

Android6.0 编译系统介绍

Version: 1.0

Date: 2016-06-20



www.spreadtrum.com

重要声明

版权声明

本文档中的任何内容受《中华人民共和国著作权法》的保护，**版权所有 © 2016 展讯通信有限公司，保留所有权利**，但注明引用其他方的内容除外。

商标声明

展讯通信有限公司和展讯通信有限公司的产品是展讯通信有限公司专有。在提及其他公司及其产品时将使用各自公司所拥有的商标，这种使用的目的仅限于引用。

不作保证声明

展讯通信有限公司不对此文档中的任何内容作任何明示或暗示的陈述或保证，而且不对特定目的的适销性及适用性或者任何间接、特殊或连带的损失承担任何责任。

保密声明

本文档（包括任何附件）包含的信息是保密信息。接收人了解其获得的本文档是保密的，除用于规定的目的外不得用于任何目的，也不得将本文档泄露给任何第三方。

前 言

文档说明

本文档简要介绍了展讯公司 Android6.0 平台 32bit 芯片的编译系统及其使用方法。

阅读对象

本文档针对于所有展讯公司 Android6.0 平台 32bit 芯片产品客户软件工程师。


内容介绍

本文档包括三个章节，分别为：

- 第一章：编译方法介绍
- 第二章：新建工程和配置介绍
- 第三章：其他相关配置

文档约定

本文档采用下面醒目标志来表示在操作过程中应该特别注意的地方。

 注意：

提醒操作中应注意的事项。

相关文档

目 录

第 1 章 代码编译的方法	4
1.1 适用范围	4
1.2 代码和编译环境的准备	4
1.3 完成一次全新的编译	5
1.4 单项编译和其它编译命名	8
1.5 编译的成果	9
第 2 章 新建和配置一个项目	12
2.1 新建项目编译配置文件	12
2.2 配置新项目的 kernel 部分	15
2.3 配置新项目的 u-boot 部分	18
2.4 配置 chipram 部分	19
2.5 完成配置准备编译	20
第 3 章 其它编译相关的内容	21
3.1 OTA 包的编译	21
3.2 如何制作多国语言版本	21
附录 A Revision History	错误!未定义书签。

第1章 代码编译的方法

1.1 适用范围

该文档适用于展讯 SC7731G、SC9830、SC9832、SC7731C 芯片在 android6.0 编译和配置。

1.2 代码和编译环境的准备

首先，客户需要解压完整的平台代码包，代码包由 TAM 向客户进行发布，其中包含代码，bin 档和开发调试工具等。其中 AP 侧代码由开源代码包和非开源库文件两部分组成

开源代码包部分一般命名为 idh.code，以 rar 或者 tgz 压缩格式提供

非开源库文件一般包括：

- 非开源程序和库：proprieties-<平台名>.tar.gz
- 芯片配置文件 conf-<平台名>.tar.gz

客户解压之后，需要将 proprieties-<平台名>.tar.gz 解压对应的库文件的内容拷贝到代码包的 vendor/sprd/proprieties/目录下。将 conf-<平台名>.tar.gz 解压出来的芯片和项目配置的文件拷贝到 device/sprd/下面。

客户需要检查自己的编译环境，google 推荐使用 64 位 ubuntu 的系统。展讯推荐 14.04 的版本。Ubuntu 10.04 - 12.04 版本也可以。可以使用下面命令来查看 ubuntu 的具体版本号：

```
lsb_release -a
```

- 需要安装 1.7 版本的 openjdk，可以使用下面命令来查看 jdk 的版本：

```
java -version
```

用下面命令安装 openjdk 1.7:

```
sudo apt-get update
```

```
sudo apt-get install openjdk-7-jdk
```

- Google 推荐的 python 版本是 2.6 或者 2.7，可以在 python.org 获得，可以使用下面命令来查看 python 的版本：

```
python --version
```

- 根据 ubuntu 版本的不同，可能还需要一些其它的编译支持工具，完整的工具包在下面的网

址可以找到:

<http://source.android.com/source/initializing.html>

例如 ubuntu14.04, 可使用如下命令, 进行初始化所需工具包:

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential \
zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 \
lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z-dev ccache \
libgl1-mesa-dev libxml2-utils xsltproc unzip
```

在完成的代码和编译环境的准备之后, 就可以开始进行代码的编译工作了。

1.3 完成一次全新的编译

在完成了代码环境的准备后就可以进行一个完整的编译了, 当然, 客户也可以选择在完成自定义项目配置之后再开始编译, 但是我们还是建议不熟悉展讯环境的客户在准备好代码之后先进行一次默认项目的编译

- 首先需要了解, lunch 选项和 device/sprd/scx.. 目录的对应关系, 如下图:



- 通过 ubuntu 终端命令行工具进入代码的根目录, 默认的代码根目录是 `idh.code`:

首先执行

```
source build/envsetup.sh
```

这一步将读取各个项目的编译配置文件, 然后执行

```
lunch
```

此时终端会显示出所有被配置过的项目的列表, 如下图所示:

