

MV5410 Android 4.2 Compiling



Microvision Co., Ltd.

1. Package for Development

The following packages are in the directory /SRC/Android in the CD:

File	Description	Version
u-boot-samsung-5410.tar.gz	Bootloader	
kernel-3.4.5-5410.tar.gz	Kernel	3.4.5
jellybean.tar.gz	Jelly Bean	4.2
arm-2009q3-67-arm-none-linux-gnueabi.bin	q3-compiler	

Tool chain

This Android4.2 BSP compiles Bootloader and the Kernel uses Q3-Compiler for compilation.

You have to install linux 64Bit for compiling Android4.3 BSP.

2. Bootloader Setup

2.1. u-boot Environment Setup

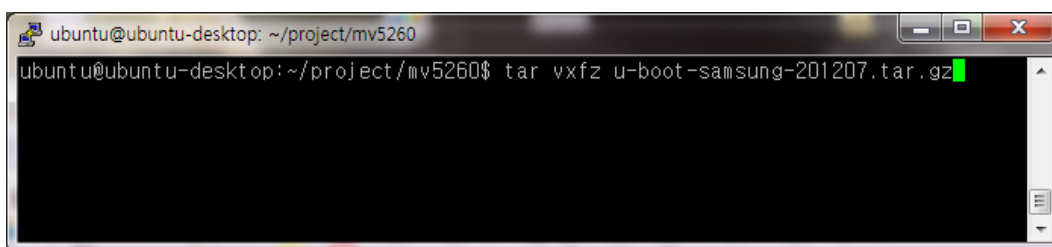
Generally, the Embedded Linux BSP is composed of 3 image files:

Embedded Linux BSP = Boot Loader + Kernel + File System

Boot Loader is the program necessary to load the kernel to the memory.

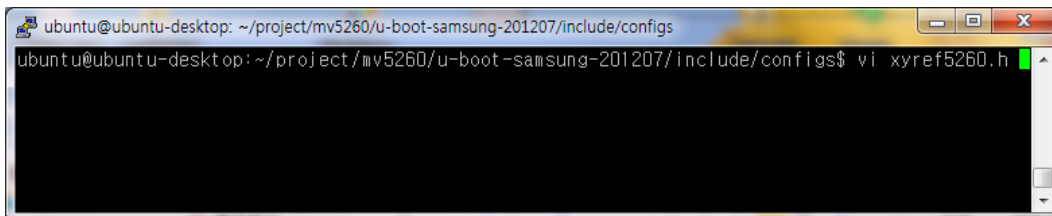
Enter in the following for file decompression:

```
# tar vxzf u-boot-samsung-5410.tar.gz
```



As shown below using the "vi" editor, open the file "[smdk5410.h](#)" and you will find the basic environment at its default. (ex: TFTP, CPU clock, DDR Program Counter)

```
# vi include/configs/smdk5410.h
```



Smdk5410.h Content

The prompt name on the mv-v210 boot board after booting the new bootloader program:

```
#define CONFIG_SYS_PROMPT      "smdk5410 # "
```

2.2. U-boot Compilation

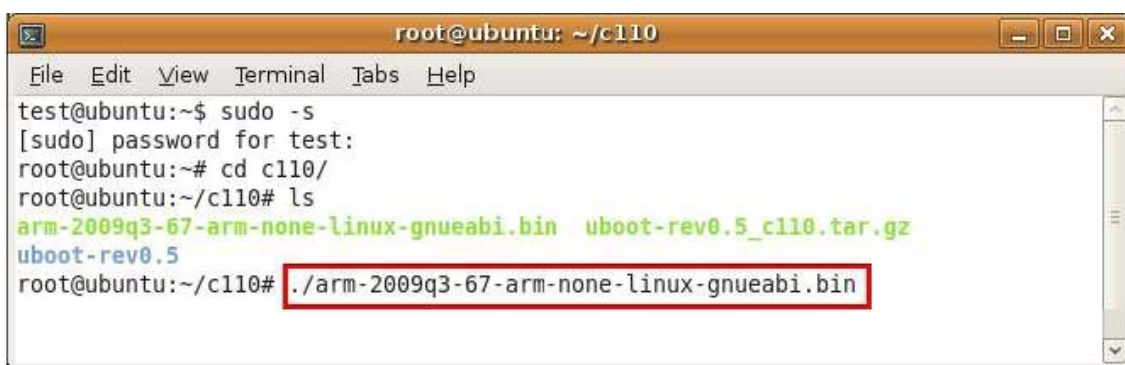
Install [arm-2009q3-67-arm-none-linux-gnueabi.bin](#) which is in

D→ /SRC/Android/q3-compiler

When installing Q3, you must install it on a Linux PC environment, not the console.

Installing procedures:

```
# ./arm-2009q3-67-arm-none-linux-gnueabi.bin
```

A terminal window titled 'root@ubuntu: ~/c110' showing the following commands and output:

```
test@ubuntu:~$ sudo -s
[sudo] password for test:
root@ubuntu:~# cd c110/
root@ubuntu:~/c110# ls
arm-2009q3-67-arm-none-linux-gnueabi.bin  uboot-rev0.5_c110.tar.gz
uboot-rev0.5
root@ubuntu:~/c110# ./arm-2009q3-67-arm-none-linux-gnueabi.bin
```

The command `./arm-2009q3-67-arm-none-linux-gnueabi.bin` is highlighted with a red box.

