

# Modifying and Rebuilding Android Kernel and Root Filesystem

# **Document History**

Date	Version	Change Description
02/27/2012	1.0	Initial version
04/16/2012	1.1	Integrating review changes
12/06/2012	1.2	Updated to support AADK 1.2

# **Table of Contents**

1	Document History		
2	·		
3		oduction	
	3.1	Problem Solved	4
	3.2	Audience	4
	3.3	Assumptions	4
	3.4	Scope	4
	3.5	Theory of Operation	4
4	Basics		4
	4.1	Preparing the system and installing tools	4
	4.2	Configuring the environment	5
	4.3	Re-Configuring Android Kernel	7
	4.4	Modifying Root filesystem	9
	4.5	Re-building Android Kernel and rootfs	10
	4.6	Checking generated Android Kernel and rootfs images	11
	4.7	Deploying new images on to the target	
	4.7.1	1 1	
	4.7.2	NFS BOOTING (NFS server required)	12
	4.8	Debugging new system	
5	Con	clusion	13
6	App	ppendix	
	6.1	Glossary of terms	13

#### 3 Introduction

 Customers who need to add additional device support, drivers and features to Android will have to rebuild Android Kernel and rootfs.

#### 3.1 Problem Solved

This document will show how to customize and rebuild Android kernel and root filesystem images

#### 3.2 Audience

Customers using Digi embedded Android SDK on ccwmx51 and ccwmx53 based modules, who need to add additional device drivers our other functionality and/or customize root filesystem to contain their own applications and configurations.

### 3.3 Assumptions

This document was created in reference to Android SDK v1.2.0

### 3.4 Scope

- Extract and install sources and toolchain
- Configure and customize Android kernel and root filesystem
- Rebuild
- Load and run new images on target

#### 4 Basics

The copy of the source code and toolchain required to rebuild Android kernel and root filesystem images is located on the Android SDK DVD in the tar file. Following the procedure below will require having an installation of Linux distribution on a host machine. Rebuilding Android Kernel and rootfs under Windows is currently not possible. As we only supply Linux based toolchain. Kubundu 10.x installed from Digi Embedded Linux Live DVD was used as an example to demonstrate the process below.

# 4.1 Preparing the system and installing tools

The copy of the source code and toolchain required to rebuild Android kernel and rootfs images is located on the Android SDK DVD in the tar file. Extract the code from a tarball included in the DVD (/src folder). Once uncompressed, the whole project occupies nearly 3.5 GBs.

cd \${HOME} tar xvfj /<path to tarfile on DVD>/aadk-<version>.tar.bz2

Here is an example showing my steps on a host machine. I am using a Kubuntu Linux that comes with Digi Embedded Linux kit. Digi Android SDK DVD v1.2 is mounted under /media/AADK:

leonidm@ubuntu:~\$ ls -la /media/AADK/sources/aadk-1.2.0.tar.bz2 -r-xr-xr-x 1 leonidm 4294967295 1023820093 2012-02-15 15:10 /media/AADK/sources/aadk-1.2.0.tar.bz2

I will switch to my home directory(or any other folder you have write permission to):

leonidm@ubuntu:~\$ cd leonidm@ubuntu:~\$ pwd /home/leonidm

Make sure I have at least 4Gb of free space (I have almost 8GB):

leonidm@ubuntu:~/aadk-1.2.0\$ df -k.

Used Available Use% Mounted on Filesystem 1K-blocks

/dev/sda1 29640780 20148416 <mark>7986676</mark> 72%/

And will start uncompressing the tar file:

leonidm@ubuntu:~\$ tar xvfj /media/AADK/sources/aadk-1.2.0.tar.bz2

tar: Record size = 8 blocks

aadk-1.2.0/

aadk-1.2.0/external/

aadk-1.2.0/external/neven/

# 4.2 Configuring the environment

To prepare the environment:

cd aadk-<version>

. build/envsetup.sh

lunch

Choose the desired platform from the lunch menu:

