DM3730/AM3703 Android Gingerbread 2.3.4 BSP User Guide

BSP Documentation

Logic PD // Products
Published: April 2012
Last revised: August 2014

Revision History

REV	EDITOR	DESCRIPTION	APPROVAL	DATE
A	JY	-Initial release	JY, XC	04/03/12
B	XC	-Section 1.3: Added note; -Section 4.1: Updated commands to install android-10, rather than android-13; -Section 4.3.3: Added Step 1; -Section 5.1: Added Step2; Added specific instructions to Step 3 for use with v1.0 or v1.2 of the BSP; -Section 5.2: Added important note; -Section 5.3: Updated command in Step 6 for v1.2 of the BSP; -Section 7.2.2: Updated JDK version to 32; added second important note; Changed *profile* references to *bashrc*; updated commands; -Section 7.3: Added specific instructions to Step 2 for use with v1.0 v1.2 of the BSP; -Section 7.4: Updated commands in Step 1 for JDK version 32	JY, LN, SO	06/19/12
C	XC, SO	-Throughout: Added references to AM3703 SOMs, as the v1.3 release of the BSP now supports these modules; -Added Section 1.1; -Table 3.1: Updated description of how to use power suspend/resume key on development kits; -Section 4.1: Updated steps and commands throughout; -Section 4.2: Added Step 2 to ensure necessary software has been installed; added note to Step 3 about location of .inf file; added Steps 6 through 0; -Section 4.3.3: Added information in Step 6 about the time necessary for the wired LAN speed negotiation to complete; -Section 5.1: Added command for v1.3 of the BSP to Step 3; updated the command for v1.2 of the BSP in Step 3; added note to Step 4 to reminder users the SD card will be formatted; -Section 6: Added reference to *Install Android SD* article for proper ADB setup; -Added Section 7.2.1; -Section 7.3: Added note about correctly configuring the host PC before proceeding; added command for v1.3 of the BSP to Step 2; updated the command for v1.2 of the BSP in Step 2; -Section 7.4: Added Step 3 for AM3703 users only; -Added Section 7.5, 7.6, and 0; -Removed section that outlined individual steps to build and deploy an Android system	SWE, XC	11/05/12
D	XC	-Table 3.1: Updated key mapping for Menu and Search keys on the DM3730 Torpedo Development Kit for v1.4 release of the BSP; -Section 5.1: Added command for v1.4 of the BSP to Step 3; -Section 7.3: Added command for v1.4 of the BSP to Step 2; -Section 7.5.2: Added Step 6c; added note that U-Boot environmental variables are configured for SD-boot Android image; -Added Section 7.7 and 7.8	SWE, SO	02/04/13
E	AF	Updated URL link for Repo tool in section 7.3. Link now points to a Logic PD hosted copy.	JC, AF	8/7/14

Table of Contents

1 Introduction

This document describes how to work with the Logic PD Android Gingerbread 2.3.4 Board Support Package (BSP) release for supported DM3730/AM3703 SOMs. The BSP includes bootloaders; the Android Linux kernel and root file system; and tools and documentation to ease development, deployment, and execution of Android-based systems.

This document provides instructions on how to run the pre-built SD card, install the Android Software Development Kit (SDK), create a new SD card image, and download and build the complete Android source tree for both the Zoom™ DM3730 Torpedo Development Kit and the Zoom™ DM3730 SOM-LV Development Kit. Please note that the system requirements and time involved to build the Android platform are significantly greater than what is required for Android application development.

1.1 Nomenclature, Notices, and Conventions Used in This Document

- Within this document, DM3730 is used to denote both the DM3730 and AM3703 processors. It is important to note, however, that one major difference does exist between the two processor sub-families: the DM3730 processor has a built-in SGX hardware graphics accelerator, which results in better graphics performance, as compared to that on the AM3703 processor.
- This document covers the DM3730/AM3703 SOM-LV, DM3730/AM3703 Torpedo SOM, and DM3730/AM3703 Torpedo + Wireless SOM. Use of “DM3730 SOM” suggests text that applies to all three platforms; information specific to one platform will call out the precise name.
- Use of “DM3730 Development Kit” suggests text that applies to both the DM3730 SOM-LV Development Kit and DM3730 Torpedo Development Kit; information specific to one development kit will call out the precise name.

1.2 Android Application Development Requirements

See the **System Requirements** link on Android’s *Get the Android SDK* web page[1] for host PC requirements.

1.2.1 Hardware Requirements

- DM3730 Development Kit
  - DB9 null-modem serial cable
  - USB A to mini-AB cable
- Host PC with 2 GB memory and 600 MB available disk space after Eclipse and JDK installation

1.2.2 Supported Operating Systems

- Windows XP (32-bit), Vista (32- or 64-bit), or Windows 7 (32- or 64-bit)
- Mac OS X 10.5.8 or later (x86 only)
- Linux (tested on Ubuntu 10.04 LTS - Lucid Lynx)
- GNU C Library (*glibc*) 2.7 or later

NOTE: 64-bit distributions must be capable of running 32-bit applications. For information about how to add support for 32-bit applications, see the Ubuntu Linux installation notes.