When this message displays “% [sudo dpkg-reconfigure -plow dash](#)” follow the steps below:

```
# sudo dpkg-reconfigure -plow dash
```

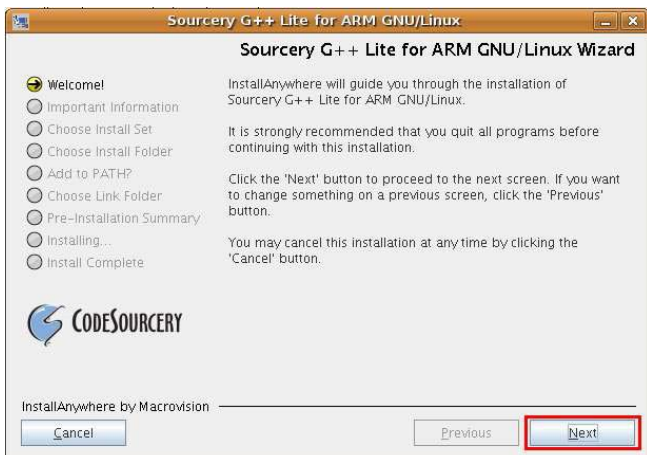
When a [\[yes/no\]](#) screen pops up, click “No” and enter in the command as shown below:

```
# ./sudo sh arm-2009q3-67-arm-none-linux-gnueabi.bin
```

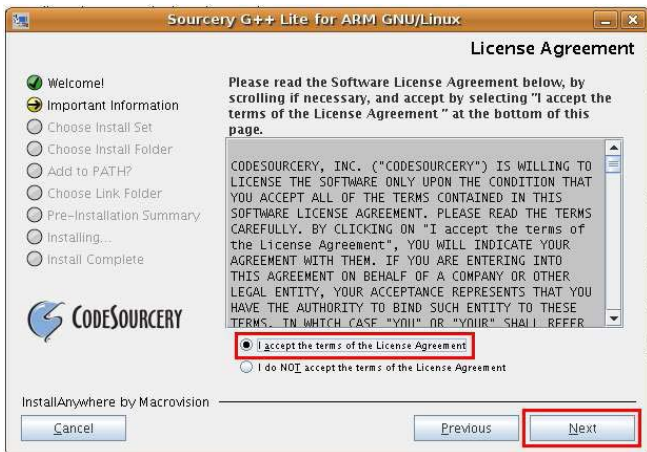
Below is a picture of the loading process:



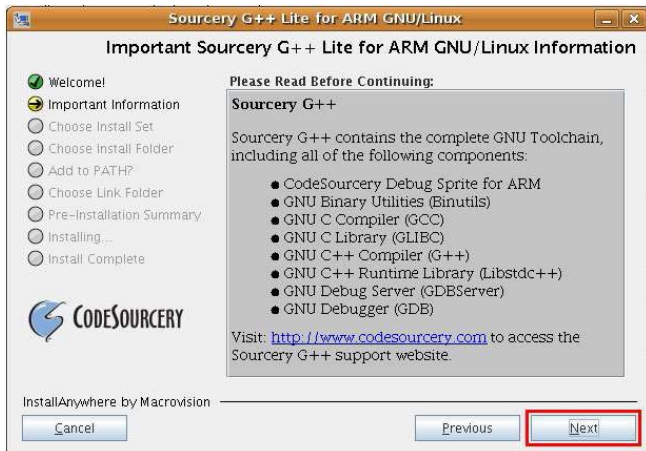
Click "Next"



Agree to the terms of the License Agreement then click "Next"



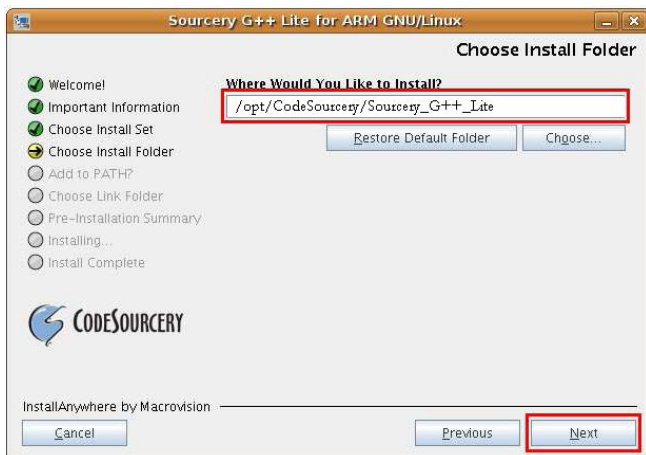
Click "Next"



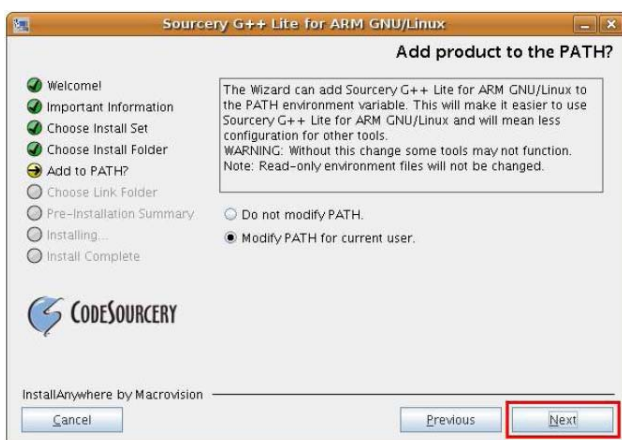
Click "Next"