```
SPREADTRUM\wenxue1.zhuang@sh60802upcw:~/work/Spreadtrum_Src/Moc6.0_Trunk_W16.10.4/idx.codes$ lunch
```

```
You're building on Linux
```

```
Lunch menu... pick a combo:
```

1. aosp_arm-eng
2. aosp_arm64-eng
3. aosp_mips-eng
4. aosp_mips64-eng
5. aosp_x86-eng
6. aosp_x86_64-eng
7. sp7731c_1h10_32v4_native-userdebug
8. sp7731c_1h10_32v4_oversea-userdebug
9. sp7731c_1h10_32v4_telcel-userdebug
10. sp7731c_1h10_32v4_vodafone-userdebug
11. sp7731c_1h10_32v4_orange-userdebug
12. sp7731c_1h10_native-userdebug
13. sp7731c_1h10_multi-userdebug
14. sp7731c_1h10_telcel-userdebug
15. sp7731c_1h10_vodafone-userdebug
16. sp7731c_1h10_orange-userdebug
17. sp7731c_1h10_oversea-userdebug
18. sp9830a_5h10_5mvolte-userdebug
19. sp9830a_5h10_4mvolteril-userdebug
20. sp9830a5_32v4_4mvolsea-userdebug
21. sp9830a_7h10_5mvolte-userdebug
22. sp9830i_j3lte-userdebug
23. sp9832a_2h11_4m-userdebug
24. sp9832a_2h11_5msinglsim-userdebug
25. sp9832a_2h11_4mvodafone-userdebug
26. sp9832a_2h11_4morange-userdebug
27. sp9832a_2h11_volte-userdebug
28. sp9832a_2h11_4mvoltesea-userdebug
29. sp9832a_2h11_voltetelc-userdebug
30. sp9832a_2h10_4mvolte-userdebug
31. sp9832a_2h10_4mvolteril-userdebug
32. sp9832a2_32v4_4mvolsea-userdebug
33. sp9832a_3h10_5mvolte-userdebug
34. sp9832a_3h10_voltecucc-userdebug
35. sp9832a_3h10_voltecucc-userdebug
36. sp7730sw_1h10_native-userdebug
37. sp7731g_1h10_32v4_plus-userdebug
38. sp7731g_1h10_32v4_adf-userdebug
39. sp7731g_1h10_hd_native-userdebug
40. sp7731g_1h10_hd_oversea-userdebug
41. sp7731g_1h10_hd_multi-userdebug
42. sp8730se_1h10_qhd_native-userdebug

```
Which would you like? [aosp_arm-eng]
```

选择需要编译的 debug 版本，输入对应的数字或项目名，如：17 或 sp7731c_1h10_oversea-userdebug

如果想使用 user 版本，则直接输入对应的项目名并去除 debug 关键字，例如使用 17 对应的 user 版本：

```
which would you like? [aosp_arm-eng] sp7731c_1h10_oversea-user
=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=6.0
TARGET_PRODUCT=sp7731c_1h10_oversea
TARGET_BUILD_VARIANT=user
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a-neon
TARGET_CPU_VARIANT=cortex-a7
TARGET_2ND_ARCH=
TARGET_2ND_ARCH_VARIANT=
TARGET_2ND_CPU_VARIANT=
HOST_ARCH=x86_64
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.2.0-23-generic-x86_64-with-Ubuntu-12.04-precise
HOST_BUILD_TYPE=release
BUILD_ID=MRA58K
OUT_DIR=out
=====
```

PLATFORM_VERSION: 为平台对应的版本, 我们使用的是 6.0 版本

TARGET_PRODUCT: 要编译的工程名

TARGET_BUILD_VARIANT: 显示要编译的工程是 user 版本还是 userdebug 版本

TARGET_ARCH: 为对应的硬件信息

TARGET_CPU_VARIANT: cpu 信息

HOST_ARCH: 编译平台对应的硬件信息

OUT_DIR: 编译生成的目标文件目录

其中 base 或 plus 关键字分别代表单卡或者双卡方案。这里建议客户选择最接近自己项目形态的参考项目。

更新内核头文件

kheader

在选择完编译项目后, 使用

make

命令来进行编译, 如果编译使用的机器是支持多线程编译的, 则可以使用 -j 选项来加快编译的速度, 比如

make -j24

-j 之后的数值表示多线程并行编译, 主要在于编译器是否支持多线程并行编译, 同时跟 cpu 有很大关系。一次全新的编译根据编译服务器的性能大约需要几十分钟到几个小时不等。

1.4 单项编译和其它编译命名

在完成一次全编之后，在不改变当前编译项目的前提下，修改代码后可以使用单项的编译来编译对应的部分，加快开发的速度。

👁 注意：

如若打开新的终端并进行重新编译，则需执行 `source` 重新配置编译环境，并使用 `lunch` 的操作选择对应的编译项目

- 各个部分的编译命令如下

1. 单独编译 u-boot