[1] http://developer.android.com/sdk/requirements.html

1.2.3 Supported Development Environments

- Eclipse IDE
  - Eclipse 3.6.2 (Helios) or greater
  - Eclipse JDT[2] plugin (included in most Eclipse IDE packages)
  - If you need to install or update Eclipse, you can download it from the Eclipse website[3]
- Several types of Eclipse packages are available for each platform. For developing Android applications, we recommend that you install one of the following packages:
  - Eclipse IDE for Java Developers
  - Eclipse Classic (versions 3.6.2 and higher)
  - Eclipse IDE for Java EE Developers
  - JDK 6[4] (JRE alone is not sufficient)
  - Android Development Tools plugin[5] (recommended)
  - Not compatible with Gnu Compiler for Java (gcj)
- Other development environments or IDEs
  - Apache Ant[6] 1.8 or later
  - Not compatible with Gnu Compiler for Java (gcj)

1.3 Android Platform Development Requirements

- Ubuntu Linux 10.04 LTS 64-bit

  NOTE: Ubuntu 10.04 64-bit is recommended, as using a newer version of Ubuntu or a 32-bit Ubuntu to build is only experimentally supported at this time. It is not guaranteed to work on branches other than master. For more information, see Android's *Initializing a Build Environment* wiki article.[7]

- 30 GB free disk space
- At least 4 GB memory
- Internet access to download and install the BSP sources and additional software
- Administrator (sudo) privileges on the Linux host PC
- Software packages described above

2 Download Android BSP

- DM3730/AM3703 Android Gingerbread 2.3.4 BSP Pre-Built Binaries[8]
  - X-Loader (*MLO*), U-Boot, Android Linux kernel, boot script, and root file system
- Source
  - git.logicpd.com (This will be covered in Section 7.3 below.)

[2] http://www.eclipse.org/jdt
[3] http://www.eclipse.org/downloads/
[4] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[5] http://developer.android.com/sdk/eclipse-adt.html
[6] http://ant.apache.org/
[7] http://source.android.com/source/initializing.html
[8] http://support.logicpd.com/downloads/1494/

3 Run Android BSP on DM3730 Development Kit

This section will explain how to run the DM3730/AM3703 Android Gingerbread 2.3.4 BSP on a DM3730 Development Kit.

3.1 Boot with Pre-Built Image

The DM3730 Development Kit includes a preformatted SD card that will boot Android. Insert the card into the bootable card slot on the development kit baseboard and slide the power switch to the ON position.

The first boot should be allowed to run to completion and will take much longer, up to a couple of minutes, as it installs optimized versions of code and updates local databases.

There are a few keys that are essential for navigation in the Android environment. These Android keys have been mapped to the baseboard buttons on DM3730 Development Kit as described in Table 3.1 below.

Table 3.1: Application Key Mapping

Android Key	DM3730 SOM-LV Development Kit	DM3730 Torpedo Development Kit	Notes
Reset	S1 : MSTR_nRST	S1 : MSTR_nRST	Press to reset the processor.
Power suspend/ resume	S2 : PMIC:uP_nWAKEUP	S2 : PWRON	Press and release the button to wake up the system from suspend mode. To power down, press and hold for more than one second for the pop-up menu.
Back	S4 : GPIO_111	S3 : GPIO_2	—
Menu	S5: PMIC:gpio.15	S7 : GPIO_178	Beginning with the v1.4 release of the BSP, S7 replaced S4 as the Menu key for the DM3730 Torpedo Development Kit.
Home	S6: PMIC:gpio.7	S6 : GPIO_181	—
Search	—	—	The Search key is not required on Android standard. Therefore, support for that key has been removed for the DM3730 Torpedo Development Kit beginning with the v1.4 release of the BSP.

3.2 Console Support

The console is mapped to the debug UART on the DM3730 Development Kit. Connect a serial cable between the host PC and the kit, and configure the host PC terminal application (e.g., minicom, Tera Term, HyperTerminal) for the correct settings: 115200, 8N1.

3.3 Boot Sequence

To watch the boot process, run the *logcat* command which dumps output from the Linux kernel and the Android logs. Press **Ctrl+C** to quit.

```
$ logcat
```

NOTE: The Android console has limited functionality. If you want the backspace to behave better, change your terminal program to send DEL instead of backspace (e.g., in minicom, press **Ctrl+A**, then **T**, then **B**; or in Tera Term select Setup > Keyboard). You may have a better user experience by using the Android Debug Bridge (ADB) over USB, which is part of the Android SDK.

Alternatively, BusyBox, which is a collection of common utilities, has been included in the BSP for development purposes. Enter the *BusyBox* shell with the following command:

```
$ busybox sh
```

Some BusyBox commands are hidden because the Android toolbox version is found in the path first. You may change the *PATH* environment variable or simply prefix the command with *busybox*. For example, to run the BusyBox version of *ls*, use the following command:

```
$ busybox ls –la
```


4 Install Android SDK

The Android SDK includes several tools, such as the ADB, that are extremely beneficial when interacting with the running system. The ADB is used to establish a connection over USB between the host PC and Android running on the DM3730 Development Kit. If you intend to build and deploy Android applications, you will need to install the Android SDK. You do not need to build the entire Android Platform as described in Section 7.

To begin the installation process, read Android's *Setting Up an Existing IDE* wiki article.[9]

4.1 Linux

1. On Android's *Get the Android SDK* web page,[10] click the **Download for Other Platforms** link and select the SDK package for Linux platforms. Follow the instructions to download and install the SDK.

2. Install the related software packages using the following command:

```
$ sudo apt-get install openjdk-6-jdk ia32-libs
```

3. Add an entry to the *udev* rules and the Android *adb_usb.ini* files so that the DM3730 Development Kit is recognized as an ADB capable device. **NOTE:** The *cat <<'EOF' > 51-android.rules* command below will create the *51-android.rules* file with the contents entered on each line before the *EOF* string.

```
$ cat <<'EOF' > 51-android.rules
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", MODE="0666", GROUP="plugdev"
EOF

$ sudo cp 51-android.rules /etc/udev/rules.d/
$ rm 51-android.rules
$ sudo chmod a+r /etc/udev/rules.d/51-android.rules
$ echo "0x18d1" >> ~/.android/adb_usb.ini
```

4.2 Windows

1. On Android's *Get the Android SDK* web page,[11] click the **Download the SDK ADT Bundle for Windows** link and follow the instructions to download and install the SDK package for Windows platforms.

