

Unisoc Confidential For hiar

# Android 9.0 IDH 包编译使用指南

文档版本                      V1.7  
发布日期                      2021-02-25

**版权所有 © 紫光展锐（上海）科技有限公司。保留一切权利。**

本文件所含数据和信息都属于紫光展锐（上海）科技有限公司（以下简称紫光展锐）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。

Unisoc Confidential For hiar

**紫光展锐（上海）科技有限公司**



# 前言

## 概述

本文档主要介绍 Android 9.0 IDH 包的内容、结构、以及如何进行编译并将编译输出下载到手机中。

本文档是基于 Linux njand02 3.19.0-25-generic #26~14.04.1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015 x86\_64 x86\_64 x86\_64 GNU/Linux 的编译环境撰写，故在其他版本的 host 环境上可能存在差异。

## 读者对象




本文档主要适用于 Android 9.0 平台开发的开发人员。

## 缩略语

| 缩略语 | 英文全名                           | 中文解释                |
|-----|--------------------------------|---------------------|
| IDH | Independent Design House       | 独立方案设计公司            |
| PAC | Package                        | 打包 Android 镜像一种文件格式 |
| OTA | Over-the-Air Technology        | 空中下载技术              |
| WCN | Wireless Communication Network | 无线通信网络              |

## 符号约定

在本文中可能出现下列标志，它所代表的含义如下。

| 符号  | 说明  |
|---|---|
|  <b>说明</b> | 用于突出重要/关键信息、补充信息和小窍门等。<br>“说明”不是安全警示信息，不涉及人身、设备及环境伤害。 |
|  <b>注意</b> | 用于突出容易出错的操作。<br>“注意”不是安全警示信息，不涉及人身、设备及环境伤害。           |
|  <b>警告</b> | 用于可能无法恢复的失误操作。<br>“警告”不是危险警示信息，不涉及人身及环境伤害。            |

## 变更信息

| 文档版本 | 发布日期       | 修改说明   |
|------|------------|--|
| V1.0 | 2019-03-01 | 初稿。  |
| V1.1 | 2019-03-27 | 新增常见问题，脚本打包 pac 等内容。   |
| V1.2 | 2019-05-15 | board 相关开源更新。  |
| V1.3 | 2019-06-18 | 第四章增加脚本位置说明。   |
| V1.4 | 2019-06-21 | 增加 4.2 创建流程。   |
| V1.5 | 2019-07-26 | 新增 UDS710/UMS312 modem bin 相关映射关系。   |
| V1.6 | 2019-10-10 | 新增 UDS710 NPU 相关差异说明。  |
| V1.7 | 2021-02-25 | <ul style="list-style-type: none"><li>新增关于 git 推送版本的脚本打包说明。</li><li>更新文档格式，优化文档内容。</li></ul> |

## 关键字

IDH、编译、OTA、pac。

Unisoc Confidential For hiar

# 目 录

|                                   |    |
|-----------------------------------|----|
| 1 概述.....                         | 1  |
| 1.1 IDH 包组件 .....                 | 1  |
| 1.2 IDH 包解压 .....                 | 2  |
| 1.2.1 非 Git 推送版本 .....            | 2  |
| 1.2.2 Git 推送版本 .....              | 4  |
| 1.3 解压 modem 部分 .....             | 4  |
| 1.4 解压 NPU 部分(仅针对 UDS710 工程)..... | 4  |
| 2 编译环境准备.....                     | 6  |
| 2.1 硬件环境要求 .....                  | 6  |
| 2.2 软件环境要求 .....                  | 6  |
| 2.2.1 操作系统要求 .....                | 6  |
| 2.2.2 工具包安装 .....                 | 6  |
| 2.2.3 python 安装 .....             | 6  |
| 2.3 常见问题处理 .....                  | 7  |
| 2.3.1 系统库缺失 .....                 | 7  |
| 2.3.2 ubuntu 版本低 .....            | 7  |
| 2.3.3 Java 版本错误 .....             | 7  |
| 3 编译操作指导.....                     | 8  |
| 3.1 编译步骤 .....                    | 8  |
| 3.2 全编译 .....                     | 8  |
| 3.2.1 初始化环境变量及命令 .....            | 8  |
| 3.2.2 新增及选择工程 .....               | 8  |
| 3.2.3 执行编译及输出 out 目录 .....        | 9  |
| 3.3 单独编译 .....                    | 11 |
| 3.3.1 伪目标镜像 .....                 | 11 |
| 3.3.2 特定模块 .....                  | 11 |
| 3.4 版本编译说明 .....                  | 12 |
| 3.4.1 Userdebug 版本 .....          | 12 |
| 3.4.2 User 版本 .....               | 12 |
| 3.4.3 GMS 版本 .....                | 13 |
| 3.4.4 运营商版本 .....                 | 13 |
| 3.5 编译结果验证 .....                  | 14 |
| 3.5.1 adb push 验证 .....           | 14 |
| 3.5.2 镜像烧录验证 .....                | 14 |
| 3.6 其它编译说明 .....                  | 15 |

