar.bz2 file-system to the rootfs partition of MMC/SD card.
- Now boot the device from MMC/SD card and determine the MTD device for the file system partition of you Flash device

```
target# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00260000 00020000 "U-Boot"
mtd1: 00020000 00020000 "U-Boot Env"
mtd2: 00440000 00020000 "Kernel"
mtd3: 0c820000 00020000 "File System"
mtd4: 03120000 00020000 "Reserved"
```

The file system partition of the flash device is /dev/mtd/mtd3

- Erase the file system partition

```
target# flash_eraseall /dev/mtd/mtd3
Erasing 16 Kibyte @ 3b9c000 -- 99 % complete.
```

- Mount the Flash file system partition using the block device node on a temporary directory /tempd.

```
target# mkdir /tempd
```


Note: **For small block NAND devices (i.e. DM6446) you must mount the file system as a YAFFS file system.**
For large block NAND devices (i.e. DM355/AM389X) the file system should be mounted as YAFFS2.

Using the wrong version of YAFFS will result in error messages similar to:

```
target# mount -t yaffs2 /dev/block/mtdblock3 /tempd
yaffs: dev is 32505859 name is "mtdblock3"
yaffs: Attempting MTD mount on 31.3, "mtdblock3"
yaffs: MTD device does not support have the right page sizes
mount: wrong fs type, bad option, bad superblock on /dev/mtdblock3,
       or too many mounted file systems
```

- Untar the contents of the tarball file system image to the Flash device

```
target# cd /tempd
```

```
target# busybox tar xjf /rootfs.tar.gz
```

- Unmount the Flash file system partition

```
target# cd /
```

```
target# umount /tempd
```

- Now the target is ready to mount the rootfs from NAND.

```
target# reboot
```

Booting with YAFFS2 rootfs

- Set the YAFFS2 bootarguments from u-boot prompt.

```
# setenv bootargs 'init=/init console=ttyS0,115200n8 noinitrd ip=off
 androidboot.console=ttyS0 rootwait mem=256M \
    omap_vout.vid1_static_vrfb_alloc=y rw root=/dev/mtdblock3
rootfstype=yaffs2 bootdelay=2'
```

To Do List

- Fastboot flashing support for YAFFS2

Building Android Sources

Android sources (filesystem) can be built by following the instructions documented here

We recommend to use Ubuntu 10.04+ (32-bit) , but also you can use CentOS 5.3 (32 bit).

- Install DHCP server
- Install tftp server (actual for OMAP3EVM)
- Install repo tool
- Set up your Linux development environment, make sure you have the following:

Required packages:

Git 1.5.4 or newer and the GNU Privacy Guard. JDK 6.0 (32-bit). JDK 5 is not supported. flex, bison, gperf, libsdl-dev, libesd0-dev, libwxgtk2.6-dev (optional), build-essential, zip, curl, minicom, tftp-server, uboot-mkimage

For Ubuntu 32-bit use such command:

```
  $ sudo add-apt-repository "deb http://archive.canonical.com/
lucid partner"
  $ sudo add-apt-repository "deb-src http://archive.canonical.com/
ubuntu lucid partner"
  $ sudo apt-get update
  $ sudo apt-get install git-core gnupg sun-java6-jdk flex bison gperf
libsdl-dev libesd0-dev libwxgtk2.6-dev build-essential zip curl
libncurses5-dev zlib1g-dev minicom tftpd uboot-mkimage expect
  $ sudo update-java-alternatives -s java-6-sun
```

Ubuntu Intrepid (8.10) users may need a newer version of libreadline:

```
  $ sudo apt-get install lib32readline5-dev
```

Configure your network

- Configure your host Ethernet adapter

```
  sudo ifconfig ethX 10.10.10.1 netmask 255.255.255.0 up
```

- Configure DHCP server for target's network

```
     ; Example dhcpd.conf
     default-lease-time 600;
     max-lease-time 7200;
     subnet 10.10.10.0 netmask 255.255.255.0 {
       range 10.10.10.2 10.10.10.10;
```

```
     host beagleboard_rev_c3 {
       hardware ethernet 00:80:C8:xx:xx:xx;
       fixed-address 10.10.10.2;
     }
```

```
     host omap3evm_rev_d {
       hardware ethernet 00:50:c2:xx:xx:xx;
       fixed-address 10.10.10.3;
       filename "evm/uImage"; <- this string actual for boot with
tftp
     }
```