```
make bootloader
```

主要生成目标文件: `fdl2.bin` `u-boot.bin`

2. 单独编译 fdl1 和 uboot-16k

```
make chipram
```

主要生成目标文件: `fdl1.bin` `u-boot-spl-16k.bin`

3. 单独编译 boot image

```
make bootimage
```

主要生成目标文件: `boot.img` `dt.img` `kernel` `ramdisk.img`

4. 单独编译 system image

```
make systemimage
```

5. 单独编译 userdata image

```
make userdataimage
```

主要生成目标文件: `userdata.img`

我们还可以单独编译 android 的每一个模块，比如单独编译一个 apk，一个 java 或者本地库或者本地程序，这时我们需要进入到对应模块的 `Android.mk` 所在的目录，执行 `mm` 指令，比如需要重新编译“设置”这个 apk，我们就需要这样做

```
cd packages/apps/Settings/
```

```
mm
```

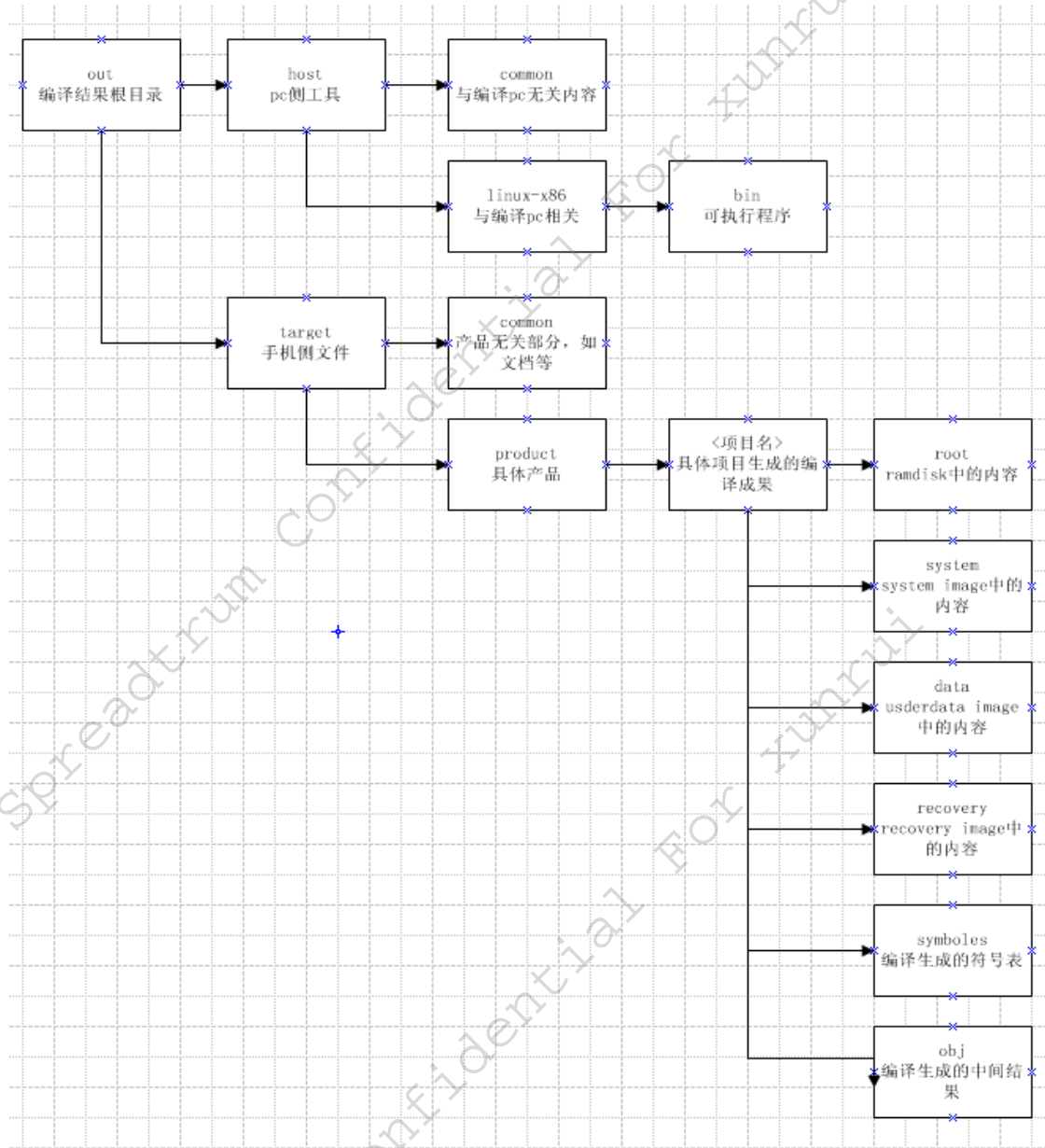
这样被单独编译出来的模块可以通过 adb push 的方式推入调试手机进行使用，是调试阶段被经常使用到的方式。

👁 注意：

ramdisk（手机根目录或者/bin 目录）中的文件不建议使用 adb push，需要重新下载 bootimage

1.5 编译的成果

Android 的编译输出路径为 out，编译成果如下图：



其中最重要的目录就是 out/target/product/<项目名>，这里存放着用于下载的所有 bin 和 image 文件，包括 fd11.bin fd12.bin u-boot-spl-16k.bin u-boot.bin boot.img system.img usderdata.img recovery.img cache.img。

注意：

并非所有的下载用文件都是编译生成的，比如 CP 侧的 bin 就是在版本发布中直接提供，如对于 sc7731c，路径：device/sprd/scx20/<project>/AndroidBoard.mk，通过 \$(call add-radio-file,modem

_bins/xxx.bin)方式拷贝 cp 侧的 bin 文件。在执行 make dist 时，会执行 copy 。

out/target/product/<项目名>/root

out/target/product/<项目名>/system

out/target/product/<项目名>/data

out/target/product/<项目名>/recovery

这四个目录分别是 boot system userdata 和 recovery image 中的直接内容，其中的文件和手机运行后各个对应分区中的内容是一一对应的，当我们通过 mm 指令来编译某个特定的 Android 模块时，更新的也是这些目录中的文件。

另外编译的符号表在很多调试和 bug 解决中是非常重要的，所有符号表可以在 out/target/product/<项目名>/syboles 目录下找到，我们也可以在 out/target/product/<项目名>/obj 目录下找到同样的内容，不同的是 obj 目录更加具体，不仅仅有符号表，而且有所有 c/c++ java 文件的中间编译结果。同时，kernel 的符号表 vmlinux 是一个包含 linux kernel 的静态链接的可执行文件（比如在解析 sysdump 时，就需要提供此文件）。Vmlinux 对应的路径在 out/target/product/<项目名>/obj/KERNEL 目录下。

在 out/target/host/linux-x86/bin 目录下有一些常用的 PC 侧工具，包括 fastboot mkbooting adb 等。

第2章 新建和配置一个项目

2.1 新建项目编译配置文件

项目的编译配置文件所在目录为 device/sprd/scx 芯片/项目名，由于项目的编译配置内容较多，我们建议客户选择一个已有的项目作为参考，通过拷贝的方式新建自己的项目，通常选择和自己所要创建的比较接近的项目作为参考。例如，可以以 SC7731C 平台 sp7731c_1h10 为模版新建一个 sp7731c_test_1h10 项目（在 device/sprd/scx20 目录下）：

