**Android 2.3.4 on BeagleBoard-xm with PowerMemo Porting Guide**

Chih-Chen Kao (Jim)/version 1.

Vincent/version 2.

**Host (PC) setup requirements**

The host development environment for Android is based on Ubuntu; both 32-bit and 64-bit version is suitable. However, the 64-bit version requires additional libraries and software packets due to the compatibility issue. The host installation would need few more Android specific dependencies; these can be installed dynamically over network using below commands.

$ sudo apt-get update

$ sudo apt-get install git-core gnupg openjdk-6-jdk flex bison gperf libsdl-dev libesd0-dev build-essential zip curl libncurses5-dev zlib1g-dev uboot-mkimage expect

For 64-bit version, you may also need:

$ sudo apt-get install lib32ncursesw5-dev gcc-multilib ia32-libs lib32z1-dev

**Source Locations**

Using gitorious.org/rowboat

A tool called “Repo” helps to fetch the Android sources from gitorious.org/rowboat. Repo is a tool that makes it easier to work with Git in the context of Android.

To install, initialize, and configure Repo, follow these steps:

Make sure you have a “bin/” directory in your home directory and that it is including in your path:

$ mkdir ~/bin

$ PATH=~/bin:$PATH

Download the Repo script and ensure it is executable:

$ curl <https://android.git.kernel.org/repo> > ~/bin/repo

$ chmod a+x ~/bin/repo

The following commands help developers to clone sources from Gitorious.org/rowboat repository

$ mkdir rowboat-android

$ cd rowboat-android

$ repo init -u git://gitorious.org/rowboat/manifest.git -m TI-Android-GingerBread-2.3.4-DevKit-2.1.xml

$ repo sync

repo init -u [git@gitorious.org](mailto:git@gitorious.org):~changyihsin/rowboat/changyihsins-manifest.git -m TI-Android-Powermemo-0.0.1.xml

**Tool Chain setup**

Setup the tool chain path to print to arm-eabi- tools in prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin

$ export PATH=rowboat-powermemo/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin:$PATH

**Compiler error**

If you found compiler errors,

**Case 1:**

host Executable: acp (out/host/linux-x86/obj/EXECUTABLES/acp\_intermediates/acp)  
host SharedLib: libneo\_cs (out/host/linux-x86/obj/lib/libneo\_cs.so)  
host C++: libutils <= frameworks/base/libs/utils/RefBase.cpp  
frameworks/base/libs/utils/RefBase.cpp: In member function ‘void android::RefBase::weakref\_type::trackMe(bool, bool)’:  
frameworks/base/libs/utils/RefBase.cpp:483:67: error: passing ‘const android::RefBase::weakref\_impl’ as ‘this’ argument of ‘void android::RefBase::weakref\_impl::trackMe(bool, bool)’ discards qualifiers [-fpermissive]  
make: \*\*\* [out/host/linux-x86/obj/STATIC\_LIBRARIES/libutils\_intermediates/RefBase.o] Error 1

**Solution:**

gedit frameworks/base/libs/utils/Android.mk

Chage below compiler flag

**LOCAL\_CFLAGS += -DLIBUTILS\_NATIVE=1 $(TOOL\_CFLAGS)**

to

**LOCAL\_CFLAGS += -DLIBUTILS\_NATIVE=1 $(TOOL\_CFLAGS) -fpermissive**

**Case 2:**

In the gingerbread branch of AOSP, on Ubuntu 11.10 x86\_64 running Linux 3.0.0-12-generic, builds fail on obbtool (frameworks/base/tools/obbtool/Main.cpp) with:

error: "\_FORTIFY\_SOURCE" redefined [-Werror]

Interestingly, the same build worked on Ubuntu 11.04 x86\_64 -- the source tree was restored with rsync -av after a fresh install of Ubuntu 11.10 beta1 (which today has the latest OS packages). I am unable to see if the HEAD of the branch has fixed the issue as the origin git host is down.