5. imx51\_ccwmx51js-eng

6. imx53\_ccwmx53js-eng

In my example I will install some missing tools needed for build:

leonidm@ubuntu:~\$ sudo apt-get install bison flex gperf

To prepare the environment:

leonidm@ubuntu:~/aadk-1.2.0\$ pwd

/home/leonidm/aadk-1.2.0

```
leonidm@ubuntu:~/aadk-1.2.0$. build/envsetup.sh
including device/fsl/imx5x/vendorsetup.sh
leonidm@ubuntu:~/aadk-1.2.0$ lunch
```

You're building on Linux

Lunch menu... pick a combo:

- 1. generic-eng
- 2. simulator
- 3. imx51\_bbg-eng
- 4. imx51\_bbg-user
- 5. imx51\_ccwmx51js-eng
- 6. imx53 ccwmx53js-eng
- 7. imx53\_evk-eng
- 8. imx53\_evk-user
- 9. imx53\_ard-eng
- 10. imx53 ard-user
- 11. imx53\_smd-eng
- 12. imx53\_smd-user
- 13. imx50\_rdp-eng
- 14. imx50\_rdp-user

[I will do this for ccwmx53 so I will choose 6, choose 5 if you use ccwmx51]

Which would you like? [generic-eng] 6

\_\_\_\_\_\_

PLATFORM VERSION CODENAME=REL

PLATFORM\_VERSION=2.3.4

TARGET PRODUCT=imx53 ccwmx53js

TARGET\_BUILD\_VARIANT=eng

TARGET SIMULATOR=false

TARGET\_BUILD\_TYPE=release

TARGET\_BUILD\_APPS=

TARGET\_ARCH=arm

HOST ARCH=x86

HOST\_OS=linux

*HOST\_BUILD\_TYPE=release* 

BUILD\_ID=GRJ22

\_\_\_\_\_\_

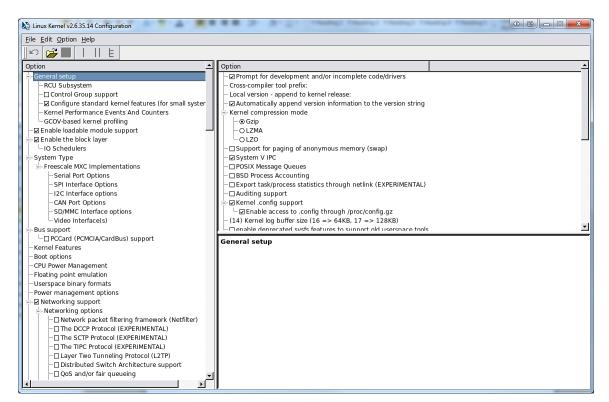
# 4.3 Re-Configuring Android Kernel

To make changes to android kernel cd to kernel imx folder and run following command:

#### make ARCH=arm xconfig

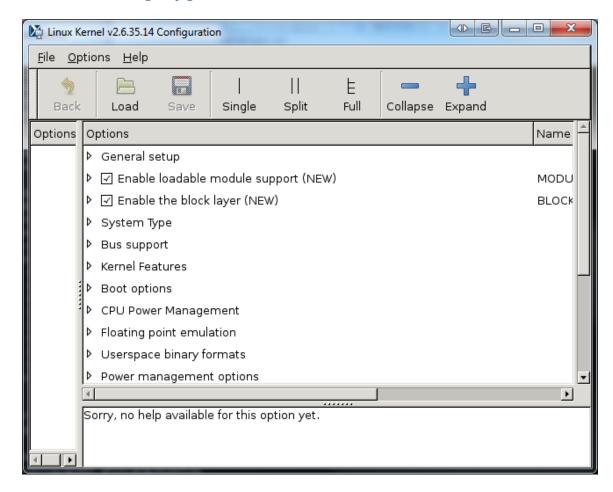
```
in my example:
leonidm@ubuntu:~/aadk-1.2.0$ cd kernel_imx/
leonidm@ubuntu:~/aadk-1.2.0/kernel imx$ pwd
/home/leonidm/aadk-1.2.0/kernel_imx
leonidm@ubuntu:~/aadk-1.2.0/kernel_imx$ make ARCH=arm xconfig
scripts/kconfig/qconf arch/arm/Kconfig
```