- Configure TFTP-server

```
    /etc/xinet.d/tftpd:
```

```
    service tftp
    {
      protocol        = udp
      port            = 69
      socket_type     = dgram
      wait            = yes
      user            = nobody
      server          = /usr/sbin/in.tftpd
```

```
      server_args     = /tftpboot
      disable         = no
    }
```

- Clone the Sources

```
   #> mkdir rowboat-android
   #> cd rowboat-android
   #> repo init -u git://gitorious.org/rowboat/manifest.git -m
TI-Android-GingerBread-2.3-DevKit-1.0.xml
   #> repo sync
```

- Build the root file system for AM37x (Beagle XM, AM37x REV G EVM)

```
   #> make TARGET_PRODUCT=omap3evm OMAPES=5.x -j8
```

- Build the root file system for AM3517

```
   #> make TARGET_PRODUCT=am3517evm OMAPES=3.x -j8
```

- Build the root file system for OMAP35x

```
   #> make TARGET_PRODUCT=omap3evm OMAPES=3.x -j8
```

- Install the SGX (Open GL drivers) libraries and package into filesystem

This step is not needed, but still given here to explain how SGX components gets added into a Filesystem. For more details, http:/ / code. google. com/ p/ rowboat/ wiki/ ConfigureAndBuild#Install_the_Android_Graphics_SGX_SDK_on_Host_Machine

- Prepare the root filesystem

```
   Follow the steps below to populate the Android filesystem.

   #> sudo ../../../../build/tools/mktarball.sh
../../../host/linux-x86/bin/fs_get_stats android_rootfs . rootfs
rootfs.tar.bz2

   The rootfs.tar.bz2 is the android filesystem, it can be put on a
SD/MMC Card or used our NFS.
```

ADB Android Debugger & Downloader

Android Debug Bridge (adb) is a versatile tool lets you manage the state of the Android-powered device. For more information about what is possible with adb, see Android Debug Bridge page at http://developer. android. com/ guide/developing/tools/adb.html.The ADB tool can be used to

- Download an application from a host machine, install & run it on the target board.
- Start a remote shell in the target instance.
- Debug applications running on the device using the debugging tool DDMS (Dalvik Debug Monitor Server) which runs on top of adb connection.
- Copy files to and from the board to host machine

Downloading "ADB" & Host setup

The adb tool is a part of Android SDK package located at http://developer.android.com/sdk/index.html.For an overview of how to install and set up the Android SDK, follow download & setup instructions from http://developer.android.com/sdk/index.html.Once you install Android SDK, the directory contents look like this.

```
  add-ons/
  docs/
  platforms/
    <platform>/
       data/
       images/
       skins/
       templates/
       tools/
       android.jar
  samples/
  tools/
SDK Readme.txt
```

The adb tool is located in tools/ directory under the Android SDK installation. Export the tools directory path as shown below.

```
  $> export PATH=${PATH}:<your_sdk_dir>/tools
```

Connecting Host machine & board through adb

This release of DevKit has been tested for three different methods of connecting a given board with host machine

- adb over USB
- adb over USB Ethernet
- adb over Ethernet

The below sections describe each of these methods and provides necessary instructions for the same.

adb over USB

- Make sure that the mini-usb cable is connected between the host usb port and the target's USB OTG port
- Turn on "USB Debugging" on your board. On the board (UI screen)-
  - Go to home screen, press MENU,
  - Select Applications, select Development, then enable USB debugging.
  - Alternatively, you can navigate to Settings->Applications->Development and then enable the "USB debugging" option.
- Setup host machine to detect the board. On Ubuntu Linux host machines this is done by adding a rules file to configure device vendor ID of on-board OMAP device.
- For the EVMs and Boards covered here, the vendor ID is "18d1".
  - Log in as root and create this file: **/etc/udev/rules.d/51-android.rules**

```
    For Gusty/Hardy, edit the file to read:
    SUBSYSTEM=="usb", SYSFS{idVendor}=="18d1", MODE="0666"
```

```
    For Dapper, edit the file to read:
    SUBSYSTEM=="usb_device", SYSFS{idVendor}=="18d1", MODE="0666"
```

- Execute the following to change the user mode for the rules file.

```
    $> chmod a+r /etc/udev/rules.d/51-android.rules
```