**Solution:**

The fix is to apply a patch from Jim Huang (jserv) found on CyanogenMod's android\_build from Sep 6, 2011. Simply change build/core/combo/HOST\_linux-x86.mk line 61:

-HOST\_GLOBAL\_CFLAGS += -D\_FORTIFY\_SOURCE=0

+HOST\_GLOBAL\_CFLAGS += -U\_FORTIFY\_SOURCE -D\_FORTIFY\_SOURCE=0

**To build Linux kernel**

Change directory to kernel

$ cd rowboat-android/kernel

Do the following to build sources for BeagleBoard-xm

$ make ARCH=arm CROSS\_COMPILE=arm-eabi- distclean

$ make ARCH=arm CROSS\_COMPILE=arm-eabi- omap3\_beagle\_android\_defconfig

$ make ARCH=arm CROSS\_COMPILE=arm-eabi- uImage

**To build Android Filesystem**

$ make TARGET\_PRODUCT=beagleboard OMAPES=5.x

The above command builds Android Filesystem, SGX drivers, but NO wireless driver. We have to do several modifications in order to use Wi-Fi on BeagleBoard-xm/ Android 2.3.4, see below chapter for Wi-Fi porting.

Prepare the root filesystem

$ cd out/target/product/beagleboard

$ mkdir android\_rootfs

$ cp -r root/\* android\_rootfs

$ cp -r system android\_rootfs

$ sudo ../../../../build/tools/mktarball.sh ../../../host/linux-x86/bin/fs\_get\_stats android\_rootfs . rootfs rootfs.tar.bz2

**To build x-loader**

\* Change directory to x-loader

$ cd x-loader

Execute the following commands

Setup the tool-chain path to point to arm-eabi- tools in prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin

$ export PATH=/home/chiehyang/rowboat-android/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin:$PATH

$ make CROSS\_COMPILE=arm-eabi- distclean

$ make CROSS\_COMPILE=arm-eabi- omap3beagle\_config

$ 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/signGP directory of the Devkit.

$ ./signGP ./x-load.bin

The signGP tool will create a .ift file, rename the x-load.bin.ift to MLO

$ mv x-load.bin.ift MLO

**To build boot loader (u-boot)**

\* Change directory to u-boot-omap3

$ cd u-boot-omap3

\* Do the following to build u-boot for AM37x EVM

$ make CROSS\_COMPILE=arm-eabi- distclean

$ make CROSS\_COMPILE=arm-eabi- omap3\_beagle\_config

$ make CROSS\_COMPILE=arm-eabi-

This command will generate the u-boot Image "u-boot.bin"

**To generate SD/MMC card to boot Android**

These compiled Images can be copied to a SD / MMC card to boot Android on AM37x EVM The utility mk-bootscr from Tools package([Tools.tar.gz](http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/TI_Android_GingerBread_2_3_4_DevKit_2_1/exports/Tools.tar.gz) downloaded from <http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/TI_Android_GingerBread_2_3_4_DevKit_2_1/index_FDS.html>) can be used to generate a boot script(boot.scr) for the evm to boot automatically.(Provided the NAND environment is empty)

\* Generate boot.scr using mk-bootscr from tools folder

o Edit mk-bootscr with the desired 'bootcmd' and 'bootargs'. For a reference see http://processors.wiki.ti.com/index.php/TI-Android-GingerBread-2.3.4-DevKit-2.1\_UserGuide#Boot\_arguments

$ ./mkbootscr

\*Copy all the images to one folder

$cd ~/rowboat-android

$ mkdir bbxm

$ cp kernel/arch/arm/boot/uImage bbxm

$ cp u-boot/u-boot.bin bbxm

$ cp x-loader/MLO bbxm

$ cp Tools/mk-bootscr/boot.scr bbxm

$ cp out/target/product/beagleboard/rootfs.tar.bz2 bbxm

$ cp media\_clips bbxm

$ cp Tools/mk-mmc/mkmmc-android.sh bbxm

$ **./mkmmc-android <sd card mounted dev folder example:/dev/sdc> MLO u-boot.bin uImage boot.scr rootfs.tar.bz2 Media\_Clips**