|                            |    |
|----------------------------|----|
| 3.6.1 非开源 apk 重新签名说明 ..... | 15 |
| 3.6.2 编译报错说明 .....         | 16 |
| 3.6.3 编译选项说明 .....         | 16 |
| 3.6.4 库的依赖关系说明 .....       | 16 |
| 4 新建 board.....            | 17 |
| 4.1 创建步骤 .....             | 17 |
| 4.2 创建流程 .....             | 17 |
| 5 OTA 整体及差分升级包制作 .....     | 22 |
| 5.1 OTA 整体升级包制作 .....      | 22 |
| 5.2 OTA 差分升级包制作 .....      | 27 |
| 6 pac 制作及下载 .....          | 28 |
| 6.1 pac 制作 .....           | 28 |
| 6.1.1 工具制作 .....           | 28 |
| 6.1.2 脚本制作 .....           | 29 |
| 6.2 pac 下载 .....           | 30 |

Unisoc Confidential For hiar

# 图目录

|  |    |
|--|----|
| 图 1-1 IDH 包组成 .....  | 1  |
| 图 1-2 UDS710 NPU 部分说明 .....                                | 2  |
| 图 1-3 IDH 包结构 .....  | 2  |
| 图 1-4 开源源码解压 .....   | 3  |
| 图 1-5 非开源部分解压 .....  | 4  |
| 图 1-6 NPU 部分解压 .....                                       | 5  |
| 图 3-1 初始化环境变量及命令 .....                                     | 8  |
| 图 3-2 编译工程列表 .....   | 9  |
| 图 3-3 编译参数 .....   | 9  |
| 图 3-4 编译输出目录说明 .....                                       | 10 |
| 图 3-5 项目名目录 .....  | 10 |
| 图 3-6 编译类型选择 .....   | 12 |
| 图 3-7 编译工程选择 .....   | 12 |
| 图 3-8 编译版本及编译变量选择 .....                                    | 13 |
| 图 3-9 运营商编译说明 .....  | 14 |
| 图 3-10 烧录 system 镜像 .....                                  | 15 |
| 图 3-11 预编译方式自动签名 .....                                     | 16 |
| 图 4-1 修改 kernel4.4/arch/arm64/boot/dts/sprd/Makefile ..... | 19 |
| 图 4-2 修改 u-boot 相关配置 .....                                 | 20 |
| 图 4-3 修改 board.def 配置 .....                                | 21 |
| 图 6-1 工具制作 pac .....                                       | 28 |
| 图 6-2 修改 size 大小 .....                                     | 29 |
| 图 6-3 脚本内容结构 .....   | 30 |
| 图 6-4 脚本使用说明 .....   | 30 |
| 图 6-5 ResearchDownload 下载工具使用说明 .....                      | 31 |

## 表目录

|                               |    |
|-------------------------------|----|
| 表 3-1 伪目标镜像单独编译说明 .....       | 11 |
| 表 3-2 bootloader 模式说明 .....   | 15 |
| 表 5-1 UDS710 改名映射 .....       | 22 |
| 表 5-2 SC7731/7731C 改名映射 ..... | 23 |
| 表 5-3 SC9832A 改名映射 .....      | 23 |
| 表 5-4 SC9820A 改名映射 .....      | 23 |
| 表 5-5 SC9850K 改名映射 .....      | 24 |
| 表 5-6 SC9853I 改名映射 .....      | 24 |
| 表 5-7 SC9820E 改名映射 .....      | 24 |
| 表 5-8 SC9832E 改名映射 .....      | 25 |
| 表 5-9 SC9863A 改名映射 .....      | 25 |
| 表 5-10 SC9850E 改名映射 .....     | 26 |
| 表 5-11 SC7731E 改名映射 .....     | 26 |
| 表 5-12 UMS312 改名映射 .....      | 26 |