2. In the SDK manager, make sure you have the following items installed:

   - □ Tools
   - □ Android 4.1 (API16)
   - □ Google USB Driver

3. If you do not have the Google USB driver installed, do so as described in Android's *Google USB Driver* wiki article.[12]

[9] http://developer.android.com/sdk/installing.html
[10] http://developer.android.com/sdk/index.html
[11] http://developer.android.com/sdk/index.html
[12] http://developer.android.com/sdk/win-usb.html

4. Add the following lines to the *android_winusb.inf* file under both the [Google.NTx86] and [Google.NTamd64] sections. The *android_winusb.inf* file was installed in Step 3 above.

```
;LogicPD DM3730/AM3703 Android BSP
%SingleAdbInterface%        = USB_Install, USB\VID_18D1&PID_0001
%CompositeAdbInterface%     = USB_Install, USB\VID_18D1&PID_0001&MI_01
```

5. Edit *C:\Documents and Settings*\<*USER*>*.android**adb_usb.ini* with a suitable text editor that understands UNIX file format (e.g., Textpad, Notepad++) and add the Vendor ID *(0x18d1)* reported by the DM3730 Development Kit.

```
# ANDROID 3RD PARTY USB VENDOR ID LIST --DO NOT EDIT.
# USE 'ANDROID UPDATE ADB' TO GENERATE.
# 1 USB VENDOR ID PER LINE.
0x18D1
```

6. Connect the DM3730 Development Kit Baseboard to the host PC with a USB A to mini-AB cable through the USB OTG port on the baseboard.
7. Navigate to the device manager on your host PC. Right-click the DM3730/AM3703 device that appears in the *Device Manager* window and select "Update Driver."
8. Select "Browse my computer for driver software."
9. Select "Let me pick from a list of device drivers on my computer."
10. Select "Have Disk...."
11. Browse to *extras**google**usb_driver**android_winusb* installed in Step 3 above.

Now you should have the USB ADB driver correctly installed.

4.3 Using ADB

To use the ADB, first connect a USB A to mini-AB cable between the USB OTG port on the DM3730 Development Kit Baseboard and your host PC. Then, follow the appropriate instructions below.

4.3.1 ADB Setup for Windows

1. Open a command shell by selecting Start > Run and typing *cmd*.
2. Connect to a shell on the device using the ADB and list the files.

```
> cd C:\Program Files\Android\android-sdk\platform-tools
> adb devices
> adb shell
# ls
# exit
```


4.3.2 ADB Setup for Linux

1. Open a command shell by selecting Applications > Accessories > Terminal.
2. Connect to a shell on the device using the ADB and list the files.

```
$ cd ~/android-sdk-linux/platform-tools
$ ./adb devices
$ ./adb shell
# ls
# exit
```

4.3.3 ADB Usage

The ADB tool can be used to copy files to and from the device, install applications, send keystrokes, and much more. See Android's *Android Debug Bridge* wiki article[13] for more information.

1. First, add the ADB tool to your path.

```
$ export PATH=$PATH:$HOME/android-sdk-linux/platform-tools
```

2. Start the calculator using the touch screen on the DM3730 Development Kit and then send text to the current input field over ADB.

```
$ adb shell input text "1234"
```

3. Send the back button (see Android's *KeyEvent* wiki article[14] for additional key codes).

```
$ adb shell input keyevent 4
```

4. Copy a file from the device to the host PC.

```
$ adb pull /init.rc /tmp/init.rc
```

5. Deploy your own Android application.

```
$ adb install MyGreatApplication.apk
```

6. ADB over Ethernet and set up the target device (use the serial console to restart *adbd*). Wait approximately fifteen to thirty seconds for the wired LAN speed negotiation to complete after the Ethernet cable has been plugged into the DM3730 Development Kit.

```
# getprop dhcp.eth0.ipaddress
<target's ip address>
```

[13] http://developer.android.com/guide/developing/tools/adb.html
[14] http://developer.android.com/reference/android/view/KeyEvent.html


```
# setprop service.adb.tcp.port 5555
# stop adbd
# start adbd
```

7. ADB over Ethernet and set up the host PC.

```
$ export ADBHOST=<target's ip address>

$ adb kill-server
$ adb start-server
$ adb devices ; # The device name will be listed as emulator-5554
List of devices attached
Emulator-5554  device
$ adb shell
```


5 Create Bootable SD Card

If you wish to create a new bootable SD card with the Android binaries on it, there are many methods to do so. Choose from one of the following methods detailed below.

NOTE: The bootable SD card has both FAT32 and Linux partitions. Since Microsoft Windows does not support this type of partition scheme, use a Linux PC to create the bootable SD card.

- Use a script to write the files (see Section 5.1).
- Use *fdisk* or *sfdisk* and *mkfs* to create and format partitions (see Section 5.2).
- Use a disk utility to create and format partitions (see Section 5.3).

5.1 Use Script to Write Files (Recommended)

The shell script *build.sh* is available with the DM3730/AM3703 Android Gingerbread 2.3.4 BSP Pre-Built Binaries download (see Section 2 for a link) to simplify the process of creating an SD card.

1. Change the default shell from *dash* to *bash*.

```
$ sudo dpkg-reconfigure dash
      #(select No to use bash instead of dash as your shell)
```

2. Install the *uboot-mkimage* software utility.

```
$ sudo apt-get install uboot-mkimage
```

3. Ensure *build.sh* is executable.
   - For users evaluating the DM3730/AM3703 Android Gingerbread 2.3.4 BSP v1.4, open a new terminal and run the commands below to ensure *build.sh* is executable.

```
$ cd ~
$ tar xvf gingerbread_G2.3.4v1.4.tar.gz
$ cd gingerbread_G2.3.4v1.4
$ chmod a+x build.sh
```

   - For users evaluating the DM3730/AM3703 Android Gingerbread 2.3.4 BSP v1.3, open a new terminal and run the commands below to ensure *build.sh* is executable.

```
$ cd ~
$ tar xvf gingerbread_G2.3.4v1.3.tar.gz
$ cd gingerbread_G2.3.4v1.3
$ chmod a+x build.sh
```


- For users evaluating the DM3730 Android Gingerbread 2.3.4 BSP v1.2, open a new terminal and run the commands below to ensure *build.sh* is executable.

```
$ cd ~
$ tar xvf gingerbread_G2.3.4v1.2.tar.gz
$ cd gingerbread_G2.3.4v1.2
$ chmod a+x build.sh
```

- For users evaluating the DM3730 Android Gingerbread 2.3.4 BSP v1.0, open a new terminal and run the commands below to ensure *build.sh* is executable.

```
$ cd ~
$ tar xvf Gingerbread_G2.3.4v1.0_demo.tar.gz
$ cd Gingerbread_G2.3.4v1.0_demo
$ chmod a+x build.sh
```

WARNING: Before proceeding, you MUST determine the correct device name for the SD card or you will overwrite the wrong device (e.g., primary hard drive, USB drive, backup drive, phone, camera).