- Verify the adb connectivity between host and target board

```
    $> adb devices
```

```
    If device is connected, then output on screen should list the
device, example:

       List of devices attached
       20100720    device
```

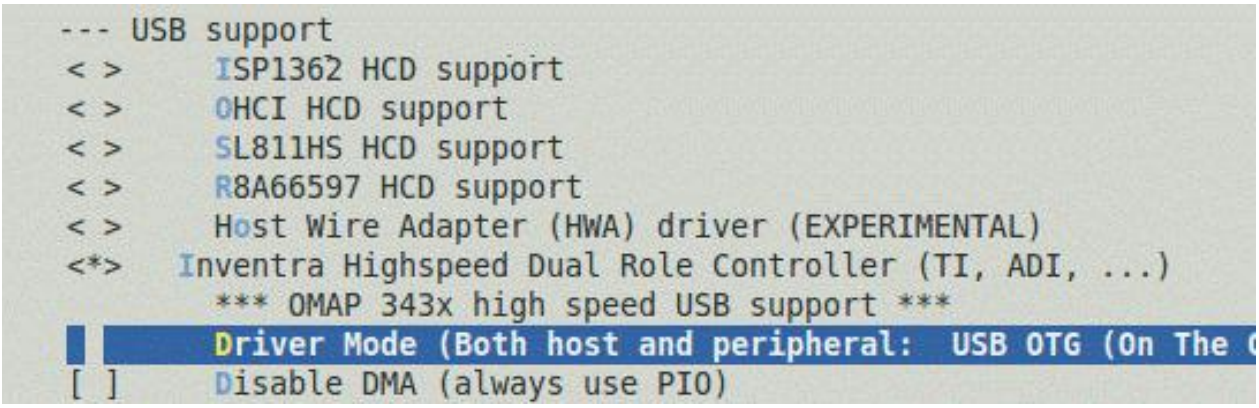
adb over USB Ethernet (Ethernet over USB)

- Make sure that the mini-usb cable is connected between the host usb port and the target's USB OTG port.
- Configure the Linux kernel to use Ethernet gadget. Enable USB support, configure the Inventra controller, and add USB gadget support.

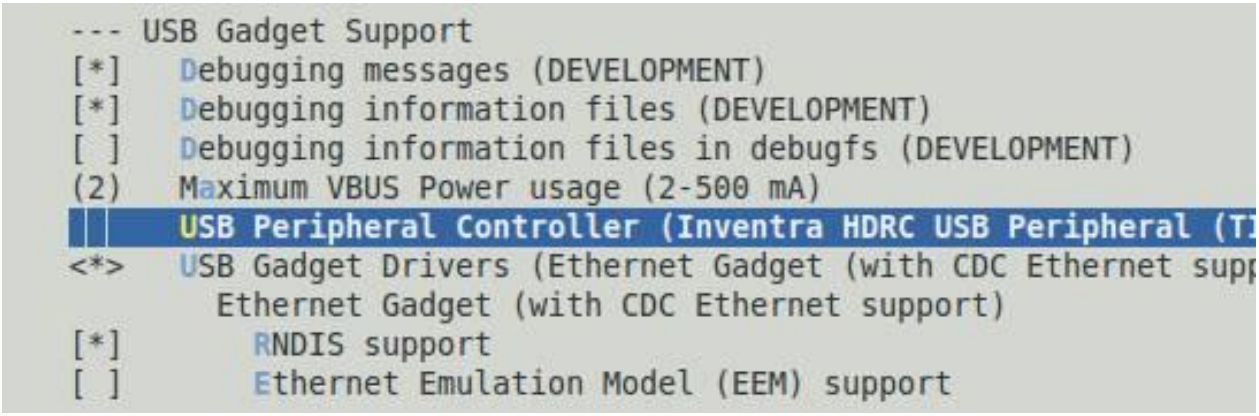
IMPORTANT NOTE: Inventra configuration must occur in two places as shown in non-highlighted lines of the screen shots below.

```
    #> make ARCH=arm CROSS_COMPILE=arm-eabi- menuconfig
```