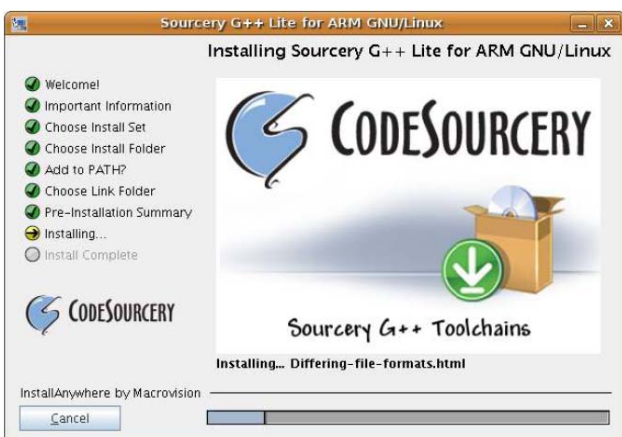
Click "Next"



Click "Next"



A series of "Next's" will lead to the screen as shown below. When the installation is complete, the Shell prompt will run automatically.



When the installation is complete, you can check the Q3 library which has been installed in `"/root/CodeSourcery/Sourcery_G++_Lite/bin"`

```

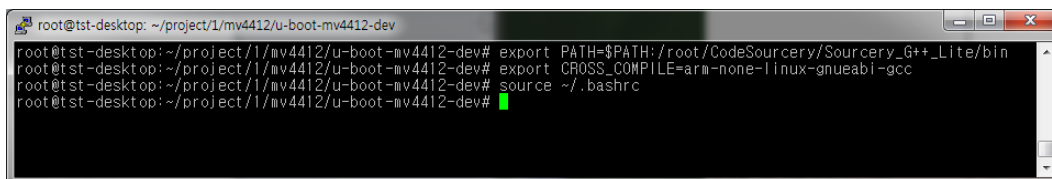
root@ubuntu: ~/c110/u-boot-rev0.5
File Edit View Terminal Tabs Help
root@ubuntu:~# cd CodeSourcery/Sourcery_G++_Lite/bin/
root@ubuntu:~/CodeSourcery/Sourcery_G++_Lite/bin# ls
arm-none-linux-gnueabi-addr2line  arm-none-linux-gnueabi-gprof
arm-none-linux-gnueabi-ar         arm-none-linux-gnueabi-ld
arm-none-linux-gnueabi-as         arm-none-linux-gnueabi-nm
arm-none-linux-gnueabi-c++       arm-none-linux-gnueabi-objcopy
arm-none-linux-gnueabi-c++filt   arm-none-linux-gnueabi-objdump
arm-none-linux-gnueabi-cpp       arm-none-linux-gnueabi-ranlib
arm-none-linux-gnueabi-g++       arm-none-linux-gnueabi-readelf
arm-none-linux-gnueabi-gcc       arm-none-linux-gnueabi-size
arm-none-linux-gnueabi-gcc-4.4.1 arm-none-linux-gnueabi-sprite
arm-none-linux-gnueabi-gcov      arm-none-linux-gnueabi-strings
arm-none-linux-gnueabi-gdb       arm-none-linux-gnueabi-strip
arm-none-linux-gnueabi-gdbtui
root@ubuntu:~/CodeSourcery/Sourcery_G++_Lite/bin#

```

Now we will work on Bashrc for the Boot Loader and kernel compilation.

Use the "vi" editor to open Bashrc.

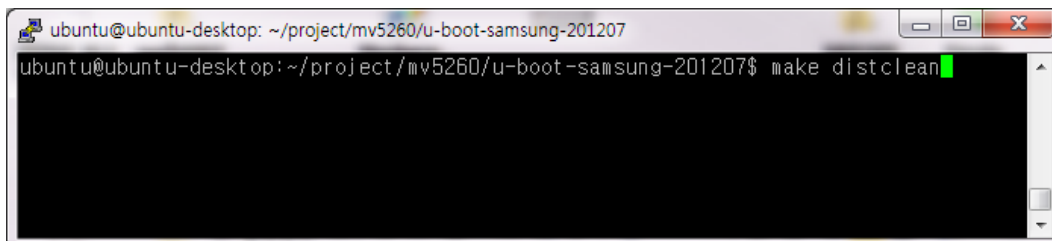
```
# export PATH=$PATH:/root/CodeSourcery/Sourcery_G++_Lite/bin
# export CROSS_COMPILE=arm-none-linux-gnueabi-
# source ~/.bashrc
```



```
root@st-desktop: ~/project/1/mv4412/u-boot-mv4412-dev
root@st-desktop:~/project/1/mv4412/u-boot-mv4412-dev# export PATH=$PATH:/root/CodeSourcery/Sourcery_G++_Lite/bin
root@st-desktop:~/project/1/mv4412/u-boot-mv4412-dev# export CROSS_COMPILE=arm-none-linux-gnueabi-gcc
root@st-desktop:~/project/1/mv4412/u-boot-mv4412-dev# source ~/.bashrc
root@st-desktop:~/project/1/mv4412/u-boot-mv4412-dev#
```