4. Run the following command, substituting *SD_DEVICE_NAME* with your correct device name. **NOTE:** The following command formats the SD card.

```
$ ./build.sh -F
sdb is 4025483264 bytes – USB SD Reader
sdd is 1967128576 bytes – Flash Reader
sde is 1977614336 bytes – Memory Card Adap
sdf is 3965198144 bytes – <Your Phone!>
Enter device: SD_DEVICE_NAME
[sudo] password for user: ****************
Unmounting /media/boot
Unmounting /media/rootfs
Setting up SD_DEVICE_NAME

Do you wish to continue? (y/N) y
Partitioning SD_DEVICE_NAME.
[sudo] password for user: ****************
mke2fs 1.41.11 (14-Mar-2010)
```

5. Deploy the Android Gingerbread image to the SD card with the *build.sh* script; this will enable the SD card to boot to the Android Gingerbread OS.

```
$ ./build.sh -S
```

As an alternative to Step 5, an SD card may be created that will erase the NAND partition on the SOM and install the bootloaders, Linux kernel and Android root file system in NAND. To do so, insert the SD card and power on the development kit. After flashing the SOM, the SD card may be removed and the system will boot from the internal NAND.

```
$ ./build.sh -N
```

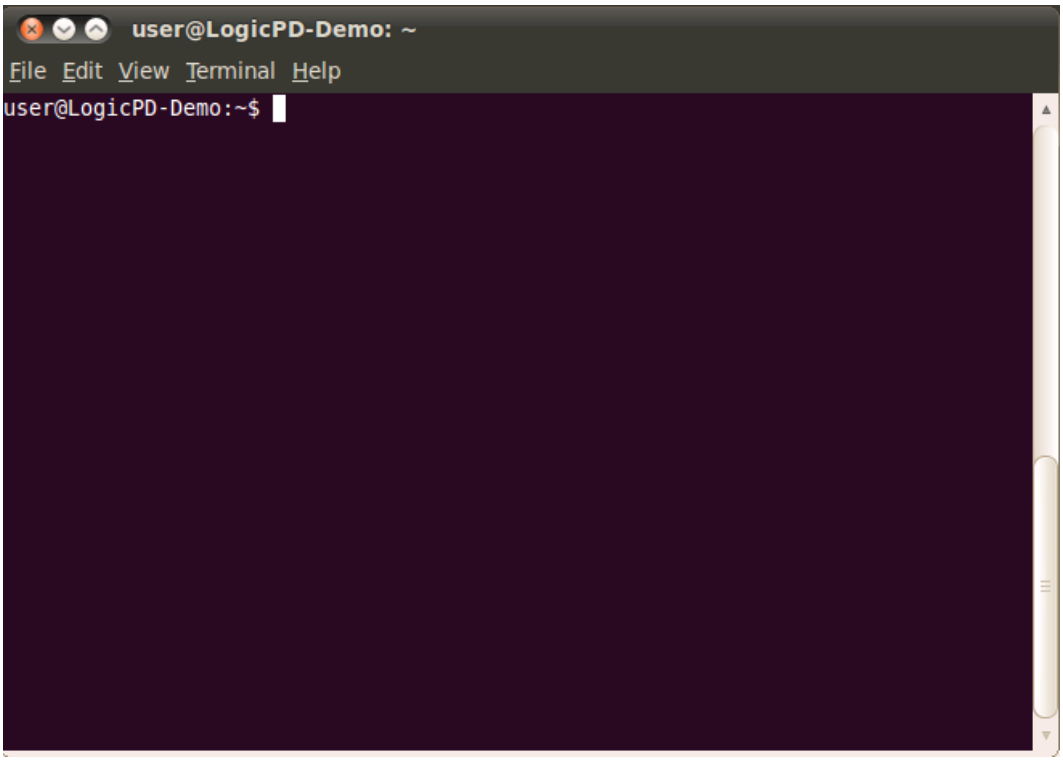

5.2 Use *fdisk* or *sfdisk* and *mkfs* to Create and Format Partitions

Experienced Linux users may prefer to use *fdisk* or s*fdisk* to create the partitions and *mkfs* to format the partitions. The resulting card should have:

- 255 heads, 63 sectors/track, X cylinders
  - OMAP ROM code requires 255 heads and 63 sectors/track
  - Units = cylinders of 16065 * 512 = 8225280 bytes
  - X= disk bytes/8225280, rounded down to nearest integer
- Partition 1: FAT32, (bootable) 64 MB, name: *boot*
  - Contents: *x-load.bin.ift* (*MLO)*, *u-boot.bin*, *uImage*, and *boot_sd.scr*
  - Copy *x-load.bin.ift* as *MLO* first, sync, and then copy the rest of the files
- Partition 2: EXT3, ≥192 MB, name: *rootfs*
  - Contents: extracted *root.tar.bz2,system.tar.bz2,* and *userdata.tar.bz2*
- Partition 3: FAT32 (optional), name: *data*

5.3 Use Disk Utility to Create and Format Partitions

1. Open the terminal application from the Ubuntu menu by selecting Applications > Accessories > Terminal.



WARNING: Before proceeding, you MUST determine the correct device name for the SD card in the following commands or you will overwrite the wrong device (e.g., primary hard drive, USB drive, backup drive, phone, camera). The device name may be something like */dev/sdb* or */dev/mmcblk0* and the partitions may be something like */dev/sdb1*, */dev/sdb2*, */dev/mmcblk0p1*, or */dev/mmcblk0p2*. If you do not know the correct device names, select System > Disk Utility to view the attached devices and partitions.