NOTE: mkmmc-android.sh and mk-bootscr are provided in tools folder in DevKit

**Ubuntu 架設nfs server**

1. Install  
$ sudo apt-get install nfs-common  
$ sudo apt-get install nfs-kernel-server

2. Edit the /etc/exports file on the host Linux workstation (not the exports file on the target filesystem). This can be done by running:

host $ sudo vi /etc/exports

Add the following line for exporting the filesys area, substituting your user name for <useracct>. Use the full path from root(/); ~ or $HOME may not work for exports on all file systems.

/home/<useracct>/workdir/filesys \*(rw,no\_root\_squash,no\_all\_squash,sync)

Note: Make sure you do not add a space between the \* and the ( in the above command.

5) Use the following commands to make the NFS server aware of the change to its configuration and to invoke an NFS restart.

host $ sudo /usr/sbin/exportfs -av

host $ sudo /etc/init.d/nfs-kernel-server restart

Note: Use exportfs -rav to re-export all directories. Use sudo /etc/init.d/nfs-kernel-server status to verify that the NFS status is running.

6) Verify that the server firewall is turned off:

host $ sudo iptables -L

If the firewall is running, disable it:

host $ sudo iptables -F

enable the NFS support in using make menuconfig

CONFIG\_NETWORK\_FILESYSTEMS=y

CONFIG\_NFS\_FS=y

CONFIG\_NFS\_V3=y

CONFIG\_NFS\_V3\_ACL=y

CONFIG\_NFS\_V4=y

CONFIG\_NFS\_V4\_1=y

CONFIG\_PNFS\_FILE\_LAYOUT=y

CONFIG\_ROOT\_NFS=y

# CONFIG\_NFS\_USE\_LEGACY\_DNS is not set

CONFIG\_NFS\_USE\_KERNEL\_DNS=y

# CONFIG\_NFS\_USE\_NEW\_IDMAPPER is not set

# CONFIG\_NFSD is not set

CONFIG\_LOCKD=y

CONFIG\_LOCKD\_V4=y

CONFIG\_NFS\_ACL\_SUPPORT=y

CONFIG\_NFS\_COMMON=y

CONFIG\_SUNRPC=y

CONFIG\_SUNRPC\_GSS=y

CONFIG\_RPCSEC\_GSS\_KRB5=y

# CONFIG\_CEPH\_FS is not set

# CONFIG\_CIFS is not set

# CONFIG\_NCP\_FS is not set

# CONFIG\_CODA\_FS is not set

# CONFIG\_AFS\_FS is not set

**Create boot.scr using rowboat-android/Tools/mk-bootscr shell script.**

#!/bin/sh

cat <<EOF > boot.cmd

if fatload mmc 0 82000000 uImage

then

echo \*\*\*\*\* Kernel: /dev/mmcblk0p1/uImage \*\*\*\*\*

fi

echo \*\*\*\*\* RootFS: /dev/mmcblk0p2 \*\*\*\*\*

setenv bootargs 'console=ttyO2,115200n8 androidboot.console=ttyO2 mem=256M root=/dev/nfs rw ip=5.48.88.99 nfsroot=5.48.88.6:/home/vincent/rowboat-android/out/target/product/beagleboard/android\_rootfs,nolock noinitrd rootdelay=1 init=/init ip=5.48.88.99 mpurate=800 omap\_vout.vid1\_static\_vrfb\_alloc=y vram=8M omapfb.vram=0:8M'

**Android 啟動 init.rc with mmc**

SD 開機， 系统跑起来，可進入console . Android只出現文字 主畫面還沒起來  
 try in init.rc change it,...failed  
checked in console,get that"Unable to chmod /system/bin/alsa\_amixer: Read-only file system"  
init: cannot execve('/system/bin/dbus-daemon'): Permission denied  
  
init: cannot execve('/system/bin/keystore'): Permission denied  
init: cannot execve('/system/bin/mediaserver'): Permission denied  
init: cannot execve('/system/bin/keystore'): Permission denied  
  