```
cp sp7731c_1h10 sp7731c_test_1h10 -r
```

- 在完成拷贝后，首先修改在 sp7731c_test_1h10/AndroidProducts.mk 中包含在该项目下的各工程：

```
2 PRODUCT_MAKEFILES := \
3   $(LOCAL_DIR)/sp7731c_test_1h10_native.mk \
4   $(LOCAL_DIR)/sp7731c_test_1h10_multi.mk \
5   $(LOCAL_DIR)/sp7731c_test_1h10_telcel.mk \
6   $(LOCAL_DIR)/sp7731c_test_1h10_vodafone.mk \
7   $(LOCAL_DIR)/sp7731c_test_1h10_orange.mk \
8   $(LOCAL_DIR)/sp7731c_test_1h10_oversea.mk
```

- 接着将 sp7731c_test_1h10 工程中对应的各 mk 修改和 AndroidProducts.mk 中的各工程同名，以 sp7731c_1h10_oversea.mk 为例(客户可以根据自己的需求，可以在此项目上派生出更多或仅配置需要的产品)：

```
mv sp7731c_1h10_oversea.mk sp7731c_test_1h10_oversea.mk
```

- 同时修改 root 路径下的 init.xx.rc 文件：

Init.xx.rc 文件主要：动态加载 ko 模块、创建相关目录、修改权限、启动 system 下的服务等操作。这两个 rc 文件会被 init.rc 调用，了解更多可参考 2.2 节中配置中->修改 ro.hardware 部分。

```
device/sprd/scx20/sp7731c_test_1h10/rootdir/root$
```

```
mv init.recovery.sp7731c_1h10.rc init.recovery.sp7731c_test_1h10.rc
```

```
mv init.sp7731c_1h10.rc init.sp7731c_test_1h10.rc
```

- 修改项目名 **TARGET_BOARD**。也是 **out** 目录后面输出的项目名目录

注意：**TARGET_BOARD** 定义可能在其他公用 **mk** 文件中被 **include** 进来的。例如 **sp7731c_test_1h10_oversea.mk**。

include device/sprd/scx20/sp7731c_test_1h10/sp7731c_test_1h10_oversea.mk

