PhyCORE-OMAP44xx Jelly Bean Android Development

1 About this Quickstart

This document describes how to install and work with the Jelly Bean Android Board Support Package (BSP) for the phyCORE-OMAP44xx platform. This BSP provides a fundamental software platform for development, deployment and execution on the phyCORE-OMAP44xx.

The Quickstart contains instructions for:

```
Host Setup
Building a BSP (Platform, Kernel, Android Filesystem)
Installing the images on the target (Barebox, Kernel, Android Filesystem)
```

Note:

- For performance reasons, the images should be installed into the target's NAND as described below.
- Be sure to use the sudo commands indicated in the following steps rather than running them as a regular, non-root user on the development machine. As a non-root user, some target files will not have root permissions and the target will not operate properly.

2 Host Setup

The following notes assume a Ubuntu 10.04 64 bit build machine.

2.1 TFTP

TFTP allows files to be downloaded from one machine to another. With most embedded Linux devices, TFTP is an efficient way to boot the kernel during development so that the user does not have to flash a new kernel every time it is modified. It is also helpful when updating images in flash from Barebox. First, start by installing the TFTP server.

```
sudo apt-get install tftpd-hpa
```

Next, files can be accessed from another machine on the same network by simply using the IP address of the host. You must specify a folder where the files will reside on the host by replacing the folder path for TFTP_DIRECTORY with whatever folder you wish to use as your TFTP file storage location, or leave the folder as the default.

sudo gedit /etc/default/tftpd-hpa

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/var/lib/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```

If you made any changes to the settings of the TFTP server, you need to restart it for them to take effect.

```
sudo restart tftpd-hpa
```

Lastly, if you would like to grant every user on the system permission to place files in the TFTP directory, use the following command, replacing <TFTP_DIRECTORY> with your chosen location.

```
sudo chmod ugo+rwx <TFTP_DIRECTORY>
```

2.2 Packages

Android development requires certain packages to be installed. Run the following commands to ensure that you have the packages installed:

```
sudo apt-get install git-core flex bison gperf libesd0-dev zip gawk ant libwxgtk2.6-dev
sudo apt-get install zlib1g-dev build-essential tofrodos x-dev libx11-dev libncurses5-dev
sudo apt-get install lib32readline5-dev libstdc++6 lib32z1 lib32z1-dev ia32-libs g++-multilib
sudo apt-get install libx11-dev libncurses5-dev uboot-mkimage libxml2-utils xsltproc
```

If you wish to store images in the target's NAND, also:

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

Install Java SE 6 JDK from Oracle:

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

Add the bin directory of the Oracle JDK package to your PATH.

Android does not build using openjdk. If you have openjdk installed on your development machine, you will need to ensure that the PATH variable includes the path to the Oracle JDK bin directory first. That is, "*type java*" should indicate the Oracle binary.

Also install the repo tool:

```
mkdir ~/bin -p
sudo apt-get install curl
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
export PATH=~/bin:$PATH
```

The Kernel and Driver sources are built using Sourcery G++ Lite 2010q1-202 for ARM GNU/Linux version. This tool chain can be obtained [here].

Add <toolchain_parent_dir>/arm-2010q1/bin to your PATH environment variable.

2.3 Board Setup - phyCORE-OMAP44xx

After the device is out of the box and setup, make sure that there is a DB9/DB25 cable attached between the host machine and UART3 (top DB9 connector) on the OMAP44xx carrier board to enable RS-232 serial communication. In a terminal on the host, access minicom and set it up, allowing console access over the serial port.

```
/* If minicom is not installed */
sudo apt-get install minicom

minicom -c on -s
```

Navigate to "Serial port setup" in minicom and modify line "A - Serial Device : " to read /dev/ttyS0.

Note: The serial device is dependent on what COM port you are connected to on your system, so /dev/ttyS0 is merely an example.

Next, modify "E - Bps/Par/Bits : " to have a speed of 115200 and 8-N-1 (8N1) for the stop bits. Return to the main menu of minicom and select "Save setup as dfl" to make this the default setup anytime minicom is loaded, meaning minicom -c on is all that needs to be done in the future for this machine to be able to communicate with the kit.