改權限，在init.rc 加一行  
**chmod  0777   /system/bin/\***

**Wi-Fi Porting Issue**

Android uses a modified wpa\_supplicant (external/wpa\_supplicant) daemon for Wi-Fi support which is controlled through a socket by hardware/libhardware\_legacy/wifi/wifi.c (WiFiHW) that gets controlled from Android UI through android.net.wifi package from framworks/base/wifi/java/android/net/wifi/ and it’s corresponding JNI implementation in frameworks/base/core/jni/android\_net\_wifi\_Wifi.cpp Higher level network management is done in frameworks/base/core/java/android/net

The following figure shows the architecture of the Android wi-fi subsystem (use WL1271 chipset as example):

1. Enable Wireless driver in kerel

In this technical report, I use TP-LINK TL-WN321G ver. 3.6 for testing purpose, which is powered by Ralink wireless chip RT2571W.

The driver can be built as either modules or built-in kernel driver. For simplicity reason, I demonstrate how to apply the built-in kernel method here:

Firstly, the wireless support should be enabled in the kernel configuration. Make sure the following configuration is set:

CONFIG\_WIRELESS=y

CONFIG\_WEXT\_CORE=y

CONFIG\_WEXT\_PROC=y

CONFIG\_CFG80211=y

CONFIG\_MAC80211=y

Secondly, you should select the Wi-Fi driver related to the chip. By applying above settings, it will also reveal the Wi-Fi driver options supported by Linux kernel. The Ralink RT2571W chip is corresponded to the following configurations:

CONFIG\_WLAN=y

CONFIG\_RT2X00=y

CONFIG\_RT73USB=y

CONFIG\_RT2X00\_LIB\_USB=y

CONFIG\_RT2X00\_LIB=y

CONFIG\_RT2X00\_LIB\_FIRMWARE=y

CONFIG\_RT2X00\_LIB\_CRYPTO=y

If you use another wifi product, you can find out the wireless chip corresponded to your device through this website:

<http://www.wikidevi.com/wiki/Main_Page>

I personally recommend Atheros and ZyDAS chipset, since it is said to have better driver support in Linux kernel.

1. Enable building of wpa\_supplicant in your Boardconfig.mk

This is by simply adding: BOARD\_WPA\_SUPPLICANT\_DRIVER := WEXT to the BoardConfig.mk. It will set WPA\_BUILD\_SUPPLICANT to TRUE in exteral/wpa\_supplicant/Android.mk enabling building of driver\_wext.c

1. Enable Debug message for wpa\_supplicant (Optional)

The log message is set to MSG\_INFO by default. To enable more messages, do the following modification:

In common.c, set wpa\_debug\_level = MSG\_DEBUG

In common.h, modify wpa\_printf to the same as below:

1. Provide a proper wpa\_supplicant.conf

The file wpa\_supplicant.conf should be copied by your AndroidBord.mk to $(TARGET\_OUT\_ETC)/wifi (usually /system/etc/wifi/wpa\_supplicant.conf). This location will be used on wpa\_supplicant from init.rc

There are two different ways in which wpa\_supplicant can be configured; one is to use a private socket in Android namespace, created by socket\_local\_client\_connect() function in wpa\_ctrl.c and another is by using a standard UNIX socket.

For Android private socket:

ctrl\_interface=wlan0

update\_config=1

For UNIX standard socket:

ctrl\_interface=DIR=/data/system/wpa\_supplicant GROUP=wifi

update\_config=1

Depending on your driver you may want to add:

ap\_scan=1

If you are suffering from AP association problems, you should change ap\_scan=0 to let the driver do the association instead of wpa\_supplicant.