# 1 概述

## 1.1 IDH 包组件

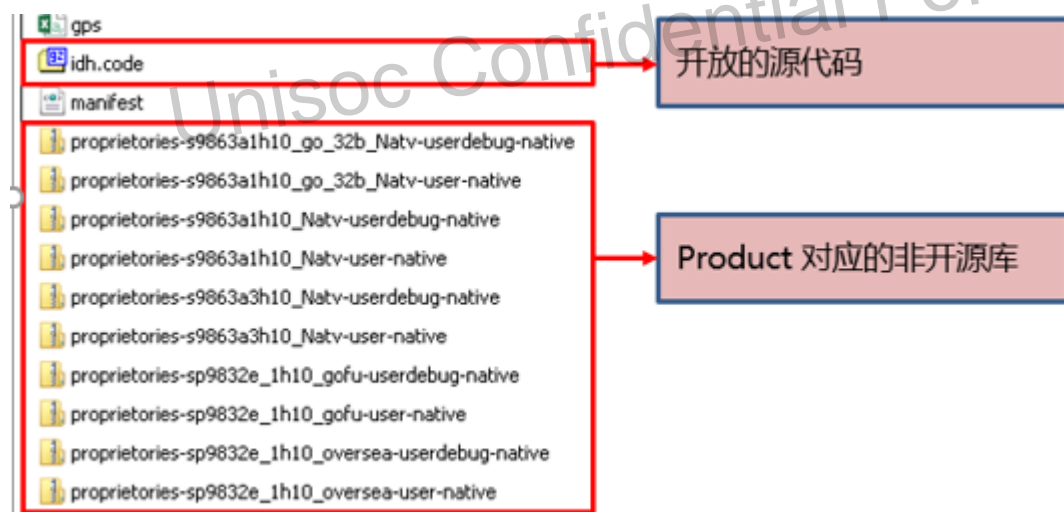
目前 UNISOC 大多采用 git 推送版本，部分使用 FTP 推送的方式提供 IDH 包。两种发布方式在源代码处理上略有差异。

Android 9.0 UNISOC IDH 包由以下几个部分组成：

- 开源源码部分  
如：idh.code.tgz
- 非开源部分  
如：proprieties-sp9832e\_1h10\_oversea-userdebug-native.zip，其中 sp9832e\_1h10\_Natv 为 product name，userdebug 为 build type。

具体可参考图 1-1。

图1-1 IDH 包组成



另外 UDS710 的 IDH 会包含如下两个压缩包，如下图 1-2 所示：

图1-2 UDS710 NPU 部分说明



## 说明

NPU 部分的代码是单独拉出打包，编译 UDS710 工程时需要解压到 IDH 源码根目录。

## 1.2 IDH 包解压

### 1.2.1 非 Git 推送版本

以 sp9832e oversea 工程为例，获取的 IDH 包结构如[图 1-3](#) 所示。

图1-3 IDH 包结构

```

├─ idh.code.tgz
├─ proprietaries-sp9832e_1h10_oversea-userdebug-native.zip
├─ proprietaries-sp9832e_1h10_oversea-user-native.zip