3 Building a BSP

3.1 Downloading Source

Create a directory to hold the source:

```
mkdir mydroid
cd mydroid
export MYDROID=`pwd`
tar xjf <path to>/phyCORE-OMAP44xx-JellyBean-2.0.bz2
```

3.1.1 Android Filesystem Sources

You can get the Android source for this release with:

```
cd $MYDROID
repo init -u git://git.omapzoom.org/platform/omapmanifest.git -b 27.x -m RLS4AJ.2.3_JellyBean.xml
```

Update the manifest by copying the file

ftp://ftp.phytec.de/pub/Products/phyCORE-OMAP44xx/Android/JellyBean/v2.0/RLS4AJ.2.3_JellyBean.xml

into the directory **.**repo/manifests, overwriting the previous version of that file.

```
repo sync
```

Apply the Android patches:

```
and_patches/patchandroid $MYDROID/and_patches
```

3.1.2 Kernel & Driver Sources

To clone kernel source from scratch:

```
cd $MYDROID
git clone git://git.omapzoom.org/kernel/omap.git kernel
cd kernel
git checkout f46bf5dee480b615454d7435ea4b5202d562eedb
```

Apply the kernel patches:

```
../kern_patches/patchkernel $MYDROID/kern_patches
```

3.1.3 Barebox Source

Source for barebox is included in the release tarball in the directory barebox.

3.2 Building The Kernel

To build the kernel:

```
cd $MYDROID/kernel
make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm pcm049_defconfig
make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm uImage
```

The kernel image will be arch/arm/boot/uImage.

If you wish to enable non-default features such as WiFi, see the Build Options section below before making uImage.

The Android kernel uses modversions to track kernel module compatibility. As a result, any time you change and rebuild the kernel you will need to rebuild the SGX kernel module and WiFi modules (if used). Later sections describe how to build those modules.

3.3 Building Barebox

To build barebox:

```
cd $MYDROID/barebox/barebox_mlo-2012.10.0
make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm
cd $MYDROID/barebox/barebox-2012.10.0
make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm
```

The first make command will build the first stage bootloader, MLO. The second make will build the second stage bootloader, barebox.bin, as well as the default environment image, barebox_default_env. If you wish to make changes to the default environment, edit the file barebox/env/config and rerun the second make. Otherwise, environment changes may be made on each target.

3.4 Building The Android Filesystem

To build the Android Filesystem:

```
cd $MYDROID
source build/envsetup.sh
lunch full_pcm049-userdebug
make
```

You can use *make -jN* where N is the number of CPUs in your development system to speed up the build.

3.5 Building The SGX Kernel Module

To build the SGX kernel module:

```
cd $MYDROID
mkdir sgx
cd sgx
tar xzf ../device/ti/proprietary-open/omap4/sgx_src/eurasia_km.tgz
cd eurasia_km/eurasiacon/build/linux2/omap4430_android
export KERNELDIR=$MYDROID/kernel
make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- TARGET_PRODUCT="blaze_tablet" BUILD=release TARGET_SGX=540
PLATFORM_VERSION=4.0 clean
make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- TARGET_PRODUCT="blaze_tablet" BUILD=release TARGET_SGX=540
PLATFORM_VERSION=4.0
cd $MYDROID
```

The Android kernel uses modversions to track kernel module compatibility. As a result, any time you change and rebuild the kernel you will need to rebuild the SGX kernel module.

4 Target Installation

4.1 Preparing Android Binaries

This step will prepare a directory, called myfs, containing all necessary Android files.

```
cd $MYDROID
sudo ./mkmyfs
```

If you wish to boot from the target's NAND, run:

```
cd $MYDROID
mkfs.ubifs -r myfs -m 2048 -c 3888 -e 126976 -o root.ubifs
ubinize -s 2048 -O 2048 -p 131072 -m 2048 -o root-pcm049.ubi ini-file
```

This will produce the file root-pcm049.ubi.

4.2 Preparing The SD Card

An SD Card is used either as a boot device or to install files to the target for NAND boot.

Configure the SD Card with the necessary partitions using the following script, mkcard. This need only be done once for each SD card.

```
#!/bin/bash
# Usage: sudo mkcard /dev/sdN   where sdN is the device associated with your SD card
if [ ! "$1" = "/dev/sda" ] ; then
        unset LANG
        DRIVE=$1
        if [ -b "$DRIVE" ] ; then
                dd if=/dev/zero of=$DRIVE bs=1024 count=1024
                SIZE=`fdisk -l $DRIVE | grep Disk | awk '{print $5}'`
                echo DISK SIZE - $SIZE bytes
                CYLINDERS=`echo $SIZE/255/63/512 | bc`
                echo CYLINDERS - $CYLINDERS
                {
                echo ,9,0x0C,*
                echo ,,,-
                } | sfdisk -D -H 255 -S 63 -C $CYLINDERS $DRIVE
                mkfs.vfat -F 32 -n "boot" ${DRIVE}1
                mke2fs -j -L "rootfs" ${DRIVE}2
        fi
fi
```

Note the Usage line in the script.

If you are using an SD card which already had partitions, Ubuntu will automatically mount them when the card is inserted. You will need to unmount them before using the script (*su umount /dev/sdN1*, *su umount /dev/sdN2*, ...).