1. Have the correct permission and paths created by init.rc

Incorrect permissions will result in wpa\_supplicant not being abel to create or open the control socket, and the libhardware\_legacy/wifi/wifi.c won’t connect. Since the wap\_supplicant for Android system, which is modified by Google, is configured to run as “wifi” user/group, the directory and file ownership should belong to “wifi” user/group (see os\_program\_init() function in wpa\_supplicant/os\_unix.c for more detail).

The incorrect permission setting will cause the error such as:

E/WifiHW ( ): Unable to open connection to supplicant on "/data/system/wpa\_supplicant/wlan0": No such file or directory

Also, wpa\_supplicant.conf should belong to “wifi” user/group because wpa\_supplicant will modify this file. If your system has /system as read-only, use a location like /data/misc/wifi/wpa\_supplicant.conf and modify wpa\_supplicant service in init.rc with new location.

To do so, make sure the paths are correctly created in init.rc:

mkdir /system/etc/wifi 0770 wifi wifi

chmod 0777 /system/etc/wifi

chmod 0777 /system/etc/wifi/wpa\_supplicant.conf

chown wifi.wifi /system/etc/wifi/wpa\_supplicant.conf

#wpa\_supplicant control socket for android wifi.c (android private socket)

mkdir /data/misc/wifi 0770 wifi wifi

mkdir /data/misc/wifi/sockets 0770 wifi wifi

chmod 0777 /data/misc/wifi

chmod 0777 /data/misc/wifi/wpa\_supplicant.conf

chown wifi.wifi /data/misc/wifi

chown wifi.wifi /data/misc/wifi/wpa\_supplicant.conf

If you use a UNIX standard socket in wpa\_supplicant.conf (see above) add:

# wpa\_supplicant socket (unix socket mode)

mkdir /data/system/wpa\_supplicant 0771 wifi wifi

chmod 0771 /data/system/wpa\_supplicant

chown wifi wifi /data/system/wpa\_supplicant

Do not add these if you use Android private socket because it will make wpa\_supplicant non-functional. Since hardware/libhardware\_legacy/wifi/wifi.c checks for existence of the /data/system/wpa\_supplicant folder, and will pass a wrong interface name to wpa\_ctrl\_open() function.

1. Make sure the wpa\_supplicant and dhcpd start from init.rc

For Android private socket:

service /data/misc/wifi/wpa\_supplicant.conf  
socket wpa\_wlan0 dgram 660 wifi wifi  
group system wifi inet  
disabled  
oneshot

For UNIX standard socket:

service wpa\_supplicant /system/bin/wpa\_supplicant -dd -Dwext -iwlan0 -c /system/etc/wifi/wpa\_supplicant.conf

group system wifi inet

disabled

oneshot

If your wifi driver creates a wifi interface with other name than wlan0, you will have to modify the above line accordingly.

For dhcpcd setting:

service dhcpcd\_wlan0 /system/bin/dhcpcd –t 120 –d wlan0

group system dhcp

disabled

oneshot

1. Provided the driver either as modules or built-in kernel and proper kernel support for it

First make sure that CONFIG\_PACKET and CONFIG\_NET\_RADIO (wireless extensions) are enabled in your kernel. The driver can be built as module (default android way) or built in kernel (if you want to rely in kernel auto probing to support multiple driver eg. USB wifi) but will require source code modifications (see below).

Build as kernel module:

Define in your BoardConfig.mk:

* 1. WIFI\_DRIVER\_MODULE\_PATH := path to the module to be loaded. You also need to specify module name in that path too. For example, “/system/lib/modules/wlan.ko”.
  2. WIFI\_DRIVER\_MODULE\_NAME:= the name of the network interface that the driver creates, for example “wlan0”.
  3. WIFI\_DRIVER\_MODULE\_ARG:= any arguments that you want to pass to the driver on insmod, for example “nohwcrypt”.

Make sure you copy your kernel module to the correct location while building Android system.

Build as build-in kernel driver:

Modify init.rc to inform “hardware/libhardware\_legacy/wifi/wifi.c” about the name of the interface, that the driver is already loaded and set the status of wpa\_supplicant to running:

setprop wifi.interface "wlan0"

setprop wlan.driver.status "ok"

Do NOT add setprop init.svc.wpa\_supplicant "running”, since it will prevent wpa\_supplicant from starting from init.