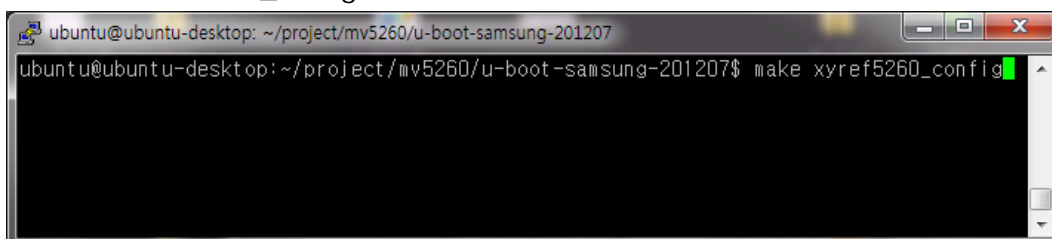
Compilation Steps: make distclean -> make smdk5410_config -> make

```
# make distclean
```



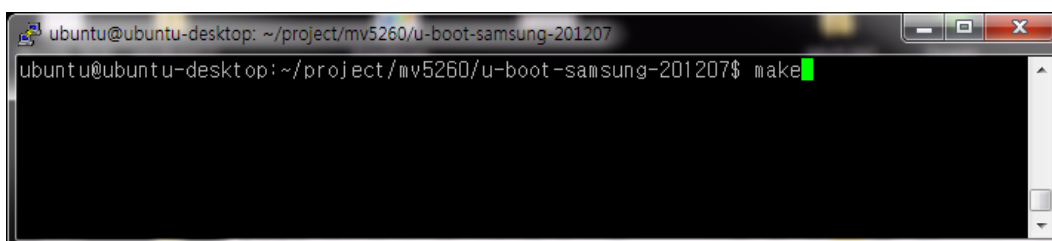
```
ubuntu@ubuntu-desktop: ~/project/mv5260/u-boot-samsung-201207
ubuntu@ubuntu-desktop:~/project/mv5260/u-boot-samsung-201207$ make distclean
```

```
# make smdk5410_config
```



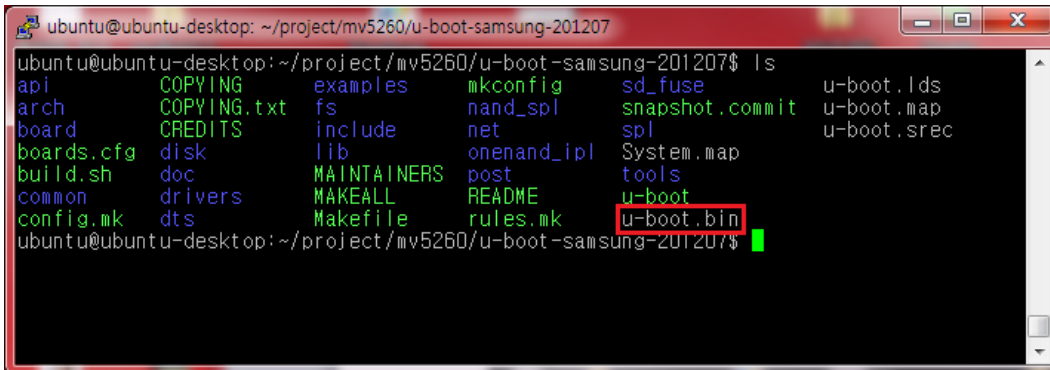
```
ubuntu@ubuntu-desktop: ~/project/mv5260/u-boot-samsung-201207
ubuntu@ubuntu-desktop:~/project/mv5260/u-boot-samsung-201207$ make xyref5260_config
```

```
# make
```



```
ubuntu@ubuntu-desktop: ~/project/mv5260/u-boot-samsung-201207
ubuntu@ubuntu-desktop:~/project/mv5260/u-boot-samsung-201207$ make
```

When compilation is complete, u-boot.bin file is generated in /uboot.



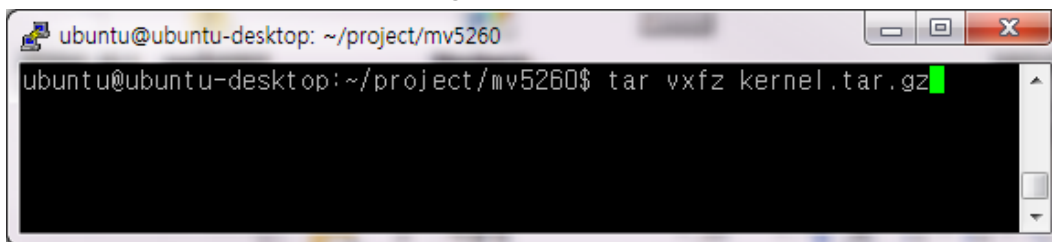
```
ubuntu@ubuntu-desktop: ~/project/mv5260/u-boot-samsung-201207
ubuntu@ubuntu-desktop: ~/project/mv5260/u-boot-samsung-201207$ ls
api          COPYING     examples   mkconfig   sd_fuse     u-boot.lds
arch        COPYING.txt fs          nand_spl   snapshot.commit u-boot.map
board       CREDITS    include    net         spl         u-boot.srec
boards.cfg  disk       lib        onenand_ipi System.map
build.sh    doc        MAINTAINERS post        tools
common     drivers    MAKEALL    README     u-boot
config.mk   dts       Makefile   rules.mk   u-boot.bin
ubuntu@ubuntu-desktop: ~/project/mv5260/u-boot-samsung-201207$
```