```

1 TARGET_PLATFORM := sc8830
2 TARGET_BOARD := sp7731c_test_1h10
3
4 PLATDIR := device/sprd/scx20
5 PLATCOM := $(PLATDIR)/common
6 BOARDDIR := $(PLATDIR)/$(TARGET_BOARD)
7 ROOTDIR := $(BOARDDIR)/rootdir
8 ROOTCOM := $(PLATCOM)/rootdir
9
10 # include the base version features
11 include $(PLATCOM)/plus.mk
12
13 CHIPRAM_DEFCONFIG := sp7731cea
14 UBOOT_DEFCONFIG := sp7731cea
15 KERNEL_DEFCONFIG := sp7731cea_dt_defconfig
16 DT5_DEFCONFIG := sprd-scx20_sp7731cea

```

- 修改各个产品的 **mk** 文件中的产品名。

以 **sp7731c_test_1h10_oversea.mk** 为例：

```

119 # Overrides
120
121 # Overrides
122 PRODUCT_NAME := sp7731c_test_1h10_oversea
123 PRODUCT_DEVICE := $(TARGET_BOARD)
124 PRODUCT_BRAND := SP7731CEA
125 PRODUCT_BRAND := SP7731CEA
126 PRODUCT_MANUFACTURER := SP7731CEA
127 PRODUCT_USE_FPGA := false
128 PRODUCT_USE_OPENPHONE := false
129
130 ifndef PRODUCT_LOCALES

```

PRODUCT_NAME //产品名

PRODUCT_MODEL //一般同项目名

- 在 **vendorsetup.sh** 将产品添加到 **lunch** 编译选项

需要修改的是 **add_lunch_combo**，如

add_lunch_combo <产品名>-userdebug

至此，在 **android** 根目录下，重新执行 **envsetup.sh** 和 **lunch** 之后就会看到新增加的项目产品了(客户根据自己的需求添加新的项目，没必要全部增加，此处只是例样)。

```

20. sp7731c_1h10_orange-userdebug
21. sp7731c_1h10_oversea-userdebug
22. sp7731c_test_1h10_native-userdebug
23. sp7731c_test_1h10_multi-userdebug
24. sp7731c_test_1h10_telcel-userdebug
25. sp7731c_test_1h10_vodafone-userdebug
26. sp7731c_test_1h10_orange-userdebug
27. sp7731c_test_1h10_oversea-userdebug
28. sharklc_z2lte-userdebug
29. sp9830a_5h10_5mvolte-userdebug

```

- 在 **system.prop** 中添加自己定义的硬件信息

需要修改的是 **ro.product.hardware**

ro.product.hardware=SP7731C_TEST_V1.0.0 //此处关系不大，客户可修改或不做修改

- 配置 BoardConfig.mk

TARGET_BOOTLOADER_BOARD_NAME //u-boot 所使用的板级配置名称

```
#ext4 partition
include $(BOARDIR)/BoardPartitionConfig.mk

# board configs
TARGET_BOOTLOADER_BOARD_NAME := sp7731cea

# select camera 2M,3M,5M,8M
CAMERA_SUPPORT_SIZE := 5M
TARGET_BOARD_NO_FRONT_SENSOR := false
TARGET_BOARD_FRONT_CAMERA_ROTATION := false
```

- 在对应项目中(sp7731c_test_1h10_oversea.mk)中添加 kernel 和 u-boot 的支持

需要修改的项包括

CHIPRAM_DEFCONFIG //chipram 所使用的配置文件名

UBOOT_DEFCONFIG //u-boot 所使用的配置文件名

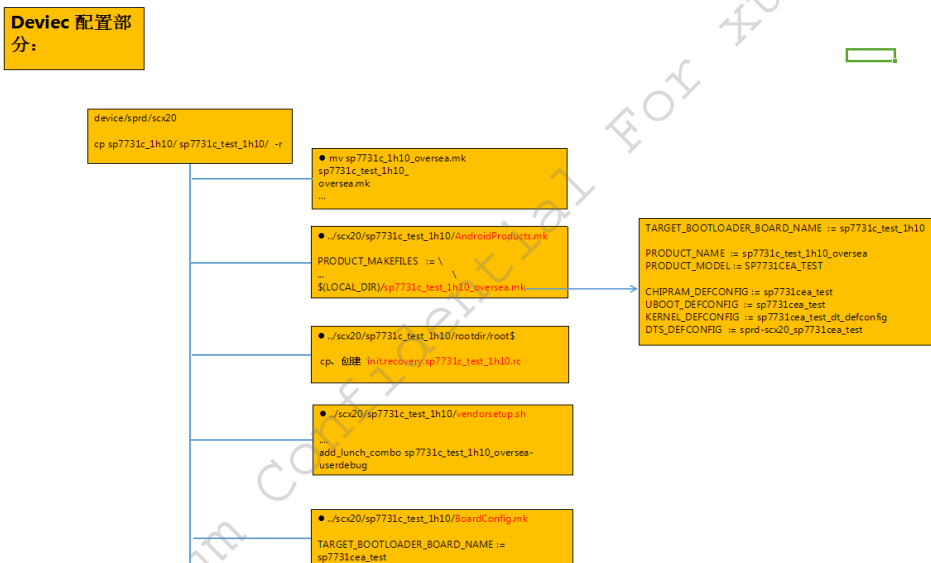
KERNEL_DEFCONFIG //kernel config 文件名

DTS_DEFCONFIG //dts config 文件名

这些项的配置是用于配置 u-boot 与 kernel 的，在 u-boot 配置与 kernel 配置的部分会具体解释它们的使用方法。例如修改如下：