“hardware/libhardware\_legacy/wifi/wifi.c” need to be modified so the functions insmod() and rmmod() return 0 (simply add return 0; as the first line in functions since they are not needed when driver is built in kernel) and return before checking for /proc/modules in check\_driver\_loaded() function.

1. Provide a firmware for the driver

If your driver needs a firmware you will have to copy this firmware file to “/system/etc/firmware” on the Android build. The init process takes response to the firmware events and loads the firmware file from the directory. (see “system/core/init/devices.c handle\_firmware\_event() function” for more detail). Firmware file name is defined by the driver and might also contain a folder like: RTL8192SU/rtl8192sfw.bin, entire file path should be available in “/system/etc/firmware”.

1. Make your driver work with Android custom wpa\_supplicant commands and SIOCSIWPRIV ioctl.

Android uses SIOCSIWPRIV ioctl to send commands to modify driver behavior and receive information like signal strength, MAC address of the AP, link speed etc. This ioctl is usually not implemented in any known wireless drivers except bcm4329.

The errors from not having this ioctl implemented will look like:

E/wpa\_supplicant( ): wpa\_driver\_priv\_driver\_cmd failed wpa\_driver\_priv\_driver\_cmd RSSI len = 4096

E/wpa\_supplicant( ): wpa\_driver\_priv\_driver\_cmd failed

D/wpa\_supplicant( ): wpa\_driver\_priv\_driver\_cmd LINKSPEED len = 4096

E/wpa\_supplicant( ): wpa\_driver\_priv\_driver\_cmd failed

I/wpa\_supplicant( ): CTRL-EVENT-DRIVER-STATE HANGED

After 4 WEXT\_NUMBER\_SEQUENTIAL\_ERRORS errors, Android will abort using the device.

To quickly test your wifi from interface you can disable error checking in “external/wpa\_supplicant/driver\_wext.c” by simply making ret = 0; in wpa\_driver\_priv\_driver\_cmd() function after the SIOCSIWPRIV ioctl call. This will make all access points in android UI appear without signal or MAC address.

To proper implement the ioctl you will need to modify your kernel driver to reply to SIOCSIWPRIV ioctl with RSSI (signal strength) and MACADDR commands being the most important.

A better way is to add a custom driver\_xxx.c to Google external/wpa\_supplicant/ implementing wpa\_driver\_priv\_driver\_cmd() function that will take care of RSSI, MACADDR and others, through calls to SIOCGIWSTATS, SIOCGIFHWADDR ioctls, with the rest of the functions being called from driver\_wext.c.

**Disable Module Version Check**

Edit [android]/kernel/arch/arm/configs/omap3\_beagle\_android\_defconfig

Disable CONFIG\_MODVERSIONS and CONFIG\_MODULE\_SRCVERSION\_ALL

**Create A simple Application in the root filesystem**

1. Create a hello directory in <android source dir>/external folder.

2. Put you application in this directory and  write Android.mk

 Example Android.mk

**LOCAL\_PATH:= $(call my-dir)  
  
include $(CLEAR\_VARS)  
  
LOCAL\_MODULE\_TAGS := eng  
LOCAL\_C\_INCLUDES += $(LOCAL\_PATH) $(LOCAL\_PATH)/$(KERNEL\_DIR)/include  
LOCAL\_SRC\_FILES := hello.c  
LOCAL\_MODULE := hello  
include $(BUILD\_EXECUTABLE)**

3. use following command to configure environments.

    $ source <android source dir>/build/envsetup.h

4. compile your application.

    $ cd <android source dir>/external/hello

    $ mm TARGET\_PRODUCT=beagleboard.