3. Kernel Setup

3.1. How to Compile

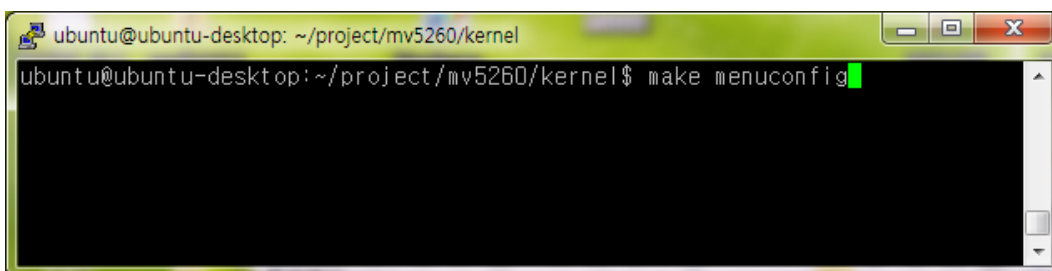
Enter in the following commands for decompression:

```
# tar vxzf kernel-3.4.5-5410.tar.gz
```



Put in the following commands for compilation to execute the kernel environment setup:

```
# make menuconfig
```



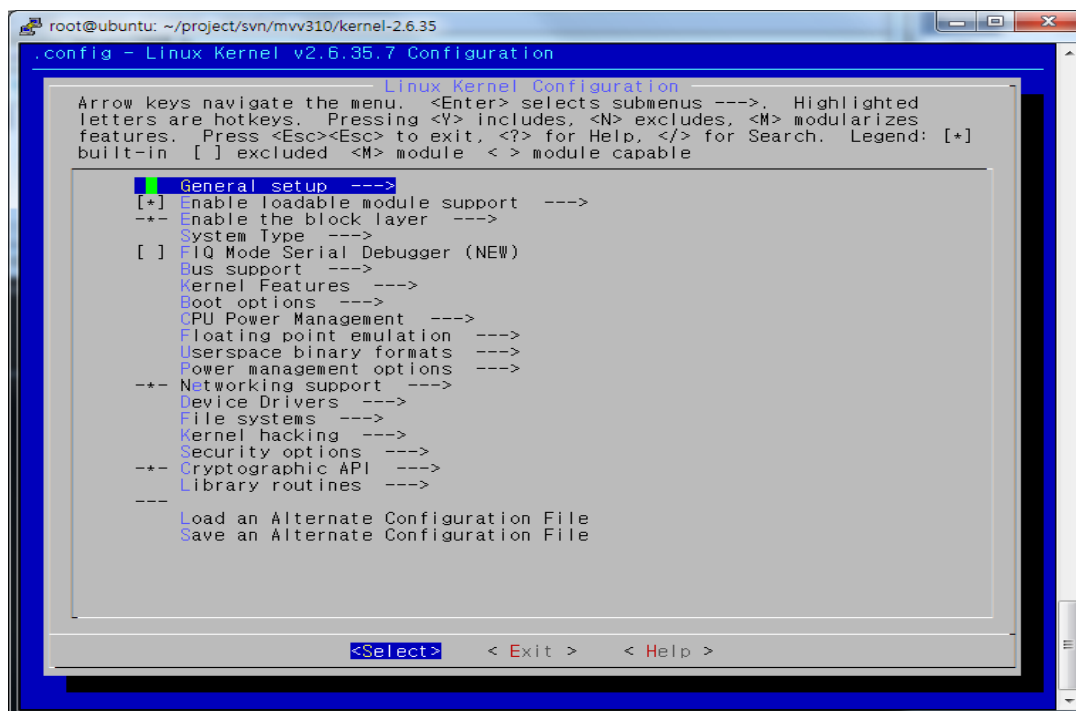
Besides make menuconfig, there are Kernel setting commands such as make config and make xconfig but the most popular one is the make menuconfig which is simple UI (User Interface) to use with the arrow keys known as the console (monitor) or telnet terminal is used for the Kernel Configuration.

If all the content of the setting menu is set, it doesn't have to be newly set in each time. So to save the previous configuration to a separate file, there is an option in the menu down below as "Save Configuration to an Alternate File". In opposite, previous setup configuration can be reloaded, Load and Kernel Configuration can be made by reading the file from "mv5410_defconfig" which is saved at arch/arm/configs/ which is Kernel Source directory.

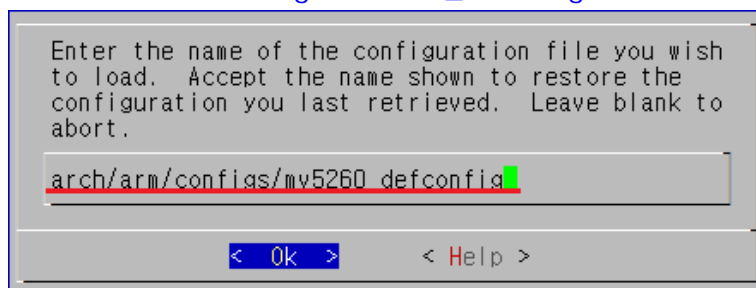
There is a "Save Configuration to an Alternate File" menu. On the other hand, you

can also load the configuration file. Load "mv210_linux_defconfig" which is under the kernel source directory arch/arm/configs/ .