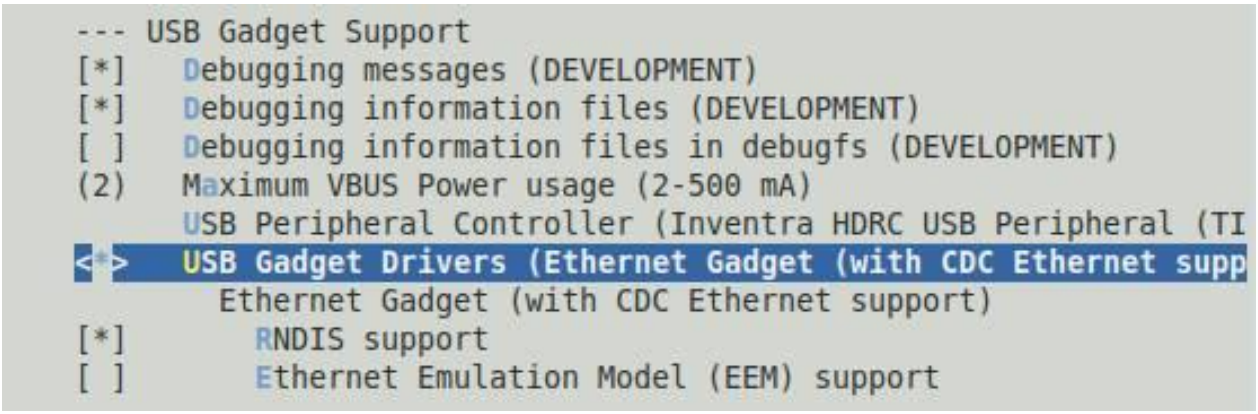
Device Drivers --- USB Support



Device Drivers --- USB Support --- USB Gadget Support



Device Drivers --- USB Support --- USB Gadget Support --- Enable Gadget Ethernet support



- Build the Kernel with the above configuration changes and use the uImage to boot the board. Refer to Kernel compiling instructions above.
- Establish network connection
  - Assign an IP address to the usb ethernet adapter.

The USB network gadget g_ether is named usb0 (instead of eth0 or other network interface names). The normal set of Ethernet configuration tools should work, such as ifconfig, netstat, and route.

For example, the following commands will assign the network address 192.168.194.2 to the target. Run this on the target:

```
$> ifconfig usb0 192.168.194.2 netmask 255.255.255.224 up
```

On Host machine, run the following commands to establish the connection to the target:

```
$> sudo ifconfig usb0 192.168.194.1 netmask 255.255.255.224 up
$> sudo route add 192.168.194.2 dev usb0
```

The target and the host machine should be connected, run ping command to test the same:

```
$ ping -c 3 192.168.194.2
PING 192.168.194.2 (192.168.194.2) 56(84) bytes of data.
64 bytes from 192.168.194.2: icmp_seq=1 ttl=64 time=6.08 ms
64 bytes from 192.168.194.2: icmp_seq=2 ttl=64 time=0.511 ms
64 bytes from 192.168.194.2: icmp_seq=3 ttl=64 time=0.485 ms
--- 192.168.194.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.485/2.361/6.089/2.636 ms
```

- Establish ADB connection

On the host machine execute following commands to establish adb connection

```
$ export ADBHOST=<target's ip address>
$ adb kill-server
$ adb start-server
```

Verify the connection by executing

```
$ adb devices
```

If connected, device name should be listed as a "emulator"

```
$ adb devices
List of devices attached
```

```
    emulator-5554    device
    $ adb shell
```

adb over Ethernet

- Make sure Ethernet port on board and host machine are connected to the network
- Check Ethernet configuration for the board

```
    target #> netcfg

       lo       UP    127.0.0.1       255.0.0.0       0x00000049

       eth0     UP    172.24.190.59   255.255.252.0   0x00001043
```

- If Ethernet was not configured, configure Ethernet of the board using ifconfig/netcfg as shown below.

```
    target #> netcfg eth0 dhcp
```

- Configure the ADB Daemon to use an ethernet connection using setprop as shown below.

```
    target #> setprop service.adb.tcp.port 5555
```

- If network is configured successfully (above steps) then Restart service adbd on the target,

```
    target #> stop adbd
    target #> start adbd
```

- On the host machine use following commands to establish adb connection

```
    $> export ADBHOST=<target's ip address>
    $> adb kill-server
    $> adb start-server
```

- Verify for device connectivity, by executing the following commands

```
    $> adb devices If connected, you'll see the device name listed
as a "emulator"
```

```
    $> adb devices
```

```
    If connected, find the device name listed as a "emulator"
```

```
    List of devices attached
    emulator-5554    device
    $ adb shell
```

For more information about adb commands, see Android Debug Bridge page at http:// developer. android. com/ guide/developing/tools/adb.html

adb over USB on Windows Machine

Follow the below instructions to get ADB over USB work on a Windows PC

- Download latest Android SDK

(http://developer.android.com/sdk/index.html) and uncompress it in a local folder (i.e. c:\android_sdk).