#### Graphical Kernel configuration screen will come up:

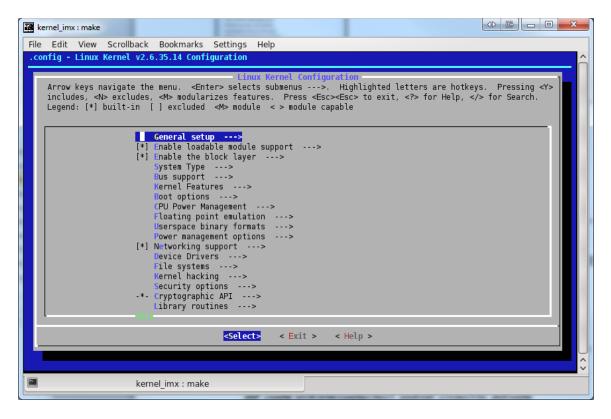


Alternatively depending on your Linux distribution and available packages you can use GTK based GUI (might require sudo apt-get install libgtk2.0-dev libglib2.0-dev libglade2-dev)

#### make ARCH=arm gconfig



#### Or text menu based Kernel configuration tool make ARCH=arm menuconfig



This is standard Linux Kernel configuration screen you should be familiar with. Make the changes you need to the Kernel configuration, save and exit the tool. This step will update .config files with the changes you've just made in Kernel configuration tool.

To see the differences between new .config file created and the default one you can use following command issued from the kernel\_imx folder:

diff.config arch/arm/configs/imx5 android ccwmx53js defconfig

To discard your changes and go back to defaults manually delete .config file.

### 4.4 Modifying root filesystem

Whenever a file is modified, the whole root filesystem must be built again to generate new images to boot from FLASH or from NFS. The build process after modifying the root filesystem is much quicker than the first time.

### 4.5 Re-building Android Kernel and rootfs

Now the build process can be launched: Go back to the top level of ./ aadk-1.2.0/
First make clean to prepare the build:

make -j<number\_of\_CPUs> clean

Now you can start building the entire tree:

make -j<number\_of\_CPUs>

#### or to rebuild just the kernel:

make -j<number\_of\_CPUs> del\_linux\_image

The resulting images can be found at: out/target/product/<platform>

To clean the results of the last build: make -j<number\_of\_CPUs> clean

Depending on the number of CPU's, building process can take from 15 minutes to several hours.

#### In my example:

\_\_\_\_\_

[UBIFS]: out/target/product/imx53\_ccwmx53js/android-ccwmx53js-128.ubifs [UBI]: out/target/product/imx53\_ccwmx53js/android-ccwmx53js-128.ubi [UBIFS]: out/target/product/imx53\_ccwmx53js/android-ccwmx53js-512.ubifs [UBI]: out/target/product/imx53\_ccwmx53js/android-ccwmx53js-512.ubi [NFS]: out/target/product/imx53\_ccwmx53js/android-ccwmx53js.tar.bz2

Build UBIFS/UBI/NFS images for Android rootfs

### 4.6 Checking generated Android Kernel and rootfs images

All the files generated after building the source code. Below is brief description of those files:

out/target/product/<platform> contains:

- · files to be flashed into the device:
  - uImage-android-<platform>: kernel image
  - android-<platform>-<size>.ubifs: root filesystem
- · files necessary to boot the device from NFS:
  - uImage-android-<platform>: kernel image
  - root folder
  - data folder: copy content to root/data
  - system folder: copy content to root/system

#### So in our example all images go to

```
aadk-1.2.0/out/target/product/imx53_ccwmx53js/:
```

leonidm@ubuntu:~/aadk-1.2.0\$ ls -lh out/target/product/imx53\_ccwmx53js/ |more total 488M

```
-rw-r--r-- 1 leonidm leonidm 78M 2012-02-27 15:49 android-ccwmx53js-128.ubi
```