```
7 ROOTDIR := $(BOARDIR)/rootdir
8 ROOTCOMM := $(PLATCOM)/rootdir
9
10 # include the base version features
11 include $(PLATCOM)/plus.mk
12
13 CHIPRAM_DEFCONFIG := sp7731cea
14 UBOOT_DEFCONFIG := sp7731cea
15 KERNEL_DEFCONFIG := sp7731cea_dts_defconfig
16 DTS_DEFCONFIG := sprd-scx28_sp7731cea
17
18 #use sprd's four(wifi bt gps fm) integrated one chip
19 USE_SPRD_WCN := true
20
21
22 BOARD_KERNEL_PAGESIZE := 2048
23 # 93 lines: BOARD_KERNEL_SEPARATED_DT := true
116 CONNECTIVITY_HW_CHISET := $(shell grep BOARD_SPRD_WCNBT $(BOARDIR)/BoardConfig.mk)
117 $(call inherit-product, vendor/sprd/open-source/res/connectivity/device-sprd-wcn.mk)
```

- Device 配置项，如图所示：



2.2 配置新项目的 kernel 部分

在完成了项目的编译配置后，我们需要配置项目的 kernel 部分，这是通过向 kernel 中添加一个新的 board 来实现的

- 添加对应的 kernel defconfig

所有项目对应的 kernel config 文件都位于 kernel/arch/arm/configs 目录下，以 _defconfig 结尾。需要注意的是，kernel config 的文件名需要与编译配置中 BoardConfig.mk 文件中 KERNEL_DEFCONFIG 项的配置一致。我们同样建议通过拷贝参考项目对应文件的方式来添加 kernel config。例如：

```
cp sp7731cea_dt_defconfig sp7731cea_test_dt_defconfig
```

一旦完成了这个文件的添加，客户可以修改这个文件来改变 kernel 的配置。

- 增加 kernel CONFIG_MACH* config

```
kernel/arch/arm/mach-sc/Kconfig
```



```

143 config MACH_SP7731CEA
144 bool "sp7731cea Board"
145 depends on ARCH_SC
146 depends on ARCH_SCX35
147 depends on ARCH_SCX20
148 select CLKSRC_OF if OF
149 select COMMON_CLK if OF
150 default n
151 help
152 SP20EA board based on SC8830g serial.
153
154 config MACH_SP7731CEA_TEST
155 bool "sp7731cea test Board"
156 depends on ARCH_SC
157 depends on ARCH_SCX35
158 depends on ARCH_SCX20
159 select CLKSRC_OF if OF
160 select COMMON_CLK if OF
161 default n
162 help
163 SP7731CEA_TEST board based on SC8830g serial.
164
165 config MACH_SP7731CEB

```

- 添加__board-xxx.h , 建议从参考项目进行拷贝。例如:

```

cp kernel/include/soc/sprd/__board-sp7731cea.h
kernel/include/soc/sprd/__board-sp7731cea_test.h

```

```

10 * This program is distributed in the hope that it will be useful,
11 * but WITHOUT ANY WARRANTY; without even the implied warranty of
12 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 * GNU General Public License for more details.
14 */
15
16
17 #ifndef __ASM_ARCH_BOARD_H
18 #error "Don't include this file directly, include <mach/board.h>"
19 #endif
20
21 #define GPIO_TOUCH_RESET 81

```

- board.h 中增加 include __board-xxx.h

```

kernel/include/soc/sprd/board.h
#ifdef CONFIG_MACH_SP7731CEA_TEST
#include "__board-sp7731cea_test.h"
#endif

```

- 增加 board-xxx.c

```

kernel/arch/arm/mach-sc/Makefile
obj-$(CONFIG_MACH_SP7731CEA_TEST) += board-sp7731cea_test.o

```

- board-xxx.c 建议从参考项目进行拷贝。例如:

```

cp kernel/arch/arm/mach-sc/board-sp7731cea.c kernel/arch/arm/
mach-sc/board-sp7731cea_test.c

```

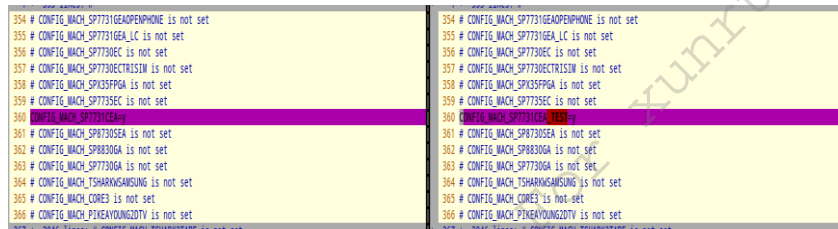
- 添加对应的 dts 文件

1. 首先, 修改对应的 kernel defconfig, 根据新建项目的名称正确配置相关 CONFIG_MACH<项目名>的 config

