

UDX710 CPE miniAP 编译指导

适用产品信息	UDX710
适用版本信息	Android 8.1
关键字	UDX710 CPE

Unisoc Confidential For hiar

声明

本文件所含数据和信息都属于紫光展锐所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。

版本历史

版本	日期	备注
V1.0	2019/11/12	初稿
V1.1	2019/11/18	修改格式，升级版本
V1.2	2020/06/22	增加 git 推送方式发布 idh 的信息描述

Unisoc Confidential For hiar

目录

1 前言.....	1
1.1 范围.....	1
1.2 缩略语	1
2 代码编译的方法	2
2.1 适用范围.....	2
2.2 代码准备.....	2
2.3 编译环境准备	3
2.4 完成一次全新的编译	4
2.5 单项编译和其它编译命名	7
2.6 单项编译命令	7
2.7 单项编译注意事项.....	8
2.8 编译的成果.....	8
3 其他编译相关的内容	11
3.1 OTA 包的编译	11
3.2 如何解决 Jack server 端口冲突导致编译失败的问题	11

1 前言

1.1 范围

文档说明

本文档简要介绍了紫光展锐 Android8.1 编译系统及其使用方法。

阅读对象

本文档针对于所有使用紫光展锐 Android8.1 产品的软件工程师。

1.2 缩略语

此章节请列出此文档所涉及的缩略词。

名称	全称	定义
EOL	End of Life	产品生命周期结束,包括停止制造,停止销售,停止服务三个活动。

2 代码编译的方法

2.1 适用范围

该文档适用紫光展锐 UDX710 芯片在 Android8.1 的编译和配置。

2.2 代码准备

首先，您需要解压完整的平台代码包。代码包由软件 CPM 发布，其中包含代码，bin 文件和开发调试工具等。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>-xxx.zip

解压 proprieties-<平台名>-xxx.zip 后，需要将产生的 out 目录移动到一个路径下
/home6/idh_orca/out 下。

 注意：

如若 IDH 包通过 Git 推送方式发布，推送的属性包、proprieties-xxx.zip 包在：
vendor/sprd/release/IDH/\$project 目录下。Git 推送编译参考文档在
vendor/sprd/release/IDH/Script 目录下

-
- 芯片配置文件 conf-<平台名>.tar.gz

解压 conf-<平台名>.tar.gz 包的内容到~/idh_orca/idh.code/device/sprd/路径下

然后在该项目的项目产品配置文件 ~/idh_orca/idh.code/device/sprd/orca/udx710_3h10

udx710_3h10_native.mk 中添加 SPRD_IDH_PROP:= /home6/idh_orca/out 即可。

```

KERNEL_PATH := kernel
export KERNEL_PATH
BOARD_PATH=$(KERNEL_PATH)/sprd-board-config/orca/udx710_3h10/udx710_3h10_native
include $(BOARD_PATH)
PLATDIR := device/sprd/orca
TARGET_BOARD := udx710_3h10
BOARDDIR := device/sprd/orca/$(TARGET_BOARD)
PLATCOMM := $(PLATDIR)/common
ROOTDIR := $(BOARDDIR)/rootdir
TARGET_BOARD_PLATFORM := udx710

SPRD_IDH_PROP := /home6/bo.zhou/idh_orca/out
include $(PLATCOMM)/ModemCommon.mk
#add for engpc/bbat

```

图 2.1 配置文件路径范例

👁 注意：

如若 IDH 包通过 git 推送方式发布，conf-xxx.tar.gz 代码已经开源，在代码 device 目录下

2.3 编译环境准备

您需要检查自己的编译环境，Google 推荐使用 64 位 Ubuntu 的系统。Ubuntu 10.04 - 12.04

版本都可以。紫光展锐推荐使用 14.04 的版本。使用下面命令来查看 Ubuntu 的具体版本号：

```
lsb_release -a
```

- 需要安装 1.8 版本的 Openjdk，使用下面命令来查看 Jdk 的版本：

```
java -version
```

用下面命令安装 openjdk 1.8:

```

sudo apt-get update

sudo apt-get install openjdk-8-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，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。

2.4 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，当然您也可以选择在完成自定义项目配置之后再开始编译。但是我们还是建议您在准备好代码之前先进行一次默认项目的编译，来保证环境的正确性。

通过 Ubuntu 终端命令行工具进入代码的根目录，

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：


```
including vendor/sprd/external/tools-build/vendorsetup.sh
including sdk/bash_completion/adb.bash
bo.zhou@cdand01:~/idh_orca/idh.code$ 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. full_fugu-userdebug
  8. aosp_fugu-userdebug
  9. aosp_car_emu_arm-userdebug
 10. aosp_car_emu_arm64-userdebug
 11. aosp_car_emu_x86-userdebug
 12. aosp_car_emu_x86_64-userdebug
 13. mini_emulator_arm64-userdebug
 14. m_e_arm-userdebug
 15. m_e_mips64-eng
 16. m_e_mips-userdebug
 17. mini_emulator_x86_64-userdebug
 18. mini_emulator_x86-userdebug
 19. uml-userdebug
 20. aosp_dragon-userdebug
 21. aosp_dragon-eng
 22. aosp_angler-userdebug
 23. aosp_bullhead-userdebug
 24. aosp_bullhead_svelte-userdebug
 25. hikey-userdebug
 26. hikey960-userdebug
 27. sp9650_haps_native-userdebug
 28. udx710_1h10_native-userdebug
 29. udx710_2h10_native-userdebug
 30. udx710_3h10_32b_native-userdebug
 31. udx710_3h10_native-userdebug
 32. udx710_haps_native-userdebug