2. Run the following command, substituting *SD_DEVICE_NAME* with your correct device name.

```
$ sudo fdisk /dev/SD_DEVICE_NAME –l

Disk /dev/sdz: 4012 MB, 4012900352 bytes
```

3. Determine the new number of cylinders by dividing the disk size by (255*63*512). In this example, the equation would be *4012900352/(255*63*512)=487.87*. Drop the fractional part and the resulting number of cylinders is 487 for this card.

4. Follow the commands below to format the bootable SD card, substituting *SD_DEVICE_NAME* with your correct device name.

```
$ umount /dev/SD_DEVICE_NAME*

$ sudo fdisk /dev/SD_DEVICE_NAME
Command (m for help): p

Disk /dev/SD_DEVICE_NAME: 4012 MB, 4012900352 bytes
124 heads, 62 sectors/track, 1019 cylinders
Units = cylinders of 7688 * 512 = 3936256 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x17cc5980

   Device Boot      Start         End      Blocks   Id  System

###
# NOTE: Delete any partitions before entering expert mode
###

Command (m for help): x

Expert command (m for help): h
Number of heads (1-256, default 124): 255

Expert command (m for help): s
Number of sectors (1-63, default 62): 63
Warning: setting sector offset for DOS compatibility

###
# NOTE: Calculate the correct cylinders: bytes/(255*63*512)
###

Expert command (m for help): c
Number of cylinders (1-1048576, default 1019): 487

Expert command (m for help): r
```



```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-487, default 1):
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-487, default 487): 9

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (10-487, default 10):
Using default value 10
Last cylinder, +cylinders or +size{K,M,G} (10-487, default 487): 257

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (258-487, default 258):
Using default value 258
Last cylinder, +cylinders or +size{K,M,G} (258-487, default 487):
Using default value 487

Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))

Command (m for help): a
Partition number (1-4): 1

Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): c
Changed system type of partition 3 to c (W95 FAT32 (LBA))

Command (m for help): p

Disk /dev/SD_DEVICE_NAME: 4012 MB, 4012900352 bytes
255 heads, 63 sectors/track, 487 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x17cc5980

   Device Boot      Start         End      Blocks   Id  System
/dev/sdz1   *           1           9       72261    c  W95 FAT32 (LBA)
/dev/sdz2              10         257     1992060   83  Linux
```



```
/dev/sdz3             258         487     1847475    c  W95 FAT32 (LBA)

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
```

5. Remove and re-insert the SD card, then create the file systems on the card. Be sure to use the correct device and partition name – this might be something *like /dev/sdz1* or */dev/mmcblk0p1.*

```
$ sudo umount /dev/SD_DEVICE_NAME*
$ sudo mkfs.vfat –F 32 –n boot /dev/SD_DEVICE_NAME_AND_PARTIION_ONE
$ sudo mkfs.ext3 –L rootfs /dev/SD_DEVICE_NAME_AND_PARTITION_TWO
$ sudo mkfs.vfat –F 32 –n data /dev/SD_DEVICE_NAME_AND_PARTIION_THREE
```

6. Remove and re-insert the SD card. The partitions may auto-mount under Ubuntu. If not, use the disk utility by selecting System > Administration > Disk Utility to mount the partitions. The *MLO* file must be the first file copied to the SD card so the boot ROM can find it. Copy the *MLO* file to the first partition, flush writes to the disk using *sync*, and copy the other files.

```
$ cd ~
$ tar xvf Gingerbread_G2.3.4v1.2.tar.gz
$ cd Gingerbread_G2.3.4v1.2
$ cp -a x-loader/x-load.bin.ift /media/boot/MLO
$ sync
$ cp -a u-boot/u-boot.bin /media/boot/
$ cp -a kernel/arch/arm/boot/uImage /media/boot/
$ cp -a out/target/product/dm3730logic/boot_sd.scr /media/boot/boot.scr
$ sync
```

7. Extract the root file system to the second partition on the SD card and unmount the partitions using the *umount* command.

```
$ export ANDROID_PRODUCT_OUT=`pwd`/out/target/product/dm3730logic
$ cd /media/rootfs
$ sudo tar xjvf ${ANDROID_PRODUCT_OUT}/root.tar.bz2 --numeric-owner
$ sudo tar xjvf ${ANDROID_PRODUCT_OUT}/system.tar.bz2 --numeric-owner
$ sudo tar xjvf ${ANDROID_PRODUCT_OUT}/userdata.tar.bz2 --numeric-owner
$ sudo cp -a ${ANDROID_PRODUCT_OUT}/init_sd.rc init.rc
$ sync
$ cd -
$ umount /media/boot
$ umount /media/rootfs
$ umount /media/data
```


6 Android Application Development

There are many excellent resources for Android application development, including books and web resources. The classic “Hello World” application and instructions for developing can be found on Android’s *Building your First App* wiki article.[15] **NOTE:** If you only intend to build and deploy Android applications, you will not need to build the entire Android Platform as described in Section 7.

Applications can be deployed and debugged directly from the Eclipse development environment or from a command line using the ADB. **NOTE:** Refer to Section 4 for SDK installation and ADB setup instructions before proceeding.

Use the following ADB command to deploy your own Android application:

```
$ adb install MyGreatApplication.apk
```

An example of how to build an application that directly controls a GPIO can be found in Logic PD’s *AN 531 DM3730/AM3703 Android Gingerbread 2.3.4 GPIO Demo*.[16]

7 Android Platform Development

Android platform development requires a Linux or Mac host PC. These instructions assume a host PC with a 64-bit desktop version of Ubuntu 10.04 LTS is being used.

The source code for the BSP is available from Logic PD's public source tree. The first step is to set up your host PC.