- Optionally, you may want to add the location of the SDK's primary tools directory to your system PATH. Right-click on My Computer, and select Properties. Under the Advanced tab, hit the Environment Variables button, and in the dialog that comes up, double-click on Path (under System Variables). Add the full path to the tools\ directory to the path.
- Download Android USB Driver

(https://dl-ssl.google.com/android/repository/usb_driver_r03-windows.zip) and uncompress it in a local folder (i.e. c:\android_sdk\usb_driver)

- Edit (or create and then edit if it doesn't already exist) file in

"%USERPROFILE%\.android\adb_usb.ini":

```
  echo 0x18D1 > "%USERPROFILE%\.android\adb_usb.ini"
```

- Edit android_winusb.inf to match EVM/Beagle vendor and product ids:

Under [Google.NTx86] section add:

```
  ;TI EVM
  %SingleAdbInterface%        = USB_Install, USB\VID_18D1&PID_9018
  %CompositeAdbInterface%     = USB_Install,
USB\VID_18D1&PID_9018&MI_01
```

Note: Be careful to add it under Google.NTx86 and not under Google.NTamd64 unless your machine is AMD 64 bits. If you skip this step you won't be able to later install the driver as windows will reject it.

- Boot the board as normal and wait until shell prompt is available (micro-B USB cable must be disconnected).
- Connect micro-B USB cable between board and Windows PC.
- If it is proceeding as planned, Windows will tell you it found a new hardware asks you to install the driver. Install driver that was downloaded as described in step 3 above:

Answer "No, not this time" to the question about running Windows Update to search for software.

  - Choose "Install the hardware that I manually select from a list (Advanced)" this is the 2nd option, then click "Next"
  - Select "Show All Devices", then click "Next"
  - You are going to see a grayed-out text box with "(Retrieving a list of all devices)", click the "Have Disk..." button
  - Browse" to your driver folder (c:\android_sdk\usb_driver). It will be looking of a .inf file so select "android_winusb.inf" and click "Open" then "OK". It's the only file there so you shouldn't go wrong.
  - Select "Android ADB Interface" then click the "Next" button.
  - A warning will appear, answer "Yes" but read the warning anyway.
  - Click the "Close" when the wizard is completed.
- Disconnect and reconnect micro-B USB cable from Board(probably reboot it as well).
- Open command prompt and restart adb server just to make sure it is in a proper state:

```
  adb kill-server
  adb start-server
```

- List the attached devices with "adb devices". It should show your board/device with a random number.
- Type "adb shell". You should see the "#" indicating it works.

Running Applications

The root File System provided in this DevKit releases contains only standard Android components and applications. User might be interested to download & run android applications (.apk) available in the market. The below procedure gives the steps to be followed to download any .apk file to the board and run it on the platform.

Installing (.apk files) application on Target Platform

- From the host: You can use adb tool for package installation.

```
   $> adb install <package>.apk.
```

NOTE: Use -s option with the adb tool, to install the package on external storage.

On successful installation adb tool will report SUCCESS on host terminal, and the application would be listed on the android main menu.

Un-installing applications (.apk) using adb

- To un-install non-default components (that were installed later)
  - Method 1: On the host machine execute the following

```
   $> adb uninstall <package>.apk
```

  - Method 2: On target:

```
   Main menu -> Menu -> Settings -> Applications -> Manage
applications -> Find the package
   Tap on it -> Uninstall -> OK -> OK
```

  - On successful removal, the application would have been removed from the android main menu. All the short-cuts to the application also removed.
- To un-install default components, use the following commands from abd on host machine

```
   $ adb shell
   #rm /system/app/app.apk
```

On successful removal, the application would have been removed from the android main menu.

Setup ADB for application Debugging

ADB and Eclipse, with ADT(Android Development Tools plug-in) allow users to create and debug Android applications. Follow Developing In Eclipse, with ADT at http:/ / developer. android. com/ guide/ developing/ eclipse-adt.html

Steps to connect Eclipse to the board.