```

kernel/arch/arm/configs/

```



2. 然后, 根据 CONFIG_MACH 配置 kernel/arch/arm/boot/dts/Makefile

```
02 dtb-$(CONFIG_MACH_TIZEN3_3G) += sprd-scx35_tizen3_3g.dtb
03 dtb-$(CONFIG_MACH_SP7731CEA) += sprd-scx20_sp7731cea.dtb
04 dtb-$(CONFIG_MACH_SP7731CEA_TEST) += sprd-scx20_sp7731cea_test.dtb
05 dtb-$(CONFIG_MACH_SP7731CEB) += sprd-scx20_sp7731ceb.dtb
```

3. 最后, 新建或者拷贝项目的 dts 文件

```
cp kernel/arch/arm/boot/dts/sprd-scx20_sp7731cea.dts kernel/arch/arm/boot/
dts/sprd-scx20_sp7731cea_test.dts
```

- 修改 ro.hardware

具体在 init.cpp 中实现, 通过传参的方式将 dts 中的 androidboot.hardware 的参数传给 ro.hardware: androidboot.hardware→ro.boot.hardware→ro.hardware, init.rc 文件再根据 ro.hardware 具体的值加载对应的.rc 文件, 本项 init.rc 对应的是 init.sp7731c_1h10.rc。

对应路径: kernel/arch/arm/boot/dts/sprd-scx20_sp7731cea_test.dts

```
#size-cells = <1>;
interrupt-parent = <&gic>;

chosen {
    bootargs = "loglevel=1 console=ttyS1,115200n8 init=/init root=/dev/ram0 rw androidboot.hardware=sprd-scx20_sp7731c_1h10";
    linux,initrd-start = <0x85500000>;
    linux,initrd-end = <0x855a3212>;
};

memory: memory {
    device_type = "memory";
    reg = <0x80000000 0x40000000>;
};

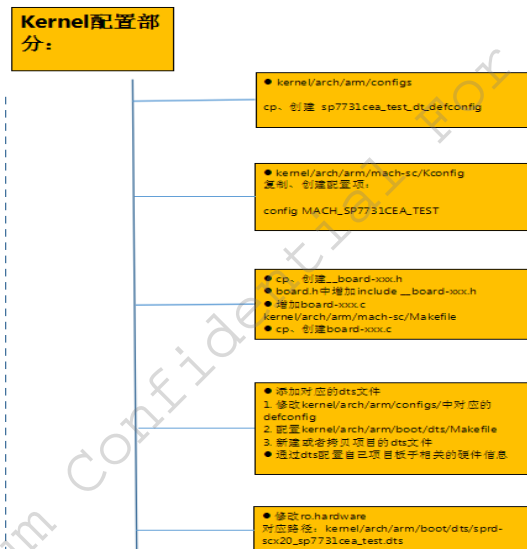
reserved-memory {
    #address-cells = <1>;
```

- 通过 dts 配置自己项目板子相关的硬件信息

```
kernel/arch/arm/boot/dts/sprd-scx20_sp7731cea_test.dts
```

这些信息包括使用的设备信息, gpio, i2c, lcd 等等, 具体请参考《Android5.1 客户化配置》。

- Kernel 配置项，如图所示：



2.3 配置新项目的 u-boot 部分

- 添加配置头文件

在 u-boot64/include/configs 添加 .h 文件，文件名必须与编译配置中 sp7731c_test_1h10_oversea.mk 文件中 UBOOT_DEFCONFIG 的值一致，同样建议拷贝添加。

```
idh.code/u-boot64/include/configs$
```

```
cp sp7731cea.h sp7731cea_test.h
```

- 添加 u-boot 的板级配置

在 u-boot64/board/spreadtrum 添加目录，目录的名称与编译配置中 BoardConfig.mk 文件中 TARGET_BOOTLOADER_BOARD_NAME 一致，内容从参考项目拷贝即可

```
idh.code/u-boot64/board/spreadtrum$
```

```
cp sp7731cea sp7731cea_test -r
```

- 添加 Makefile 支持

在 u-boot64/boards.cfg 中添加如下内容：