After using the script, the partitions will be mounted as /media/boot and /media/rootfs when the SD card is inserted into a Ubuntu system

Copy Files to the SD card:

```
cp $MYDROID/barebox/barebox_mlo-2012.10.0/MLO /media/boot
cp $MYDROID/barebox/barebox-2012.10.0/barebox.bin /media/boot
```

If you intend to run Android from NAND, the above two files are all that you need on the SD card. If you intend to run Android from the SD card, additional files are necessary:

```
cp $MYDROID/kernel/arch/arm/boot/uImage /media/boot/uImage-pcm049
sudo rm -r /media/rootfs/*
sudo cp -r $MYDROID/myfs/* /media/rootfs
```

4.3 Installing Images Into NAND

The following files should be placed in the directory used by the tftp server (eg /var/lib/tftpboot):

- barebox/barebox_mlo-2012.10.0/MLO
- barebox/barebox-2012.10.0/barebox.bin
- barebox/barebox-2012.10.0/barebox_default_env
- kernel/arch/arm/boot/uImage (renamed uImage-pcm049)
- root-pcm049.ubi

Configure boot-mode for SD: SW2 switches 2, 3, and 5 ON. The SD card should be inserted into X11. After the board is powered on, stop the autoboot. Install the images as outlined in the following table:

Description	Command
Update MLO into NAND via TFTP	`update -t xload -d nand -m tftp -f MLO`
Update barebox into NAND via TFTP	`update -t barebox -d nand –m tftp -f barebox.bin`
Update bareboxenv into NAND via TFTP	`update -t bareboxenv -d nand –m tftp -f barebox_default_env`
Update kernel into NAND via TFTP	`update -t kernel -d nand –m tftp -f uImage-pcm049`
Update rootfs into NAND via TFTP	`update -t rootfs -d nand –m tftp -f root-pcm049.ubi`

NOTE : *TFTP is the default mode, as set in /env/config in barebox, therefore, "-m tftp" can be omitted from update commands.*

Power off the board and configure boot-mode for NAND: all SW2 switches off

4.4 Using An SD Card For The Root Filesystem

Note:. The random access of the filesystem which is performed by Android causes significant delay when it is stored on an SD card. Significant sluggishness of the system results. This slowness will vary from card to card. **It is therefore strongly recommended that the root filesystem be stored in NAND as described above.**

Configure boot-mode for SD: SW2 switches 2, 3, and 5 ON. The SD card should be inserted into X11.

Stop the board from autobooting and:

edit env/config

Change the lines

```
kernel_loc=nand
rootfs_loc=nand
rootfs_type=ubifs
```

to

```
kernel_loc=mmc
rootfs_loc=mmc
rootfs_type=ext3
```

Save the file with <ctrl>d, then:

saveenv

Note that the power must be cycled on the target for the new environment to become active.

4.5 Keeping Your Kernel On The TFTP Server

For kernel development, it is handy to have the target load the kernel from the tftp server rather than replacing the kernel in NAND or on the SD card. To facilitate this, stop the board from autobooting and:

edit env/config

Change the line:

```
kernel_loc=nand
```

to

```
kernel_loc=tftp
```

Ensure that eth0.ipaddr and eth0.serverip and the other eth0 settings are appropriate. Save the file with <ctrl>d, then:

saveenv

The kernel image file, uImage-pcm049, will then be loaded from /var/lib/tftpboot on your server. The Android filesystem will continue to be loaded from NAND or the SD card.

Note that the power must be cycled on the target for the new environment to become active.

5 Build Options

5.1 Wifi

To build an image which includes WiFi support:

```
cd $MYDROID/kernel
make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm menuconfig

    Device Drivers --->
        Network device support --->
            [*] Wireless LAN --->
                [*] TI Wireless LAN support --->
                    <M> TI wl12xx support
                    -M- TI wlcore support
                    < >   TI wlcore SPI support
                    <M>   TI wlcore SDIO support
```

The kernel should then be built as described above.

To build the kernel modules used for WiFi:

```
cd ${MYDROID}/hardware/ti/wlan/mac80211/compat_wl12xx
export KERNEL_DIR=${MYDROID}/kernel
export KLIB=${KERNEL_DIR}
export KLIB_BUILD=${KERNEL_DIR}
make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
cd ${MYDROID}
```

The Android kernel uses modversions to track kernel module compatibility. As a result, any time you change and rebuild the kernel you will need to rebuild the WiFi kernel modules.

To have the kernel modules loaded when the system is started, edit the file devices/phytec/pcm049/init.pcm049.rc and remove the leading # from the line:

```
#import init.pcm049.wifi.rc
```

To use the WiFi board, you must first install a riser board, part number 1368.0. Remove the SoM from the carrier board, install the riser board in the place of the SoM and then install the SoM on top of the riser board.

The PCM-049 WiFi adapter is installed on X3 of the riser board. It is oriented with the small white antenna (U6) pointing towards the SD slots at the rear of the carrier board.