- Setup the adb connection with the board by following the instructions given above in connecting board ...

```
   Verify the connectivity by executing
   $ adb devices
```

- Open Eclipse IDE. Eclipse, with ADT plugin enable users to
  - Create an android project.

- Build and Run the project on a connected board.
- Debug the project using the Debug perspective.
- Use DDMS (Dalvik Debug Monitor Server) to monitor the connected board.

For more detailed and complete information on the above follow Developing In Eclipse, with ADT at http://developer.android.com/guide/developing/eclipse-adt.html

- Open DDMS(Dalvik Debug Monitor Server) perspective. This DDMS perspective can be opened from the eclipse menu via:

```
Window -> Open Perspective -> Other -> DDMS;
Click on OK
```

- DDMS provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process, and radio state information,incoming call and SMS spoofing, location data spoofing, and more.
- For more information on DDMS and to use it, follow Using the Dalvik Debug Monitor page at http://developer.android.com/guide/developing/tools/ddms.html

Copy any files to and from the board over ADB

- Using the adb commands "pull" and "push" user can copy files to and from the board.
- Unlike the install command, which only copies an .apk file to a specific location, the pull and push commands let you copy arbitrary directories and files to any location on the board.
- To copy a file or directory (recursively) from the board, use

```
adb pull <remote> <local>
```

- To copy a file or directory (recursively) to the board, use

```
adb push <local> <remote>
```

In the commands, <local> and <remote> refer to the paths to the file or directory on your development host (local) and on the target instance (remote).

```
Here's an example:
adb push foo.txt /sdcard/foo.txt
```

Adobe Flash 10 Integration

The Android version of Flash10 that runs on GingerBread is now available for customer download (by registration) at, http://focus.ti.com/docs/toolsw/folders/print/adobeflash-a8.html

The below steps give the procedure to download the Adobe Flash 10 library for Android GingerBread and installing the same in File system.

- Download the flashplayer installer “Flash10.1_Android_Webkit_Plugin-0.4-Linux-x86-Install.bin” from http://focus.ti.com/docs/toolsw/folders/print/adobeflash-a8.html
- Execute the installer

```
  #> ./ Flash10.1_Android_Webkit_Plugin-0.4-Linux-x86-Install.bin
  Will result in following instruction, press "Y"
```

```
  This will install Flash10.1 Android Webkit Plugin on your computer.
Continue? [n/Y] Y
  Select the source install location
```

```
Where do you want to install Flash10.1 Android Webkit Plugin?
[/home/user/flash10_androidplugin] /home/user/flash10_androidplugin
```

```
Installing Flash10.1 Android Webkit Plugin...
Installing Program
Files...

Installation complete.
After Installation the following directory structure is resulted
```

- Change to Flash installed directory on Host PC

```
#> cd flash10_androidplugin
#> ls
install_flash_player.apk  uninstall
```

- Install flash player plug in on target via adb

```
#> adb install install_flash_player.apk
```

- Do the browser configuration
  - Explained in the below section
- Test the Adobe Flash installation
  - Browse the link http://www.adobe.com/software/flash/about/
  - Should display Adobe Flash Player Successfully Installed

Compatibility Test Suite (CTS)

This section describe the procedure to run CTS on any platform.

- Pre-requisites
  - Download and extract (untar) the CTS package from http://dl.google.com/dl/android/cts/android-cts-2.3_r1-x86.zip
  - Download and extract (untar) Google Android SDK from http://dl.google.com/android/android-sdk_r06-linux_86.tgz
  - NOTE: Only the SDK mentioned on the above link will work with CTS.
- Setup an ADB connection between Host and platform as mentioned in ADB section above.
- Setup your platform to run the accessibility tests:
  - adb install -r android-cts/repository/testcases/CtsDelegatingAccessibilityService.apk
  - On the device enable Settings > Accessibility > Accessibility > Delegating Accessibility Service
- Launch the CTS.
  - Edit android-cts/tools/startcts to point SDK_ROOT to android sdk installation location.
  - Run ./tools/startcts
  - On CTS prompt check the available plans

```
cts_host > ls –plan
```

  - Start a specific Test Plan

```
cts_host > start --plan <plan name>
```

Once all the tests are executed, the results can be browsed in an browser by opening [android-cts/repository/results/session-name/testResult.xml] and use the results to adjust your design.