**Enable debugfs interface**

Enable debug file system support in kernel config

CONFIG\_DEBUG\_FS=y

#create a temporary folder for debug

mkdir /debug

#mount it

mount -t debugfs debug /debug

**Modify the CPU Frequency**

**#**

/sys/devices/system/cpu/cpu0/cpufreq/cpuinfo\_min\_freq

**#**

/sys/devices/system/cpu/cpu0/cpufreq/cpuinfo\_max\_freq

**#**

/sys/devices/system/cpu/cpu0/cpufreq/cpuinfo\_transition\_latency

**#**

/sys/devices/system/cpu/cpu0/cpufreq/scaling\_min\_freq

**#**

/sys/devices/system/cpu/cpu0/cpufreq/scaling\_max\_freq

**#**

/sys/devices/system/cpu/cpu0/cpufreq/affected\_cpus

**#**

/sys/devices/system/cpu/cpu0/cpufreq/related\_cpus

**#**

/sys/devices/system/cpu/cpu0/cpufreq/scaling\_governor

**#**

/sys/devices/system/cpu/cpu0/cpufreq/scaling\_driver

**#**

/sys/devices/system/cpu/cpu0/cpufreq/scaling\_available\_governors

**# set cpu frequency**

/sys/devices/system/cpu/cpu0/cpufreq/scaling\_setspeed

echo 300000 > scaling\_setspeed

**# show all the available frequency**

/sys/devices/system/cpu/cpu0/cpufreq/scaling\_available\_frequencies

# cat scaling\_available\_frequencies

300000 600000 800000 1000000

**# current cpu frequency**

/sys/devices/system/cpu/cpu0/cpufreq/cpuinfo\_cur\_freq

**#**

/sys/devices/system/cpu/cpu0/cpufreq/scaling\_cur\_freq

**#**

/sys/devices/system/cpu/cpu0/cpufreq/stats

**# Modify the Led drivers**

/kernel/driver/leds

gpio\_set\_value(gpio number, high or low);

**# turn on user 0 led**

echo 1 > /sys/class/leds/beagleboard::usr0/brightness

**# turn off user 0 led**

echo 0 > /sys/class/leds/beagleboard::usr0/brightness

**# turn on user 1 led**

echo 1 > /sys/class/leds/beagleboard::usr1/brightness

**# turn off user 1 led**

echo 0 > /sys/class/leds/beagleboard::usr1/brightness

**# turn on pmu stat led**

echo 1 > /sys/class/leds/beagleboard::pmu\_stat/brightness

**# turn off pmu stat led**

echo 0 > /sys/class/leds/beagleboard::pmu\_stat/brightness

**#Create file with 1M size and 10M size**

dd if=/dev/zero of=128k bs=1024 count=128

dd if=/dev/zero of=256k bs=1024 count=256

dd if=/dev/zero of=512k bs=1024 count=512

dd if=/dev/zero of=1M bs=1024 count=1024

dd if=/dev/zero of=2M bs=1024 count=2048

dd if=/dev/zero of=4M bs=1024 count=4096

dd if=/dev/zero of=8M bs=1024 count=8192

**Files related to kernel patch for uprobe profiling module. (kernelprofiling.patch)**

**Modified arch\arm\configs\omap3\_beagle\_android\_defconfig to**

1. support WiFi driver

CONFIG\_RT2500USB=y

CONFIG\_RT73USB=y

# CONFIG\_RT2800USB is not set

CONFIG\_RT2X00\_LIB\_USB=y

CONFIG\_RT2X00\_LIB=y

CONFIG\_RT2X00\_LIB\_FIRMWARE=y

CONFIG\_RT2X00\_LIB\_CRYPTO=y

CONFIG\_RT2X00\_LIB\_LEDS=y

# CONFIG\_RT2X00\_DEBUG is not set

# CONFIG\_WL1251 is not set

# CONFIG\_WL12XX is not set

CONFIG\_WL12XX\_PLATFORM\_DATA=y

# CONFIG\_ZD1211RW is not set

1. Enable CPU\_FREQ feature, just for test purpose.

CONFIG\_CPU\_FREQ\_GOV\_PERFORMANCE=y

CONFIG\_CPU\_FREQ\_GOV\_POWERSAVE=y

CONFIG\_CPU\_FREQ\_GOV\_USERSPACE=y

CONFIG\_CPU\_FREQ\_GOV\_ONDEMAND=y

CONFIG\_CPU\_FREQ\_GOV\_INTERACTIVE=y

CONFIG\_CPU\_FREQ\_GOV\_CONSERVATIVE=y

1. Enable LED system class file system, to turn on/off user LEDs. The commands can be found in previous page in this document.