7.1 System Requirements

To build the Android platform software, you will need the following items for your host PC:

- Ubuntu 10.04 LTS Desktop 64-bit[17]
- A large amount of RAM (at least 4 GB)
- Root privileges for your login through *sudo*
- Roughly 30 GB free hard drive space
- An SD card reader
- An Internet connection

7.2 Initial Host PC Configuration

7.2.1 Development Software Installation

There are a core set of software packages that must be installed for compiling the Android platform. These packages may be obtained by using the *apt-get* command from a Linux shell.

1. Start a terminal session by selecting Applications > Accessories > Terminal.

[15] http://developer.android.com/resources/tutorials/hello-world.html
[16] http://support.logicpd.com/downloads/1535/
[17] http://releases.ubuntu.com/lucid/ubuntu-10.04.4-desktop-amd64.iso

2. To determine if you are running a 32-bit or 64-bit version, enter the command below.

```
$ uname -m
```

Output of `i686` indicates it is a 32-bit version and `x86_64` indicates it is a 64-bit version. Only the 64-bit version is supported for building the Android platform.

3. Install the required software for your host PC.

```
$ sudo apt-get install git-core gnupg flex bison gperf \
  build-essential zip curl zlib1g-dev libc6-dev \
  lib32ncurses5-dev ia32-libs x11proto-core-dev libx11-dev \
  lib32readline5-dev lib32z-dev \
  libgl1-mesa-dev g++-multilib mingw32 tofrodos python-markdown \
  libxml2-utils xsltproc expect uboot-mkimage libusb-dev openjdk-6-jdk

$ sudo dpkg-reconfigure dash
      #(select No to use bash instead of dash as your shell)
```

7.2.2 Java Development Kit

Android Gingerbread development requires the Oracle (Sun) Java Development Kit (JDK) v1.6. This JDK software package is not available through the standard Ubuntu distribution anymore. This means that some of the previous methods of installing and selecting Java packages (*apt-get install sun-java6-jdk && update-java-alternatives*) will no longer work. Until a long term solution has been worked out, the JDK must be downloaded from the Oracle website.[18]

On the Oracle website provided above, scroll down and find the heading for *Java SE 6 Update xx* (*xx* refers to latest version number). Click on the download button for the JDK. **IMPORTANT NOTE:** Do not select the download for the JRE. You must accept the license agreement before the download link becomes active.

IMPORTANT NOTE: The JDK version on the Oracle website may be updated to a number higher than 32 in the future. The following command sequence serves as an example, assuming the binary file *jdk-6u32-linux-x64.bin* for JDK version 32 has been downloaded to the *~/Downloads* folder.

1. Install the JDK.

```
$  cp ~/Downloads/jdk-6u32-linux-x64.bin ~
$ chmod +x jdk-6u32-linux-x64.bin
$ ./jdk-6u32-linux-x64.bin
```

2. Edit your *.bashrc* to add *JAVA_HOME/bin* to your path so it is available to all programs.

```
$ nano ~/.bashrc
  # Note: add the following lines to the end of the file

if [ -d "$HOME/jdk1.6.0_32" ] ; then
    JAVA_HOME="$HOME/jdk1.6.0_32"
```

[18] http://www.oracle.com/technetwork/java/javase/downloads/index.html


```
    PATH="$JAVA_HOME/bin:$PATH"
fi
```

7.3 Android BSP Source Code

Source code for Android is maintained in multiple Git repositories. Git is a distributed revision-control management system developed by Linus Torvalds initially for Linux kernel development. Repo is a tool developed by Google to help manage multiple Git repositories when building Android.

NOTE: Before proceeding, refer to Section 7.2 to ensure your Ubuntu Linux host PC is correctly configured.

1. Install the *repo* tool and add it to your path.

```
$ mkdir -p ~/bin
$ curl http://git.logicpd.com/mirrors/gerrit.googlesource.com/git-repo/plain/repo > ~/bin/repo$ chmod a+x ~/bin/repo
$ export PATH=~/bin:$PATH
```

   An XML manifest file describes which Git repository, branch, and revision should be used, as well as where each repository will be placed in the working directory. The manifest file may specify a certain revision of each Git repository or it may simply reference the tip of the branch. The manifest file for Logic PD's DM3730/AM3703 Android Gingerbread 2.3.4 BSP release, *LogicPD-Android-G2.3.4v1.*.xml,* specifies the version of each component used to build this BSP.

2. Create a working directory and pull the source code using the *repo* tool.

   NOTE: Pulling these repositories will download nearly 3 GB of data that will occupy over 13 GB of space on your disk after compiling. Ensure you have ample disk space before proceeding.

   - For users evaluating the DM3730/AM3703 Android Gingerbread 2.3.4 BSP v1.4, do so using the commands below.

```
$ cd ~
$ mkdir dm3730logic-gingerbread && cd dm3730logic-gingerbread

$ repo init -u git://git.logicpd.com/projects/logicpd/dm37x-android/manifest.git -m LogicPD-Android-G2.3.4v1.4.xml
```

   - For users evaluating the DM3730/AM3703 Android Gingerbread 2.3.4 BSP v1.3, do so using the commands below.

```
$ cd ~
$ mkdir dm3730logic-gingerbread && cd dm3730logic-gingerbread

$ repo init -u git://git.logicpd.com/projects/logicpd/dm37x-android/manifest.git -m LogicPD-Android-G2.3.4v1.3.xml
```


- For users evaluating the DM3730 Android Gingerbread 2.3.4 BSP v1.2, do so using the commands below.

```
$ cd ~
$ mkdir dm3730logic-gingerbread && cd dm3730logic-gingerbread

$ repo init -u git://git.logicpd.com/manifest.git \
  -b logicpd-gingerbread -m LogicPD-Android-G2.3.4v1.2.xml
```

- For users evaluating the DM3730 Android Gingerbread 2.3.4 BSP v1.0, do so using the commands below.

```
$ cd ~
$ mkdir dm3730logic-gingerbread && cd dm3730logic-gingerbread

$ repo init -u git://git.logicpd.com/manifest.git \
  -b logicpd-gingerbread -m LogicPD-Android-G2.3.4v1.0.xml
```

3. Ensure your directory contains the same source code that was used to build the BSP.

```
$ repo sync
```

4. List the files in the directory; the output should look similar to that included below.

```
$ ls
bionic    build-tools  device      kernel    packages  u-boot
bootable  cts          external    libcore   prebuilt  x-loader
build     dalvik       frameworks  Makefile  sdk
build.sh  development  hardware    ndk       system
```

7.4 Build and Deploy Android System - Build Script

Building and packaging the Android system involves many steps which have been simplified by the *build.sh* shell script included in the DM3730/AM3703 Android Gingerbread 2.3.4 BSP.