Note that due to pin sharing on the OMAP4, the second SD card slot (SDMMC5) is not available when WiFi is enabled.

Due to a hardware issue, an SD card must be installed X11 when WiFi is in use. This can be avoided in your custom carrier board design by the use of different GPIOs for WiFi. Please contact Phytec for further details.

5.2 Audio

The default audio output device is the headset. If you prefer to use a speaker connection, the default can be changed in the file *device/phytec/pcm049 /audio/audio_policy.conf*. In the global configuration section, change:

```
attached_output_devices AUDIO_DEVICE_OUT_WIRED_HEADPHONE
default_output_devices AUDIO_DEVICE_OUT_WIRED_HEADPHONE
```

to

```
attached_output_devices AUDIO_DEVICE_OUT_SPEAKER
default_output_devices AUDIO_DEVICE_OUT_SPEAKER
```

A usage note at the top of this file provides details.

It is not possible with the 1348 carrier board to auto detect headphone insertion to switch between output devices, but this may be included with custom carrier boards.

6 Usage

6.1 Ethernet Configuration

The Ethernet interface is not enabled by default. To activate it, start the Settings application and change Ethernet from OFF to ON. Select Ethernet configuration (touch the Ethernet line on the screen) followed by eth0. Select either DHCP or a Static IP address and Apply.

6.2 Display Type

By default, the board will be setup for a 7" display. If you have a 5" display, stop the bootloader from loading Linux and run:

```
edit env/config
```

Comment out (add a leading #) two lines in the environment

```
bootargs="$bootargs panel_generic_dpi.name=pm070wl4"
bootargs="$bootargs stmpe_ts.inverty=1"
```

Uncomment (remove the leading # from) the following line in the environment

```
#bootargs="$bootargs panel_generic_dpi.name=pd050vl1"
```

Type <ctrl>d to exit the editor and then save the changes with

```
saveenv
```

The new environment will be used after a power cycle.

6.3 First Boot

The first boot following a clean rebuild of the target will be slow, taking several minutes to complete. This is due to Android's building of caches in /data.

6.4 adb

Although you have a console with the phyCORE-OMAP44xx, this is quite often not the case for Android devices. Debugging is therefore normally performed using adb, the Android debug bridge.

Connect a USB cable between the OTG port on the phyCORE-OMAP44xx board and your build machine.

adb can be found in out/host/linux-x86/bin of your Android build tree.

Common adb use includes *adb shell* to login to the device and *adb logcat* to see logging performed by Android services and applications.

Note that the shell user at the console is not root and therefore does not have permission to do many things. adb root/adb shell resolves this problem.

6.5 Installing Apps

If you wish to install an application without using the Android market:

1. Download the .apk file to your build machine.
2. Connect a USB cable between the OTG port on the phyCORE-OMAP4430 board and your build machine.
3. Run *adb install .apk_file_name*

6.6 Resume

The device will turn off the screen when it has been idle for a period of time. To bring it back from idle, press the PWRON (S3) button near the upper right corner of the carrier board.

7 Media

The phyCORE-OMAP44xx contains a number of devices which can be used for data storage. The Android application vold uses the file system/etc/vold. fstab to control which devices are mounted (when inserted) and where they are mounted. The source for the target vold.fstab file is devices/phytec/pcm049 /vold.fstab.

For example, the entry:

```
dev_mount sdcard /storage/sdcard1 auto /devices/platform/omap_hsmmc.0/mmc_host/mmc0
```

mounts a partition on the first SD card (SDMMC1 or X11) as /storage/sdcard1 when it is inserted.

If you wish to have the second SD card (SDMMC5 or X10) mounted when it is inserted, add the following line to vold.fstab:

```
dev_mount sdcard /storage/sdcard0 auto /devices/platform/omap_hsmmc.4/mmc_host/mmc1
```

The entry:

```
dev_mount usbdisk1 /storage/udisk auto /devices/platform/usbhs_omap/ehci-omap.0/usb1/1-1/1-1.1
```

mounts a partition on a USB drive when installed in the USB Host port

For the above mount commands, the mount point must be created in the "on init" section of the init.pcm049.rc boot script:

```
mkdir /storage/sdcard1 0000 system system
mkdir /storage/udisk 0000 system system
```

Depending on the intended use of the media, the mount points may be adjusted accordingly.

A note of caution: By default, the init.pcm049.rc uses /data/media for the storage of media files such as music. That is, these files are stored on the root filesystem and no SD card is required. To accomplish this, the following lines are included in the init.pcm049.rc script:

```
service sdcard /system/bin/sdcard /data/media 1023 1023
    class main
```

These lines mount /dev/media as /storage/sdcard0. If instead you wish to use a real SD card for media storage, comment out the above two lines.