```

#### 1.2.1.1 开源源码解压

将 idh.code.tgz 解压到特定目录，如 D:\AndroidP\，如[图 1-4](#) 所示。

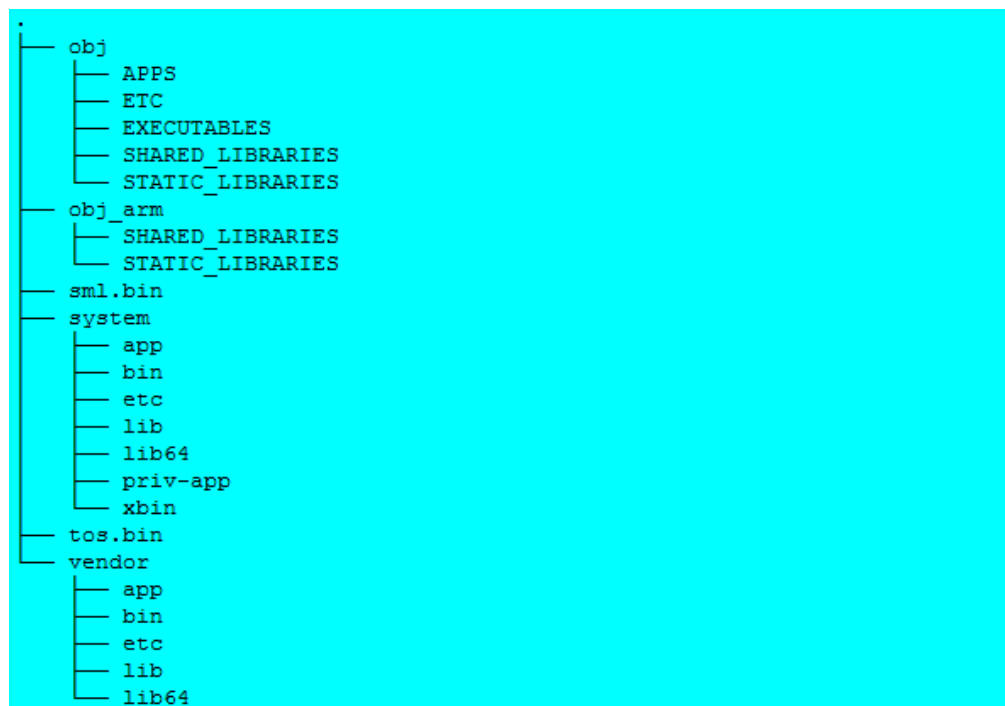
图1-4 开源源码解压

```
.
├── Android.bp -> build/soong/root.bp
├── art
├── bionic
├── bootable
├── bootstrap.bash -> build/soong/bootstrap.bash
├── build
├── chipram
├── compatibility
├── cts
├── dalvik
├── developers
├── development
├── device
├── external
├── frameworks
├── hardware
├── kernel
├── kernel4.4
├── libcore
├── libnativehelper
├── Makefile
├── out
├── packages
├── pdk
├── prebuilts
├── sdk
├── system
├── test
├── toolchain
├── tools
├── u-boot15
├── u-boot64
└── vendor
```

### 1.2.1.2 非开源部分解压

将 `proprieties-sp9832e_1h10_oversea-userdebug-native.zip` 解压之后的 `out` 文件夹放到上述目录 `/AndroidP/idh.code/out`，如图 1-5 所示。

图1-5 非开源部分解压



直接解压到开源代码目录下。

## 1.2.2 Git 推送版本

Git 推送的版本都是以源码的方式推送下载，不存在对应的压缩包。只需关注非开源代码的存放路径。

非开源代码的存放路径：vendor/sprd/release/IDH/project\_XXX（编译工程名称）。

## 1.3 解压 modem 部分

在/AndroidP/idh.code/下创建目录/pac/RELEASE\_PAC\_CP/，该目录用于存放解压后的 Modem、cp2、gnss 文件（打 pac 时需要这些文件）。

## 1.4 解压 NPU 部分(仅针对 UDS710 工程)

NPU 部分代码需要单独打包，编译 UDS710 工程时需要解压到 IDH 源码根目录 AndroidP/idh.code 目录下。NPU 部分解压后目录结构如图 1-6 所示：

图1-6 NPU 部分解压



Unisoc Confidential For hiar

# 2

## 编译环境准备

### 2.1 硬件环境要求

虚拟机编译环境硬件要求：

- 可用硬盘容量至少 250G。
- 内存至少 16G。否则编译大概率会出现 `Exception in thread "main" java.lang.OutOfMemoryError: Java heap space`。

### 2.2 软件环境要求

#### 2.2.1 操作系统要求

建议选择 14.04 版本的 64 位 ubuntu 系统，查看 ubuntu 的具体版本号命令为：

```
lsb_release -a
```

额外需要的软件包主要有：

- python.org 中提供的 Python 2.6 - 2.7。
- gnu.org 中提供的 GNU Make 3.81 - 3.82。
- git-scm.com 中提供的 Git 1.7 或更高版本。

#### 📖 说明

ubuntu10.04~12.0 版本或者更高版本 ubuntu16.04 也支持，但是不同版本依赖的编译支持工具略有差异。

#### 2.2.2 工具包安装

选择 ubuntu 14.04 系统后，可使用如下命令安装编译依赖环境的工具包：

```
$ 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 libssl-dev
```