The *JAVA_HOME* variable is used to indicate the location of the JDK to the build process. If you installed the JDK before it was removed from Linux distributions, it may already be configured correctly. This document assumes that you have downloaded and installed the JDK in your home directory.

1. Add the *JAVA_HOME* environment variable and include the *bin* directory first in your path so that it is found before any other JDK that may be installed. If you installed the JDK somewhere else, set *JAVA_HOME* to that directory instead.

```
$ export JAVA_HOME="$HOME/jdk1.6.0_32"
$ export PATH="$JAVA_HOME/bin:$PATH"
```


2. Test that the correct *JDK1.6.0_32* is found in your path.

```
$ java -version
java version "1.6.0_32"
Java(TM) SE Runtime Environment (build 1.6.0_32-b05)
Java HotSpot(TM) 64-Bit Server VM (build 20.7-b02, mixed mode)
```

3. **IMPORTANT NOTE:** This step is only for AM3703 SOM users. DM3730 SOM users should skip this step and proceed to Step 4.

   Set an environment variable to skip including SGX driver code in the Gingerbread image because the AM3703 processor does not support the SGX hardware graphics accelerator.

```
$ export SKIP_SGX=1
```

4. Build all components using the *-A* option. This may run for a long time and varies greatly from machine to machine. The *–j2* option will run two concurrent compile jobs which will keep a single processor system quite busy. If you have more processor cores and more memory, you should increase this number. If you do not specify a *-j#* option, the default *build.sh* script will use eight concurrent jobs.

```
$ cd ~/dm3730logic-gingerbread
$ ./build.sh -j2 -A
Building xloader              - took xx.xxx seconds to build
Building uboot_no_env        - took xx.xxx seconds to build
Building uboot                - took xx.xxx seconds to build
Building kernel               - took xx.xxx seconds to build
Building kernel_modules       - took xx.xxx seconds to build
Building android              - took xx.xxx seconds to build
Building wl12xx_modules       - took xx.xxx seconds to build
Building images               - took xx.xxx seconds to build
Deploying to build_out
```

5. Insert an SD card into an SD card reader attached to your host PC and format the SD card using the *-F* option to *build.sh*. Remember to substitute *SD_DEVICE_NAME* with your correct device name.

WARNING: Before proceeding, you MUST determine the correct device name for the SD card or you will overwrite the wrong device (e.g., primary hard drive, USB drive, backup drive, phone, camera).

```
$ ./build.sh -F
sdb is 4025483264 bytes – USB SD Reader
sdd is 1967128576 bytes – Flash Reader
sde is 1977614336 bytes – Memory Card Adap
sdf is 3965198144 bytes – <Your Phone!>
Enter device: SD_DEVICE_NAME
[sudo] password for user: ****************
Unmounting /media/boot
Unmounting /media/rootfs
Setting up SD_DEVICE_NAME
```



```
Do you wish to continue? (y/N) y
Partitioning SD_DEVICE_NAME.
[sudo] password for user: ****************
mke2fs 1.41.11 (14-Mar-2010)
```

6. Deploy the bootloaders, Linux kernel, boot script and Android root file system to the SD card using the *-S* option.

```
$ ./build.sh -S
Deploying to sd
sdb is 4025483264 bytes – USB SD Reader
sdd is 1967128576 bytes – Flash Reader
sde is 1977614336 bytes – Memory Card Adap
sdf is 3965198144 bytes – <Your Phone!>
Enter device: SD_DEVICE_NAME
Mounting bootloader partition
Mounting root partition
Extracting root tarball
Extracting system tarball
Extracting userdata tarball
Filtering init.rc for mtd mount commands
Unmounting bootloader partition
Unmounting root partition
```

Alternatively, an SD card may be created that will erase the NAND partition on the SOM and install the bootloaders, Linux kernel and Android root file system in NAND. To do so, insert the SD card and power on the development kit. After flashing the SOM, the SD card may be removed and the system will boot from the internal NAND.

```
$ ./build.sh –N
```

7.5 Fastboot

The Android fastboot feature is supported on the Android Gingerbread 2.3.4 BSP v1.3 and later. Fastboot is a diagnostic utility used primarily to modify the NAND flash file system on an Android device over a USB connection.

7.5.1 Fastboot Setup on Ubuntu Host PC

Fastboot requires that the Android SDK is installed on a Ubuntu Linux host PC. Please refer to Section 4 for SDK installation instructions before proceeding.

Once the SDK is installed, add an entry into the *udev* rules to configure the Ubuntu host PC to recognize the fastboot connection over a USB cable.

NOTE: The *cat <<'EOF' > 11-android.rules* command below will create the *11-android.rules* file with the contents entered on each line before the *EOF* string. Change *loginname* to your Ubuntu Linux ID.

```
$ cat <<'EOF' > 11-android.rules
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0451", ATTRS{idProduct}=="0000",
MODE="0666", OWNER="loginname"
```



```
EOF

$ sudo cp 11-android.rules /etc/udev/rules.d/
$ rm 11-android.rules
$ sudo chmod a+r /etc/udev/rules.d/11-android.rules
```

For more information on the fastboot feature, please see OMAPpedia's *Android Fastboot* wiki article[19] and Android's *Fastboot* wiki article.[20]

7.5.2 Fastboot Execution

1. Set up a serial console with a serial cable connecting the Ubuntu host PC and the DM3730 Development Kit. See Section 3.2 for additional serial console setup information.
2. Use a USB A to mini-AB cable to connect the Ubuntu host PC and the USB OTG port on the DM3730 Development Kit Baseboard.
3. At boot time, press any key to disrupt the U-Boot countdown and enter a U-Boot command shell.
4. Enter the *fastboot* command at the U-Boot command shell to enter fastboot mode.

```
U-Boot > fastboot
```

   This will allow a host PC to connect to the development kit via a USB OTG connection.