Next, select the "Load an Alternate Configuration File" menu on the bottom section of the make menuconfig screen, and enter in the following:

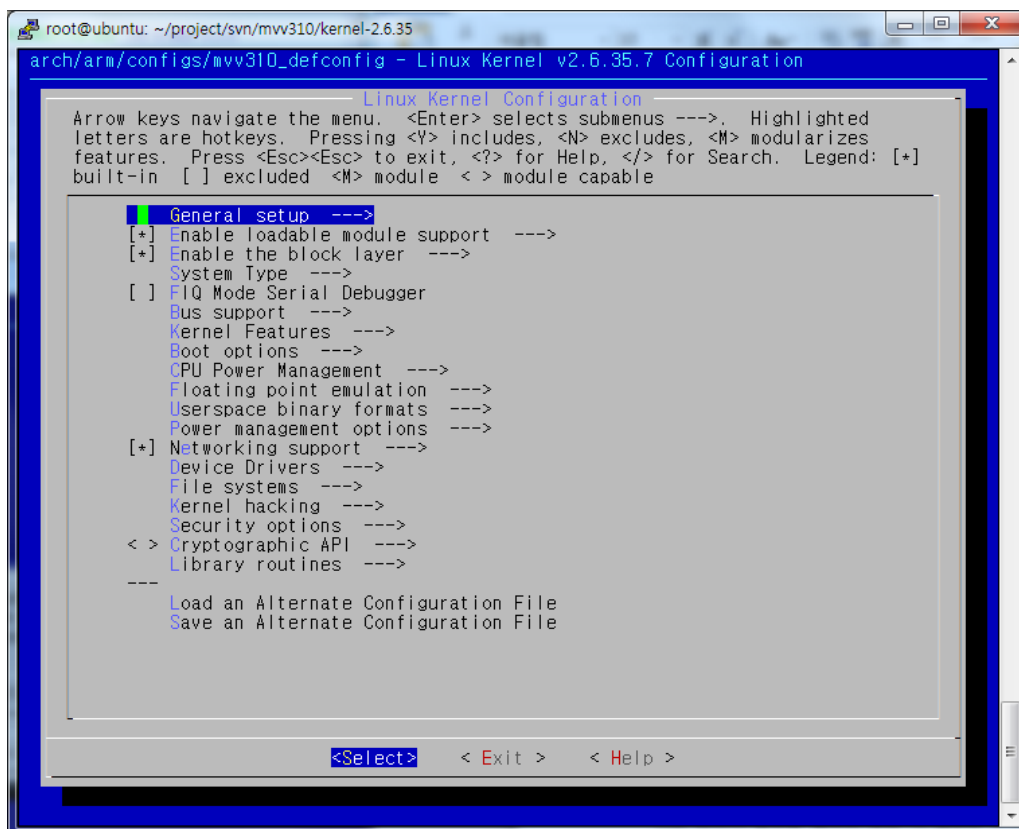


Load "arch/arm/configs/mv5410_defconfig"



The kernel configuration(make menuconfig) must be saved after the setup is complete. The kernel configuration is saved under the file name ".config" under the kernel source directory. The reason ".config" needs to be saved is that it will be checked during the "make dep" step, which is a crucial step for the compilation process. If a window asking to save pops up, make sure to answer "yes".

After loading is complete, exit.



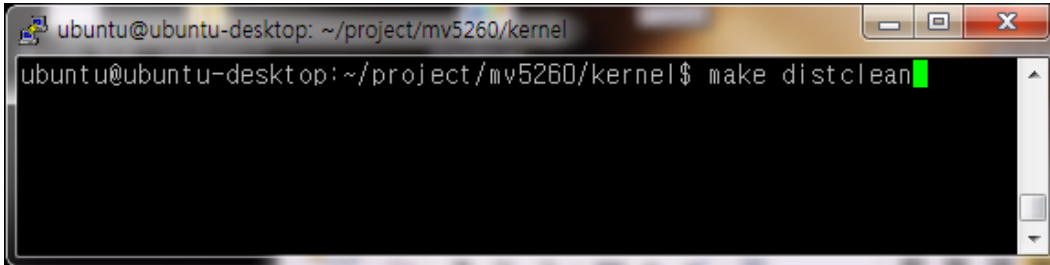
The Linux kernel image (zImage) making process is divided into compiling, linking, file type changing (ELF→BIN) by Binutil(objcopy), and file decompression (gzip). All of these combined make up the command “make” under Makefile.

For compiling kernel, you have to set up include in Android Toolchain.

```
# export PATH=$PATH:/root/CodeSourcery/Sourcery_G++_Lite/bin
# export CROSS_COMPILE=arm-eabi-
# source ~/.bashrc
```