CONFIG\_NEW\_LEDS=y

CONFIG\_LEDS\_CLASS=y

#

# LED drivers

#

# CONFIG\_LEDS\_PCA9532 is not set

CONFIG\_LEDS\_GPIO=y

CONFIG\_LEDS\_GPIO\_PLATFORM=y

# CONFIG\_LEDS\_LP3944 is not set

# CONFIG\_LEDS\_LP5521 is not set

# CONFIG\_LEDS\_LP5523 is not set

# CONFIG\_LEDS\_PCA955X is not set

# CONFIG\_LEDS\_DAC124S085 is not set

# CONFIG\_LEDS\_REGULATOR is not set

# CONFIG\_LEDS\_BD2802 is not set

# CONFIG\_LEDS\_LT3593 is not set

# CONFIG\_LEDS\_TRIGGERS is not set

#

# LED Triggers

#

**kernel\arch\arm\include\asm\kprobes.h => add UPROBE\_PATCH support**

**kernel\arch\arm\kernel\kprobes-decode.c => I have patched this file from** [**www.kernel.org**](http://www.kernel.org/)**.**

**kernel\arch\arm\kernel\kprobes.c => add UPROBE\_PATCH support**

**kernel\arch\arm\kernel\Makefile => add file uprobes.c**

**kernel\arch\arm\kernel\uprobes.c => new file for supporting userspace application probes**

**kernel\drivers\leds\leds-gpio.c => turn on/off led and trigger DAQ functions**

**kernel\fs\namei.c => add UPROBE\_PATCH support**

**kernel\include\linux\kprobes.h => add UPROBE\_PATCH support**

**kernel\include\linux\namei.h => add UPROBE\_PATCH support**

**kernel\include\linux\powermemo.h => new file patch for powermemo support**

**kernel\include\linux\sched.h => add UPROBE\_PATCH support**

**kernel\include\net\sock.h => patch for powermemo support**

**kernel\kernel\bounds.s => add UPROBE\_PATCH support**

**kernel\kernel\fork.c => add UPROBE\_PATCH support**

**kernel\kernel\kprobes.c => add UPROBE\_PATCH support**

**kernel\kernel\Makefile => add UPROBE\_PATCH support**

**kernel\kernel\ptrace.c => add UPROBE\_PATCH support**

**kernel\kernel\sched.c => patch for powermemo support**

**kernel\kernel\uprobes.c => new file for supporting userspace application probes**

**kernel\kernel\user\_bkpt\_xol.c => new file for supporting userspace application probes**

**kernel\mm\memory.c => add UPROBE\_PATCH support**

**kernel\net\core\sock.c => patch for powermemo support**

**kernel\net\ipv4\af\_inet.c => patch for powermemo support**

**kernel\net\ipv4\tcp\_ipv4.c => patch for powermemo support**

**kernel\net\ipv4\udp.c => patch for powermemo support**

**kernel\net\mac80211\rx.c => patch for powermemo support**

**kernel\net\mac80211\tx.c => patch for powermemo support**

**kernel\net\sched\sch\_generic.c => patch for powermemo support**

**kernel\Makefile => add UPROBE\_PATCH support**

How to create patch file ?

**diff -crB Tb01 Tb02 > Tb02.patch** => create patch file

**patch --dry-run -p1 -i Tb02.patch** => try to test this patch file

**patch -p1 -i Tb02.patch** => after dry-run gets good results, do this and enjoy the compilation