5. On your Ubuntu Linux host PC, the fastboot executable file is located inside the *Android SDK* folder. Enter the folder.

```
$ cd ~/android-sdk-linux/platform-tools
```

6. The fastboot feature allows users to erase and flash individual Android software modules on NAND partitions of the DM3730 SOM using the corresponding binary image files.

   NOTE: The <*path*> feature in the commands below denotes the path on your host PC to the specified code image file. Be sure to replace this with a value specific to your system.

   a. Flash the X-Loader bootstrap code.

```
$ fastboot flash x-loader <path>/x-load.bin.ift
```

   b. Flash the U-Boot bootloader code.

```
$ fastboot flash u-boot <path>/u-boot.bin.ift
```

[19] http://www.omappedia.com/wiki/Android_Fastboot
[20] http://android-dls.com/wiki/index.php?title=Fastboot

c. Erase the U-Boot environment data.

```
$ fastboot erase u-boot-env
```

NOTE: The default U-Boot environment variables are configured for the SD-boot Android image. For the NAND-boot Android image, refer to Section 7.7.2 to add U-Boot environment variables necessary for booting from NAND.

d. Flash the Linux kernel code.

```
$ fastboot flash boot <path>/boot.img
```

e. Flash the Android file system.

```
$ fastboot flash system <path>/system.img
```

f. Flash the Android user data.

```
$ fastboot flash userdata <path>/userdata.img
```

7.6 Wired Ethernet Configuration

A wired Ethernet software driver is included in the Android Gingerbread 2.3.4 BSP release to support the wired Ethernet port on the DM3730 Development Kit. DHCP mode and Ethernet hot-plug capabilities are enabled by default in this Android release.

The Android Gingerbread 2.3.4 BSP release does not include a graphic interface for Ethernet configuration. Users must use a command shell on either the ADB or a serial port console to configure the Ethernet. See Section 3.2 for serial port console setup or Section 4.3 ADB setup.

7.6.1 Display Ethernet Configuration

Use the following command to display the *eth0* up/down state, DHCP IP address, and network mask:

```
# netcfg
…
eth0     UP    10.0.5.18       255.255.252.0   0x00001043
…
```

7.6.2 Disable Ethernet Configuration

Use the following command to disable the Ethernet configuration:

```
# netcfg eth0 down
[  233.389190] eth0: link down
```


7.6.3 Enable Ethernet Configuration

Use the following command to enable the Ethernet configuration:

```
# netcfg eth0 up
[  247.081146] smsc911x smsc911x.0: eth0: SMSC911x/921x identified at
0xd0876009
```

7.6.4 Configure Ethernet in Static IP Mode

DHCP is the default mode supported on the Android Gingerbread 2.3.4 BSP. To switch to static IP mode, set up a U-Boot environment variable using the steps below.

1. At boot time, press any key to disrupt the U-Boot countdown to enter the U-Boot command shell.

2. Add an IP address assignment to the U-Boot variable *otherbootargs* and save the updated variable to non-volatile memory.

```
# setenv otherbootargs ${otherbootargs} ip=<ipaddr>:::<gatewayip>:<netmask>::
# saveenv
```

   At the next boot time, the kernel will configure the wired Ethernet in static IP mode.

3. To switch IP addressing from static IP to the default DHCP mode, empty the *otherbootarg*s definition in U-Boot.

```
# setenv otherbootargs
# saveenv
```

   Or restore the U-Boot environment variables to the default (see section 7.7).

7.7 U-Boot Environment Variables

7.7.1 Default U-Boot Environment Variables

Users can restore the U-Boot environment variables to their default settings by entering the command below under the U-Boot prompt in the serial console. (See Section 3.2 for serial port console setup.) The default U-Boot environment variables are configured for the SD-boot Android image.

```
# env default -f
# saveenv
```


7.7.2 U-Boot Environment Variables for NAND-Boot

The default U-Boot environment variables are configured for the SD-boot Android image. For the NAND-boot Android image, the commands below must be executed under the U-Boot prompt in the serial console. The commands add additional environment variables to the default variables.

```
# setenv kernel_location nand-part
# setenv rootfs_location /dev
# setenv rootfs_type yaffs
# setenv rootfs_device /dev/mtdblock5
# saveenv
```

The *saveenv* command saves all the U-Boot environment variables to a NAND-designated area, ensuring the values are preserved across power cycles.

7.8 Enable GPS Software Support

The Android Gingerbread 2.3.4 BSP supports the GPS feature on the DM3730/AM3703 Torpedo + Wireless SOM. However, GPS is not enabled by default configurations.

To enable GPS support on the DM3730/AM3703 Torpedo + Wireless SOM, open the */device/logicpd/dm3730logic/init.rc* file in the BSP source code. In the file, mainline the three lines below by removing the "#" symbol at the start of each line:

```
…
#service navl_server /system/bin/navd --android_log NAVD  -p3 -nav\"-
c/system/etc/gps/config/pathconfigfile.txt\"
#    user root
#    oneshot
…
```

8 Legal License Information

The copyright license information of all source code repositories used in the Android Gingerbread 2.3.4 BSP can be viewed using either of the following methods:

- By selecting Settings > About phone > Legal information on the DM3730 Development Kit running the Android Gingerbread 2.3.4 BSP
- By accessing the */system/etc/NOTICE.html.gz* file on the Android root file system (SD or NAND)