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

图 2.2 lunch 目录范例

输入对应的数字选择需要编译的项目，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，例如使用对应的 user 版本：

```
32. udx710_haps_native-userdebug
Which would you like? [aosp_arm-eng] 31
device/sprd/orca/common/BoardCommon.mk:133: MALI_PLATFORM_NAME:orca
=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=8.1.0
TARGET_PRODUCT=udx710_3h10_native
TARGET_BUILD_VARIANT=userdebug
TARGET_BUILD_TYPE=release
TARGET_PLATFORM_VERSION=OPM1
TARGET_BUILD_APPS=
TARGET_ARCH=arm64
TARGET_ARCH_VARIANT=armv8-a
TARGET_CPU_VARIANT=generic
TARGET_2ND_ARCH=
TARGET_2ND_ARCH_VARIANT=armv7-a-neon
TARGET_2ND_CPU_VARIANT=cortex-a15
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-4.2.0-27-generic-x86_64-with-Ubuntu-14.04-trusty
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=OPM1.171019.006
OUT_DIR=out
AUX_OS_VARIANT_LIST=
=====
bo.zhou@cdand01:~/idh_orca/idh.code$
```

图 2.3 目标 board 内容详细信息

项目	数值	说明
PLATFORM_VERSION	8.1.0	Android 8.1
TARGET_PRODUCT	udx710_3h10_native	产品名，同产品 Make 文件
TARGET_BUILD_VARIANT	userdebug	Userdebug 版本
TARGET_ARCH	arm64	64 版本，如果是 go 版本显示的是 arm
TARGET_BUILD_VERSION	gms	编译 GMS 版本
OUT_DIR	out	编译输出目录

表 2.1 目标 board 字段解释

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

选择完编译项目后，先执行 kheader,完成安装 kernel 提供给用户态程序使用的头文件。

kheader

使用 make 指令编译整个工程

```
make
```


如果编译使用的 PC 是支持多线程编译的，可以使用-j 选项来加快编译的速度，比如-j 之后的数值表示多线程并行编译，主要在于 PC 是否支持多线程并行编译，主要跟 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。

```
make -j24
```

2.5 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

2.6 单项编译命令

项目	命令	生成文件
uboot	make bootloader	fdl2-sign.bin u-boot-sign.bin
fdl1,uboot-16k	make chipram	fdl1-sign.bin u-boot-spl-16k-sign.bin
kernel	make bootimage	boot.img kernel ramdisk.img(不需下载用)
system	make systemimage	system.img
userdata	make userdataimage	userdata.img
vendor	make vendorimage	vendor.img