#### 📖 说明

安装的 ubuntu 版本的不同，需要的编译支持工具包不同，完整的工具包可在：  
<https://source.android.com/setup/build/initializing> 获取。

#### 2.2.3 python 安装

在 IDH 包开源部分的/prebuilts/ python/下有预编译版本的 python，因此无需另行安装。

如果系统已安装其他版本的 python，建议删除，并安装 2.7 或 2.6 版本；查看 python 版本的命令为：

```
python --version
```

### 📖 说明

可以在 python 官网下载具体版本：<https://www.python.org/>。

## 2.3 常见问题处理

### 2.3.1 系统库缺失

如：fatal error: openssl/bio.h 等

安装 openssl

```
sudo apt-get install openssl
```

当#include<openssl/ssl.h>后编译报错，如果再出现 xxx not found，解决办法为：

```
sudo apt-get install libssl-dev build-essential zlibzlib-bin libidn11-dev libidn11
```

### 2.3.2 ubuntu 版本低

Google 建议使用 14.04 版本的 ubuntu 系统，当使用低于 14.04 版本的 ubuntu 系统时会报错“GLIBC\_2.17/2.18 not found”。

如果必须使用该版本的 ubuntu 系统，则需要安装对应的依赖库，具体可参考如下命令：

```
sudo dpkg -i libc6_2.17-0ubuntu4_amd64.deb
```

### 2.3.3 Java 版本错误

如果编译的 Android 版本与 Java 版本不一致，make 将会终止并显示诸如以下消息：

```
*****
You are attempting to build with the incorrect version of java.
Your version is: WRONG_VERSION.
The correct version is: RIGHT_VERSION.
Please follow the machine setup instructions at
  https://source.android.com/source/initializing.html
*****
```

原因是未能安装指定的 JDK。此时需确保已经设置环境变量，将正确的 JDK 附加到路径开头，或者移除有问题的 JDK。

# 3 编译操作指导

## 3.1 编译步骤

无论是单独编译还是全编译，都需要完成以下几步：

步骤 1 source build/envsetup.sh，初始化环境变量及命令。

步骤 2 lunch，选择具体的编译工程，指定编译参数。

步骤 3 make XXX，编译特定目标。

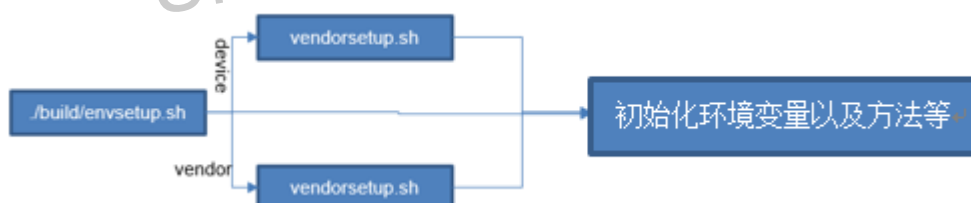
---结束

## 3.2 全编译

### 3.2.1 初始化环境变量及命令

初始化环境变量及命令，搜集编译工程等，如图 3-1 所示。

图3-1 初始化环境变量及命令



### 3.2.2 新增及选择工程

选择可以编译的工程，如图 3-2 所示。选择对应的序号即可编译对应的版本。如 30，编译带 GMS 包的 userdebug 版本。



图3-2 编译工程列表

```
61. sp9832e_1h10_native-userdebug-native
62. sp9832e_1h10_native-userdebug-gms
63. sp9832e_1h10_oversea-userdebug-native
64. sp9832e_1h10_oversea-userdebug-gms
65. sp9832e_1h10_nosec-userdebug-native
66. sp9832e_1h10_nosec-userdebug-gms

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

lunch 选择好编译的工程之后，会输出当前编译工程的一些参数，如图 3-3 所示。

图3-3 编译参数