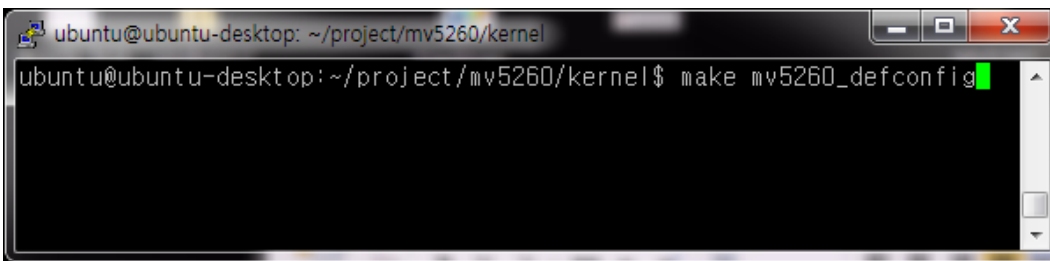
Compilation Steps: make distclean -> make mv5410_defconfig -> make

make distclean



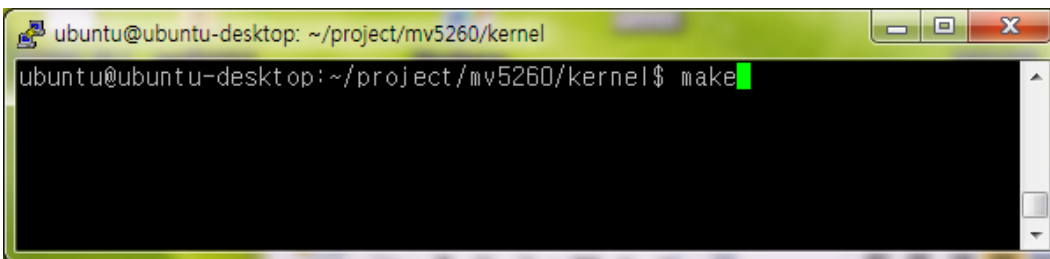
```
ubuntu@ubuntu-desktop: ~/project/mv5260/kernel
ubuntu@ubuntu-desktop: ~/project/mv5260/kernel$ make distclean
```

make mv5410_defconfig



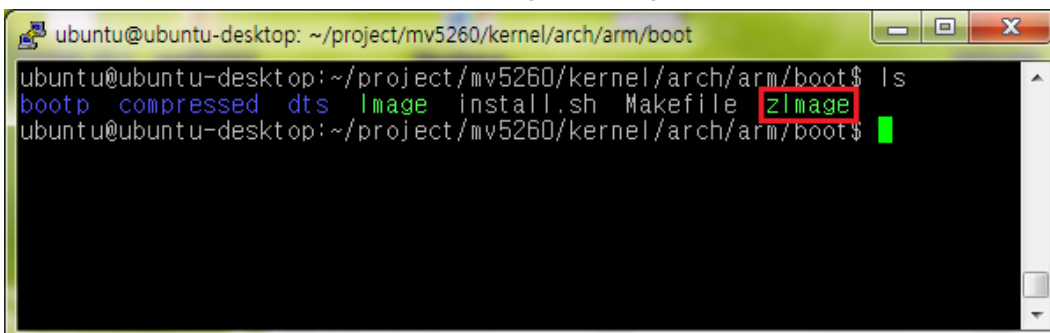
```
ubuntu@ubuntu-desktop: ~/project/mv5260/kernel
ubuntu@ubuntu-desktop: ~/project/mv5260/kernel$ make mv5260_defconfig
```

make



```
ubuntu@ubuntu-desktop: ~/project/mv5260/kernel
ubuntu@ubuntu-desktop: ~/project/mv5260/kernel$ make
```

When compilation is complete, zImage file is generated in kernel/arch/arm/boot.

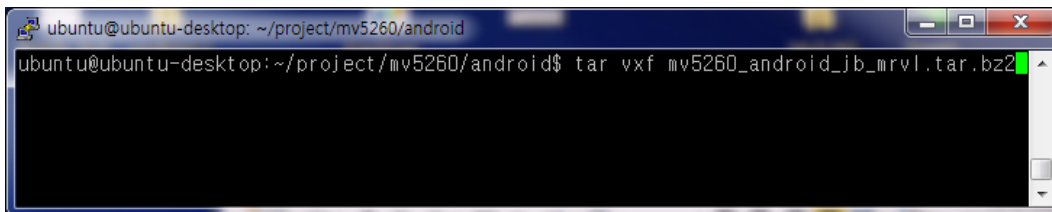


```
ubuntu@ubuntu-desktop: ~/project/mv5260/kernel/arch/arm/boot
ubuntu@ubuntu-desktop: ~/project/mv5260/kernel/arch/arm/boot$ ls
bootp compressed dts image install.sh Makefile zImage
ubuntu@ubuntu-desktop: ~/project/mv5260/kernel/arch/arm/boot$
```

4. Jelly Bean Compilation

Enter in the following command:

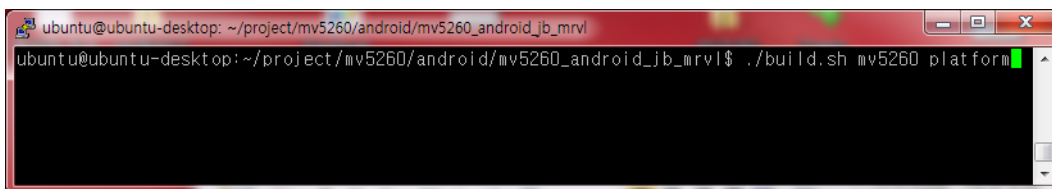
```
# tar vxf jellybean.tar.gz
```



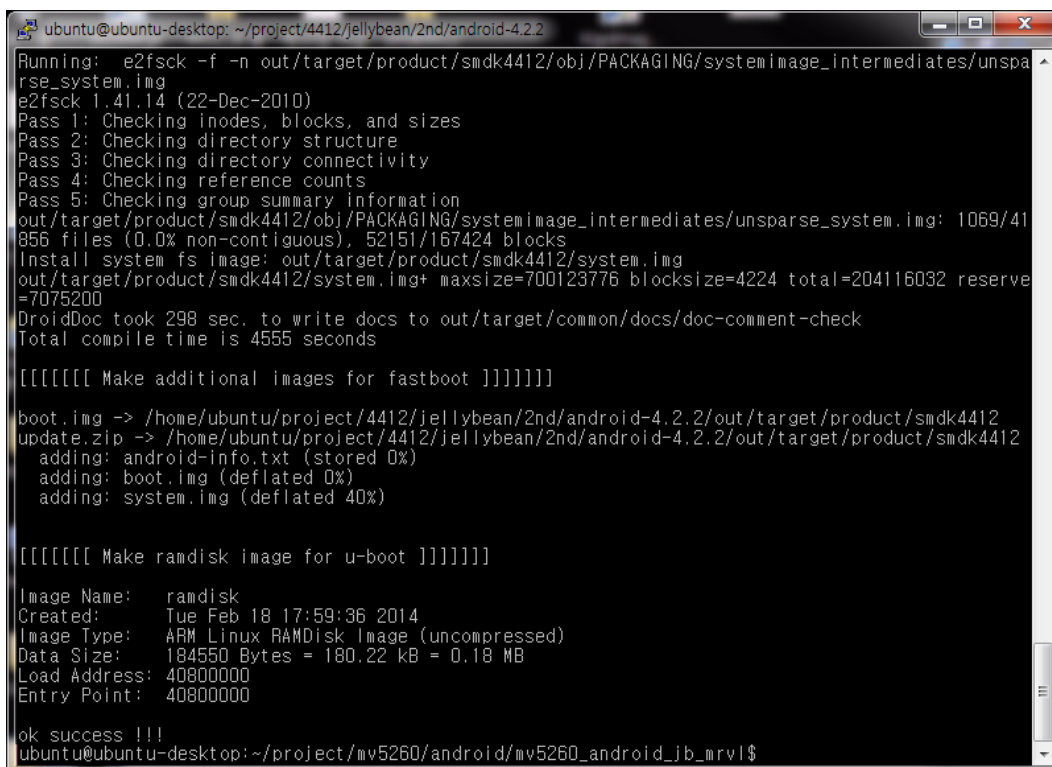
```
ubuntu@ubuntu-desktop: ~/project/mv5260/android
ubuntu@ubuntu-desktop:~/project/mv5260/android$ tar vxf mv5260_android_jb_mrvt.tar.bz2
```

This is command for compilation.

```
# ./mv5410.sh
```



```
ubuntu@ubuntu-desktop: ~/project/mv5260/android/mv5260_android_jb_mrvt
ubuntu@ubuntu-desktop:~/project/mv5260/android/mv5260_android_jb_mrvt$ ./build.sh mv5260 platform
```



```
ubuntu@ubuntu-desktop: ~/project/4412/jellybean/2nd/android-4.2.2
Running: e2fsck -f -n out/target/product/smdk4412/obj/PACKAGING/systemimage_intermediates/unspar
rse_system.img
e2fsck 1.41.14 (22-Dec-2010)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
out/target/product/smdk4412/obj/PACKAGING/systemimage_intermediates/unsparse_system.img: 1069/41
856 files (0.0% non-contiguous), 52151/167424 blocks
Install system fs image: out/target/product/smdk4412/system.img
out/target/product/smdk4412/system.img+ maxsize=700123776 blocksize=4224 total=204116032 reserve
=7075200
DroidDoc took 298 sec. to write docs to out/target/common/docs/doc-comment-check
Total compile time is 4555 seconds

[[[[[[[ Make additional images for fastboot ]]]]]]]

boot.img -> /home/ubuntu/project/4412/jellybean/2nd/android-4.2.2/out/target/product/smdk4412
update.zip -> /home/ubuntu/project/4412/jellybean/2nd/android-4.2.2/out/target/product/smdk4412
adding: android-info.txt (stored 0%)
adding: boot.img (deflated 0%)
adding: system.img (deflated 40%)


[[[[[[[ Make ramdisk image for u-boot ]]]]]]]

Image Name: ramdisk
Created: Tue Feb 18 17:59:36 2014
Image Type: ARM Linux RAMDisk Image (uncompressed)
Data Size: 184550 Bytes = 180.22 kB = 0.18 MB
Load Address: 40800000
Entry Point: 40800000

ok success !!!
ubuntu@ubuntu-desktop:~/project/mv5260/android/mv5260_android_jb_mrvt$
```

Successfully built image

Image is in the folder Android/out/target/product/mv5260



```
ubuntu@ubuntu-desktop: ~/project/mv5260/android/mv5260_android_jb_mrvt/out/target/product/mv5260
ubuntu@ubuntu-desktop:~/project/mv5260/android/mv5260_android_jb_mrvt/out/target/product/mv5260$ ls
android-info.txt  clean_steps.mk      obj                symbols            userdata.img
boot.img          data                 previous_build_config.mk  system             zImage
cache             fake_packages       ramdisk.img       system.img
cache.img         installed-files.txt  root              update.zip
ubuntu@ubuntu-desktop:~/project/mv5260/android/mv5260_android_jb_mrvt/out/target/product/mv5260$
```