- NOTE: Sometimes when CTS is restarting the board, adb connection to CTS, may not happen automatically. In that case, execute the following command on the console, soon after the board has restarted.

```
#> stop adbd;sleep 1;start adbd;
```

Configuring Android Applications

Browser Configuration

To browse web pages user should configure the Internet connection as given below.

```
#> netcfg eth0 dhcp
#> getprop net.eth0.dns1
```

This prints the dns for the ethernet port, do the following to configure the DNS entries on board.

```
#> setprop net.dns1 <your_dns_server_ip>
```

If the platform is behind proxy, then following command should be executed to set the proxy settings

```
#> setprop net.gprs.http-proxy http://proxyurl:80
```

NOTE: If network is behind a proxy, in this DevKit release, we have NOT found a method to set the proxy server. We tried using "setprop net.eth0.http-proxy hostname:port" and "setprop net.gprs.http-proxy hostname:port", but neither could get us through the proxy. Also, the option of adding an entry of (99,'http_proxy','hostname:port") to the 'system' and 'secure' tables in the /data/data/com.android.providers.settings/databases/settings.db database has also been tried, but failed.

USB Mass Storage

The Android GingerBread 2.3 supports USB Mass storage functionality, however the external storage can be mounted either on SD Card or a USB mass storage device. The user is allowed to choose one of the two options.

By default the TI Android DevKit chooses SD card as the external storage device.

If a user is interested to use storage over USB (USB mass storage) then following changes have to be done in the default root filesystem

1)Create some directory on Android Filesystem via adb shell or console

```
# mkdir /partition
```

2) Insert the usb mass storage device, Assume the device node created at /dev/block/sda1 and the device has fat partition.

```
#mount –t vfat /dev/block/sda1 /partition
```

3) Now usb mass storage device is mounted at /partition. You can careate browse the filr

Limitations

- Auto mounting usb mass storage device not supported
- The gallery app/Android Settings doesn't recognize the mass storage mounted

SD Card Recommendations

Some brands or models of SD cards are observed with poor performance on AM37x EVMs and other platforms. The symptom could be one or some of the followings.

- the boot-up time is much longer than normal (3x of normal or even longer);
- the reaction of UI operations is unacceptably delayed;
- the Gallery app cannot find the media files to create the albums;
- the video playback is sluggish.

The table below lists the SD cards tested which have no issue on performance.

	Brand/Model	Type	Class	Capacity
	SanDisk	SDHC	4	4GB
	SanDisk Ultra	SDHC	4	4GB
	SanDisk Ultra	SD	4	2GB
	Sony	SDHC	4	4GB
	Sony	SD	4	2GB
	Sony	micro SDHC	4	4GB

The table below lists the SD cards tested which have **poor** performance.

	Brand/Model	Type	Class	Capacity
	HP Invent	SDHC	4	4GB
	Kingston	SDHC	4	4GB
	Kingston	micro SDHC	4	4GB
	Lexar MULTI-USE	SDHC	4	4GB
	Lexar PLANTINUM II	SDHC	6	4GB

	PNY Optima	SDHC	4	4GB

Versioning

This is Release DevKit-V2.3 The release is available from http:/ / software-dl. ti. com/ dsps/ dsps_public_sw/ sdo_tii/TI_Android_DevKit/TI_Android_GingerBread_2_3_DevKit_1_0/index_FDS.html
The release notes is available at http:/ / processors. wiki. ti. com/ index. php/ TI-Android-GingerBread-2. 3-DevKit-1.0_ReleaseNotes

Technical Support and Product Updates

For further information or to report any problems, contact http://e2e.ti.com/android or http://support.ti.com.
For community support join http://groups.google.com/group/rowboat
For IRC #rowboat on irc.freenode.net

References

[1] http://focus.ti.com/docs/toolsw/folders/print/tmdsevm3530.html
[2] http://beagleboard.org
[3] http://focus.ti.com/docs/toolsw/folders/print/tmdsevm3517.html
[4] http://focus.ti.com/docs/toolsw/folders/print/tmdxevm3715.html?DCMP=am37x_060710&HQS=Other+OT+am37xprtf
[5] http://developer.android.com/sdk/installing.html#troubleshooting
[6] http://android.git.kernel.org/?p=platform/prebuilt.git;a=tree;f=linux-x86/toolchain/arm-eabi-4.4. 0;h=0f1763c115040381493278dde810a69e4bc37a76;hb=HEAD
[7] http://processors.wiki.ti.com/index.php/TI-Android-GingerBread-2.3-DevKit-1.0_UserGuide#adb_over_USB_on_Windows_Machine
[8] http://www.yaffs.net/howto-incorporate-yaffs-linux