```
=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=9
TARGET_PRODUCT=sp9832e_1h10_native
TARGET_BUILD_VARIANT=user
TARGET_BUILD_VERSION=native
TARGET_BUILD_TYPE=release
TARGET_ARCH=arm64
TARGET_ARCH_VARIANT=armv8-a
TARGET_CPU_VARIANT=generic
TARGET_2ND_ARCH=arm
TARGET_2ND_ARCH_VARIANT=armv7-a-neon
TARGET_2ND_CPU_VARIANT=cortex-a15
KERNEL_DEFCONFIG=sprd_sharkle_defconfig
UBOOT_DEFCONFIG=sp9832e_1h10
CHIPRAM_DEFCONFIG=sp9832e_1h10
TARGET_DTB=sp9832e-1h10-native
TARGET_DTBO=sp9832e-1h10-overlay
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.19.0-25-generic-x86_64-Ubuntu-14.04.3-LTS
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=PPR1.180610.011
OUT_DIR=out
=====
```

Android 平台大版本号  
编译目标工程  
硬件信息  
cpu信息  
Host对应的硬件信息  
Out目录

#### 注意

Android 9.0 编译 GMS 版本，需要将 GMS 包相关的应用放到指定目录：vendor/partner\_gms 下。

### 3.2.3 执行编译及输出 out 目录

通常使用 make 命令来进行全编译。一次全新的编译根据编译服务器的性能不同需要几十分钟到几个小时不等。

如果编译使用的机器支持多线程编译，则可以使用 -j 参数加快编译速度，-j 之后的数值表示并行编译的多线程数，建议数值设置为编译服务器核数的两倍。比如：

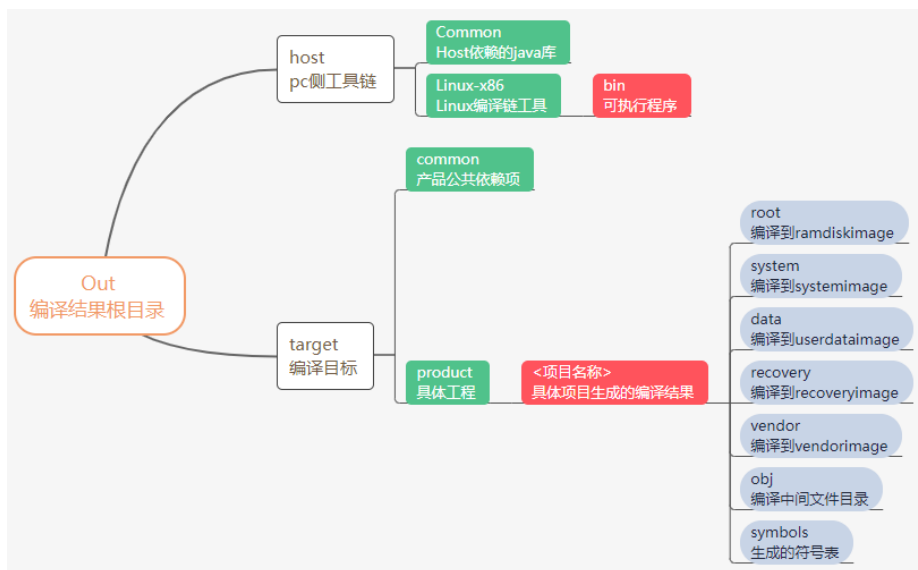
```
make -j8
```

#### 说明

-j 不建议使用较大的数值，防止由于 Makefile 依赖书写不规范导致的编译错误，线程数越大，出现错误的概率越大。

Android 的编译输出路径为 out，其目录结构如图 3-4 所示。

图3-4 编译输出目录说明



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` 等，如图 3-5 所示。

图3-5 项目名目录

```
chunlei.liu@NJand01:~/AndroidP/idh.code/out/target/product/sp9832e_1h10$
├── ./boot.img
├── ./cache.img
├── ./dtb.img
├── ./dtbo.img
├── ./persist.img
├── ./prodnv.img
├── ./product.img
├── ./ramdisk.img
├── ./ramdisk-recovery.img
├── ./recovery.img
├── ./system.img
├── ./userdata.img
├── ./vbmeta-gsi.img
├── ./vbmeta-sign.img
└── ./vendor.img