```

venxue1zhuang@androidl05:~/sprdroid6.0_trunk/u-boot64$ git diff boards.cfg
diff --git a/boards.cfg b/boards.cfg
index a7fd0cd..d8e1553 100644
--- a/boards.cfg
+++ b/boards.cfg
@@ -1307,6 +1307,7 @@
Active arm armv7 sc8830 spreadtrum sp8730seea_qhd sp8730seea_qhd sp8730seea_qhd -
Active arm armv7 sc8830 spreadtrum sp7731cea sp7731cea sp7731cea -
Active arm armv7 sc8830 spreadtrum sp7731cea_test sp7731cea_test sp7731cea_test -
Active arm armv7 sc8830 spreadtrum sp7731ceb sp7731ceb sp7731ceb -
Active arm armv7 sc8830 spreadtrum coreprime3g_ve coreprime3g_ve coreprime3g_ve -
Active arm armv7 sc8830 spreadtrum grandprime3g_ve grandprime3g_ve grandprime3g_ve -
venxue1zhuang@androidl05:~/sprdroid6.0_trunk/u-boot64$

```

上述配置项分别对应：Status, Arch, CPU:SPLCPU, SoC, Vendor, Board name, Target, Options, Maintainers。并且 mkconfig 会根据 boards.cfg 的配置进行解析, 提取各项。

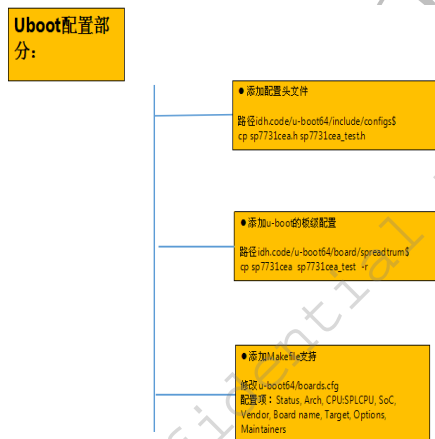
CPU:SPLCPU: armv7 //cpu 型号

Soc: sc8830 //芯片版本

Vendor: sptreadtrum //厂商号

Board name: sp7731cea_test //要与 BoardConfig.mk 文件中 TARGET_BOOTLOADER_BOARD_NAME 一致

- U-boot 配置项, 如图所示:



2.4 配置 chipram 部分

- 添加配置头文件

在 chipram/include/configs 添加 .h 文件, 文件名必须与编译配置中 sp7731c_test_lh10_oversea.mk 文件中 UBOOT_DEFCONFIG 的值一致, 同样建议拷贝添加:

```
cp sp7731cea.h sp7731cea_test.h
```

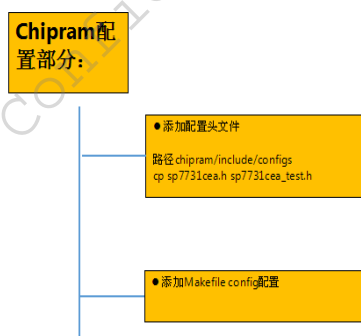
- 添加 Makefile config 配置, 根据参考项目 sp7733cea 的配置进行修改

```
@echo "CONFIG_SCX20_MUAK = y" >> $(obj)include/config.mk
@echo "CONFIG_SYS_TEXT_BASE = 0x50003000" >> $(obj)include/config.mk
#@echo "CONFIG_SECURE_BOOT = y" >> $(obj)include/config.mk

sp7731cea_test_config : preconfig
@$(MKCONFIG) $@ arm armv7 sp7731cea_test spreadtrum sc8830
@echo "CONFIG_EMMC_BOOT = y" >> $(obj)include/config.mk
@echo "CONFIG_SPL_32K = y" >> $(obj)include/config.mk
@echo "CONFIG_SPRD_DMC_R1P0 = y" >> $(obj)include/config.mk
@echo "CONFIG_SCX20_MDAR = y" >> $(obj)include/config.mk
@echo "CONFIG_SYS_TEXT_BASE = 0x50003000" >> $(obj)include/config.mk
#@echo "CONFIG_SECURE_BOOT = y" >> $(obj)include/config.mk

sp7731ceb_config : preconfig
@$(MKCONFIG) $@ arm armv7 sp7731ceb spreadtrum sc8830
@echo "CONFIG_EMMC_BOOT = y" >> $(obj)include/config.mk
```

- Chipram 配置项，如图所示：



2.5 完成配置准备编译

在完成了上面所有的操作后重新执行

```
source build/envsetup.sh
```

```
lunch
```

就可以选择新添加项目的产品进行编译了。

第3章 其它编译相关的内容

3.1 OTA 包的编译

1 版本的 ota 包生成方式

首先，我们需要完整的编译对应的版本代码，然后需要确认 cp 相关的 bin 文件已经拷贝到 device/sprd/spXXXX/modem_bins/目录下，根据不同的芯片，是不一样的。在 Dolphin 平台上，包括 dsp.bin, modem.bin, nvitem.bin, cpfdl.bin。如果采用 BT/WIFI/FM 三合一芯片的，还会有 wcnmodem.bin 和 wcnnvitem.bin，然后使用命令

```
make otapackage
```

来生成 ota 包，ota 包会在 out 目录下以 <产品名>-ota-<序列号>.zip 的命名以压缩文件的格式生成

2 ota 差分包的制作

在获得新旧两个版本的 ota 包之后，可以制作对应的升级差分包，命令为

```
./build/tools/releasetools/ota_from_target_files -i ota_old.zip ota_new.zip  
update.zip
```

ota_old.zip 与 ota_new.zip 分别是升级前和要升级到版本的 ota 包。

ota 包和升级包可以通过 sd 卡在 recovery 模式下进行 ota 升级。进入 recovery 模式的方法请参考《Android5.1 客户配置文档》。

3.2 如何制作多国语言版本

具体请参考《MocorDroid 多国语言概述(包含 Android6.0).pdf》文档。