Article Sources and Contributors

TI-Android-GingerBread-2.3-DevKit-1.0 UserGuide *Source*: http://processors.wiki.ti.com/index.php?oldid=55237 *Contributors*: Alejandro, Arunjoseph, Vishveshwarbhat

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:Mt9v113.jpeg *Source*: http://processors.wiki.ti.com/index.php?title=File:Mt9v113.jpeg *License*: unknown *Contributors*: Satish npatel

File:240px-Android_bt_on_1.png *Source*: http://processors.wiki.ti.com/index.php?title=File:240px-Android_bt_on_1.png *License*: unknown *Contributors*: Khasim

File:240px-Android_bt_pair_2.png *Source*: http://processors.wiki.ti.com/index.php?title=File:240px-Android_bt_pair_2.png *License*: unknown *Contributors*: Khasim

File:android_bt_share.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Android_bt_share.png *License*: unknown *Contributors*: Vishveshwarbhat

File:android_bt_incoming_taskbar.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Android_bt_incoming_taskbar.png *License*: unknown *Contributors*: Vishveshwarbhat

File:android_bt_incoming_message.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Android_bt_incoming_message.png *License*: unknown *Contributors*: Vishveshwarbhat

File:android_bt_incoming_popup.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Android_bt_incoming_popup.png *License*: unknown *Contributors*: Vishveshwarbhat

File:Android bt a2dp headset.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Android_bt_a2dp_headset.png *License*: unknown *Contributors*: Vishveshwarbhat

File:Android bt headset.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Android_bt_headset.png *License*: unknown *Contributors*: Vishveshwarbhat

File:Android-bt-options.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Android-bt-options.png *License*: unknown *Contributors*: Vishveshwarbhat

File:Android-disable-a2dp.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Android-disable-a2dp.png *License*: unknown *Contributors*: Vishveshwarbhat

File:Light_bulb_icon.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Light_bulb_icon.png *License*: unknown *Contributors*: DanRinkes, PagePusher

Image:Android USB ADB ENABLE.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Android_USB_ADB_ENABLE.JPG *License*: unknown *Contributors*: Khasim

Image:Android USBGadget ADB.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Android_USBGadget_ADB.JPG *License*: unknown *Contributors*: Khasim

Image:Android USBEthernet ADB.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Android_USBEthernet_ADB.JPG *License*: unknown *Contributors*: Khasim

File:Sandisk-C4-4GB.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Sandisk-C4-4GB.jpg *License*: unknown *Contributors*: BinLiu

File:Sandisk-ultra-C4-16GB.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Sandisk-ultra-C4-16GB.jpg *License*: unknown *Contributors*: BinLiu

File:SONY-C4-4GB.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:SONY-C4-4GB.jpg *License*: unknown *Contributors*: BinLiu

File:SONY-C4-2GB.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:SONY-C4-2GB.jpg *License*: unknown *Contributors*: BinLiu

File:SONY-micro-C4-4GB.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:SONY-micro-C4-4GB.jpg *License*: unknown *Contributors*: BinLiu

File:Kingston-C4-4GB.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Kingston-C4-4GB.jpg *License*: unknown *Contributors*: BinLiu

File:Lexar-Multi-Use-C4-4GB.jpeg *Source*: http://processors.wiki.ti.com/index.php?title=File:Lexar-Multi-Use-C4-4GB.jpeg *License*: unknown *Contributors*: BinLiu

File:Lexar-PlatinumII-C6-4GB.jpeg *Source*: http://processors.wiki.ti.com/index.php?title=File:Lexar-PlatinumII-C6-4GB.jpeg *License*: unknown *Contributors*: BinLiu

File:PNY-Optima-C4-4GB.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:PNY-Optima-C4-4GB.jpg *License*: unknown *Contributors*: BinLiu

License

1. 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.
2. 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.
3. 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 Ssection 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 a 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.
4. 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, MERCHANTIBILITY, 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

1. 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.
2. 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

1. 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.
2. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
3. 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.
4. 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.
5. 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.
6. The rights granted under, and the subject matter referenced, 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.