chunlei.liu@NJand01:~/AndroidP/idh.code/out/target/product/sp9832e_1h10$
├── ./ddr_scan-sign.bin
├── ./fdl1-sign.bin
├── ./fdl2-sign.bin
├── ./sml-sign.bin
├── ./tos-sign.bin
├── ./u-boot_autopoweron-sign.bin
├── ./u-boot-sign.bin
└── ./u-boot-spl-16k-sign.bin
```

#### 说明

- 并非所有下载文件都是编译生成的，如 `cp` 侧的 `bin` 就是在版本发布中直接提供。

- out/target/product/<项目名>/目录下的 root、system、data、recovery 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。
- 编译的符号表在很多调试和 bug 解决中是非常重要的，可以在 out/target/product/<项目名>/symbols 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。

## 3.3 单独编译

在完成一次全编译之后，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。

### 3.3.1 伪目标镜像

伪目标镜像单独编译项、编译命令及其编译结果参见表 3-1。

表3-1 伪目标镜像单独编译说明

| 单独编译项            | 编译命令                 | 编译结果   |
|------------------|----------------------|--|
| u-boot           | make bootloader      | 生成 fdl2-sign.bin、u-boot-sign.bin、u-boot_autopoweron-sign.bin。    |
| fdl1 和 uboot-16k | make chipram         | 生成 fdl1-sign.bin u-boot-spl-16k-sign.bin。                        |
| boot image       | make bootimage       | 生成 boot.img dt.img kernel ramdisk.img。                           |
| boot-debug Image | make bootimage_debug | 生成 boot-debug.img，用于 XTS 测试。使用 user 版本搭配 boot-debug，可进行 root 操作。 |
| system image     | make systemimage     | 生成 system.img，需要使用 vbmeta_system.img 进行校验。                       |
| vendorimage      | make vendorimage     | 生成 vendor.img，需要使用 vbmeta_vendor.img 进行校验。                       |
| product image    | make productimage    | 生成 product.img。  |
| userdata image   | make userdataimage   | 生成 userdata.img。   |

### 3.3.2 特定模块

如果单独编译 Settings 应用，可使用如下集中方式编译：

先在根目录下执行以下命令：

```
make Settings -j8
```

然后在根目录下再执行以下命令：

```
mmm packages/apps/Settings/
```

工程整编过或通过 make 命令编译后，可在 Settings 的 Android.mk 所在目录执行 mm 或 mma 命令来编译。

## 说明

- m 在源码树的根目录执行 make，相当于全编译。
- mm 编译当前目录下的模块，不会编译依赖。
- mmm 编译指定目录下的模块，不会编译依赖。
- mma 编译当前目录下的模块，会编译相关依赖。
- mmma 编译指定目录下的模块，会编译相关依赖。

## 3.4 版本编译说明

### 3.4.1 Userdebug 版本

Userdebug 版本编译步骤参见 3.2 全编译。

### 3.4.2 User 版本

一般 lunch 不会列出编译 user 版本的选项，可以输入完整的工程名将其中的 userdebug 替换为 user，如：sp9832e\_1h10\_native\_user-native。

也可以使用 choosecombo 命令来选择具体工程：

步骤 1 选择 Build Type，如图 3-6 所示。

图3-6 编译类型选择

```
chunlei.liu@NJand01:~/AndroidP/idh.code$ choosecombo
Build type choices are:
  1. release
  2. debug

Which would you like? [1] 1
```

步骤 2 输入 Product name，可参考 lunch 的列表输入特定的工程名称，如图 3-7 所示。

图3-7 编译工程选择

```
Which product would you like? [aosp_arm] sp9832e_1h10_oversea
```

步骤 3 选择 Build variant && Build version，如图 3-8 所示。

图3-8 编译版本及编译变量选择

```
Variant choices are:
  1. user
  2. userdebug
  3. eng
Which would you like? [eng] 1

Version choices are:
  1. native
  2. gms
Which would you like? [native] 1
```

---结束

### 3.4.3 GMS 版本

Android 9.0 编译 GMS 版本时，需要将 GMS 包相关的应用放到指定目录：vendor/partner\_gms 仓库下。

#### 说明

- 由于 Google GMS 维护策略的变化，具体使用如下命令获取 GMS 包，UNISOC 释放的 IDH 不再包含 GMS 包。
- git clone [https://partner-android.googlesource.com/platform/vendor/partner\\_gms](https://partner-android.googlesource.com/platform/vendor/partner_gms) -b qt-gms-dev

### 3.4.4 运营商版本

部分运营商没有对应的工程，故需要使用如下方式编译，如有对应的运营商工程（CTCC），编译对应的工程即可。

- 单个运营商的编译：