-rw-r--r-- 1 leonidm leonidm 77M 2012-02-27 15:49 android-ccwmx53js-128.ubifs

-rw-r--r-- 1 leonidm leonidm 82M 2012-02-27 15:49 android-ccwmx53js-512.ubi

-rw-r--r-- 1 leonidm leonidm 81M 2012-02-27 15:49 android-ccwmx53js-512.ubifs

-rw-r--r-- 1 leonidm leonidm 62M 2012-02-27 15:51 android-ccwmx53js.tar.bz2

-rw-r--r-- 1 leonidm leonidm 16 2012-02-27 14:58 android-info.txt

-rw-r--r-- 1 leonidm leonidm 3.4K 2012-02-27 14:33 clean steps.mk

drwxr-xr-x 3 leonidm leonidm 4.0K 2012-02-27 15:03 data

-rw-r--r- 1 leonidm leonidm 35K 2012-02-27 15:48 installed-files.txt

drwxr-xr-x 14 leonidm leonidm 4.0K 2012-02-27 15:48 obj

-rw-r--r-- 1 leonidm leonidm 297 2012-02-27 14:33 previous\_build\_config.mk

-rw-r--r- 1 leonidm leonidm 1.9M 2012-02-27 15:23 ramdisk.img

drwxr-xr-x 3 leonidm leonidm 4.0K 2012-02-27 15:23 recovery

drwxr-xr-x 8 leonidm leonidm 4.0K 2012-02-27 15:23 root

drwxr-xr-x 6 leonidm leonidm 4.0K 2012-02-27 15:23 symbols

drwxr-xr-x 11 leonidm leonidm 4.0K 2012-02-27 15:23 system

-rw----- 1 leonidm leonidm 103M 2012-02-27 15:48 system.img

-rw-r--r-- 1 leonidm leonidm 2.6M 2012-02-27 14:37 uImage-android-ccwmx53js

-rw----- 1 leonidm leonidm 1.5M 2012-02-27 15:14 userdata.img

#### 4.7 Deploying new images on to the target

#### 4.7.1 FLASH BOOTING (TFTP server required)

Instructions to flash the images into the device:

- · Copy both images, kernel and root filesystem, in the TFTP server (i.e. /tftpboot).
- · Boot the device and press any key to stop boot at u-boot.
- · Check the flash sector size to know which root filesystem image has to be flashed (FLASH erase block size will be reflected in rootfs image file name containing -128 or -512):

nand info

· Flash the kernel:

update android tftp uImage-android-<platform> or just

update android tftp

· Flash the root filesystem:

```
update androidfs tftp android-<platform>-<size>.ubifs
or just
```

update androidfs tftp

In this case system will determine — size automatically so this should be a preferred method.

- · Set *bootcmd* to boot from FLASH: setenv bootcmd dboot android flash saveenv
- · Restart the device and let it complete boot process: reboot

### 4.7.2 NFS BOOTING (NFS server required)

Assuming NFS server path is /exports/<folder>, instructions are:

· Copy built images to NFS server:

```
sudo cp -r out/target/product/<platform>/uImage-android-<platform>
/exports/<folder>
```

sudo cp -r out/target/product/<platform>/root/\* /exports/<folder> sudo cp -r out/target/product/<platform>/system/\* /exports/<folder>/system sudo cp -r out/target/product/<platform>/data/\* /exports/<folder>/data

- · Boot the device and press any key to stop boot at u-boot.
- · Set bootcmd to boot from NFS (to boot from flash, remember to set it back to "dboot android flash") and set *npath* with the NFS server path:

setenv bootcmd dboot android setenv npath /exports/<folder>

· Set the proper kernel image name: setenv aimg uImage-android-ccwmx53js To boot from NFS the uImage file can be copied in the NFS server or in the TFTP server. Depending on this, the bootcmd variable will be different.

```
- uImage in TFTP server --->
setenv bootcmd dboot android tftp
     - uImage in NFS server --->
setenv bootcmd dboot android nfs.
```

During boot, user may be asked to calibrate the touch screen if that's the selected video interface.

# 4.8 Debugging new system

The main commands to retrieve debug messages in the device's console are:

- · **dmesg**: diagnostic messages created by the kernel.
- · logcat: logging messages provided by Android.

In case of error, these two commands allow users to determine, at least in a first approach, what's going on.

#### Conclusion 5

Give the reader a synopsis of what he just read.

# **Appendix**

# 6.1 Glossary of terms

rootfs - Root File System