表 2.2 目标 board 字段解释

2.7 单项编译注意事项

- 编译 Android 模块或本地库及程序

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

2.8 编译的成果

Android 的编译输出路径为 out，编译成果如下图

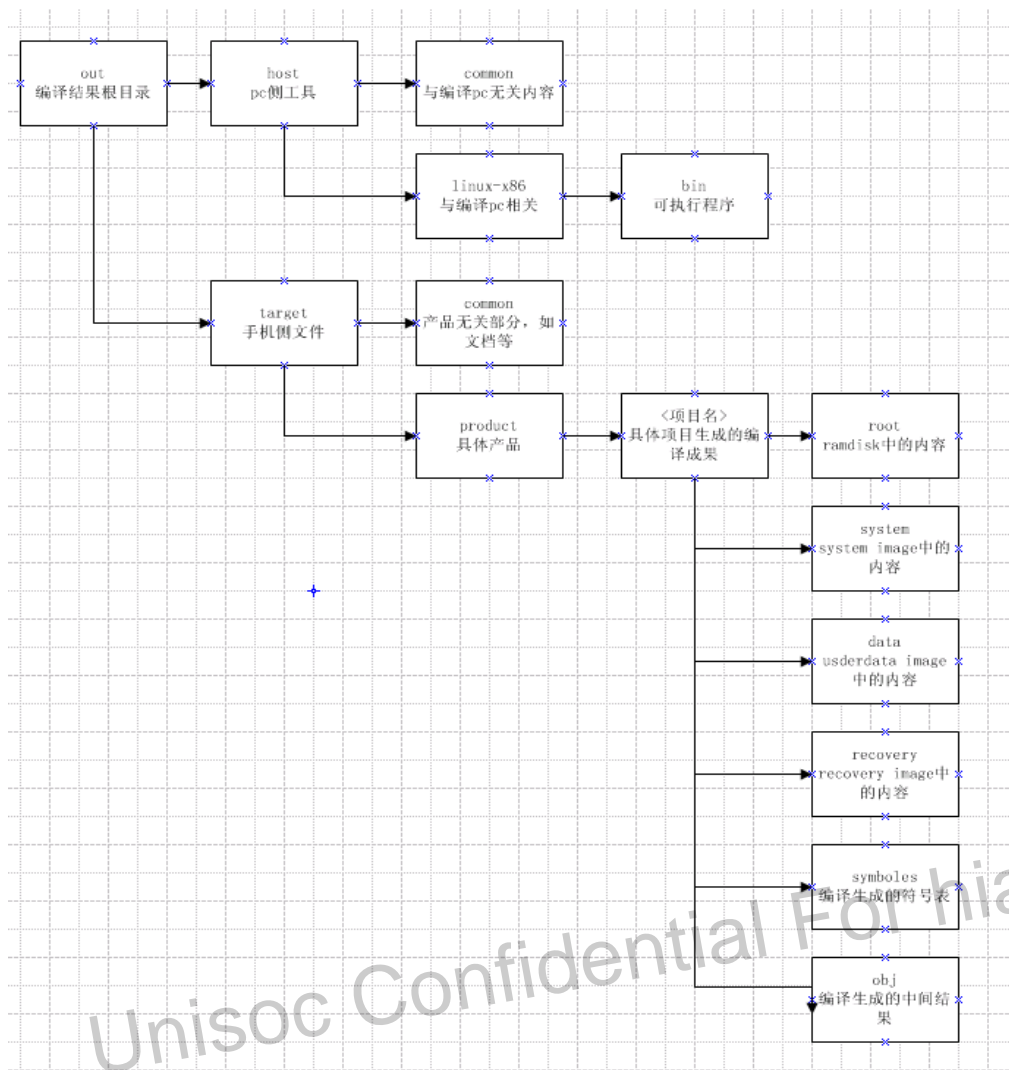


图 2.4 编译结果目录

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 fdl1-sign.bin fdl2-sign.bin u-boot-spl-16k-sign.bin u-boot-sign.bin boot.img system.img usderdata.img recovery.img cache.img vendor.img, dtb.img, dtbo.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

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——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 也可以在 out/target/product/<项目名>/obj/KERNEL 目录下找到。

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

Unisoc Confidential For hiar

3 其他编译相关的内容

3.1 OTA 包的编译

- 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 idh.code/device/sprd/orca/udx710_3h10/modem_bins/目录下，根据不同的芯片，是不一样的。在 sharkle 平台上，包括 ltemodem.bin，ltegdsp.bin 等等。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

- 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。

3.2 如何解决 Jack server 端口冲突导致编译失败的问题

Google 在 android7.0 以后版本启用了 jack-server (6.0 里也有这个东西，不过 6.0 里是可禁用的，6.0 默认是禁的，也可手动开启)，7.0 以后默认是开启的，并且无法禁用，且 Jack-server 有个 bug—端口是写死的；所以在单人 PC 或服务器上第 1 个人是没问题的，碰到服务器上多人使用时就会因为端口冲突导致失败。

目前我们解决这个问题方法是提供一个通用的脚本，在第一次编译 android9.0 时，先执行一下脚本配置好端口然后进行编译；

脚本具体执行如下：

- 首先在执行脚本之前需要将 .jack-server 放到 /usr/local/bin/Jack 目录下
- 然后将附件中的脚本也放到 /usr/local/bin/Jack 目录下
- 创建 port1 和 port2 文件，写入初始端口号；
- 执行 `sh /usr/local/bin/Jack/jack.sh`

jack.sh 内容如下：

```
#!/bin/bash
PS=`ps -ef | grep java | grep .jack-server | grep $HOME | wc -l`
if [ $PS = 0 ];then
cp -r /usr/local/bin/Jack/.jack-server ~/
cp /usr/local/bin/Jack/.jack-settings ~/
Port1=`cat /usr/local/bin/Jack/port1.txt`
Port2=`cat /usr/local/bin/Jack/port2.txt`
Port1=$((Port1+2))
Port2=$((Port2+2))
echo "$Port1" > /usr/local/bin/Jack/port1.txt
echo "$Port2" > /usr/local/bin/Jack/port2.txt
sed -i "s/8076/$Port1/g" ~/.jack-server/config.properties
sed -i "s/8077/$Port2/g" ~/.jack-server/config.properties
sed -i "s/8076/$Port1/g" ~/.jack-settings
sed -i "s/8077/$Port2/g" ~/.jack-settings
chmod 700 ~/.jack-server
chmod 600 ~/.jack-server/client.jks
chmod 600 ~/.jack-server/client.pem
chmod 600 ~/.jack-server/launcher.jar
chmod 600 ~/.jack-server/server-1.jar
chmod 600 ~/.jack-server/server.jks
chmod 600 ~/.jack-server/server.pem
chmod 600 ~/.jack-server/config.properties
echo "your jack-server port is $Port1 and $Port2"
fi
```