```
make --product_carrier telcel -j8
```

- 多个运营商的编译：

```
make --product_carrier telcel;reliance -j8
```

编译完成后可在 out 目录对应的工程下找到对应的运营商目录，针对运营商的编译最终被打包到 product image 中。编译生成文件如图 3-9 所示。

图3-9 运营商编译说明

```
chunlei.liu@NJand01:~/AndroidP/idh.code/out/target/product/sp9832e_1h10/telcel$  
.  
├── product  
│   ├── app  
│   │   ├── TouchPalGlobal  
│   │   └── TouchPalGlobal.apk  
│   ├── build.prop  
│   ├── overlay  
│   │   ├── CarrierConfig_telcel_rro.apk  
│   │   ├── CellBroadcastReceiver_telcel_rro.apk  
│   │   ├── framework-res_telcel_rro.apk  
│   │   ├── Messaging_telcel_rro.apk  
│   │   ├── Settings_telcel_rro.apk  
│   │   ├── SprdDialer_telcel_rro.apk  
│   │   ├── stk_telcel_rro.apk  
│   │   ├── SystemUI_telcel_rro.apk  
│   │   ├── Telecom_telcel_rro.apk  
│   │   └── TeleService_telcel_rro.apk  
└── product.img
```

## 3.5 编译结果验证

当完成特定目标的编译后，验证方式有 adb push 验证和镜像烧录验证两种。

### 3.5.1 adb push 验证

在电脑上安装 adb 后，可以使用 adb push 命令将目标 push 到手机对应目录。待手机启动完成后即可验证对应功能。

如验证 settings 应用：

```
adb root  
adb remount  
adb push xxx/Settings.apk product/priv-app/Settings  
adb push xxx/oat/arm64/arm/Settings.vdex product /priv-app/Settings/oat/arm64/arm  
adb push xxx/oat/arm64/arm/Settings.odex product /priv-app/Settings/oat/arm64/arm  
adb reboot
```

### 3.5.2 镜像烧录验证

使用 ResearchDownload 工具加载对应的 pac 包，替换编译生成的 img 文件，烧录、重启、验证。

如果烧录 super 镜像，同时需要替换 vbmeta\_system 镜像跟 vbmeta\_vendor 镜像。

#### 说明

ResearchDownload 工具获取地址：Modem release 的 Tools 压缩包中  
\\Tools\\DEBUG\_TOOL\\ResearchDownload。

镜像烧录验证需在 bootloader 模式下进行，bootloader 模式说明见[表 3-2](#)。

表3-2 bootloader 模式说明

| 模式            | 作用         |
|---------------|------------|
| Bootloader 模式 | 用于管理和烧录分区。 |

bootloader 模式下镜像下载步骤:

步骤 1 进入 bootloader 模式, 命令格式如下:

```
adb reboot bootloader
```

步骤 2 烧录分区 (注意 selinux 权限问题, 必须先解锁 bootloader), 命令格式如下:

```
fastboot flash <partition name> <filename>
```

步骤 3 烧录 system 镜像, 如[图 3-10](#) 所示。

图3-10 烧录 system 镜像

```
C:\Users\chunlei.liu>fastboot flash system C:\Users\chunlei.liu\Downloads\signed_signed-aosp_arm64-img-5649316\system.img
Invalid sparse file format at header magic
Resizing 'system'                                OKAY [ 0.034s]
Sending sparse 'system' 1/3 <523608 KB>          OKAY [ 24.448s]
Writing 'system'                                  OKAY [ 13.458s]
Sending sparse 'system' 2/3 <523632 KB>          OKAY [ 27.135s]
Writing 'system'                                  OKAY [ 12.974s]
Sending sparse 'system' 3/3 <130296 KB>           OKAY [ 6.239s]
Writing 'system'                                  OKAY [ 5.680s]
Finished. Total time: 102.395s
```

---结束

## 3.6 其它编译说明

### 3.6.1 非开源 apk 重新签名说明

如果需要更换签名文件, 则非开源库应用等需要重新签名。平台支持的方式有如下两种。

- 手动签名替换

```
java -Xmx2048m -Djava.library.path="out/host/linux-x86/lib64" -jar out/host/linux-x86/framework/signapk.jar --min-
sdk-version 23 build/target/product/security/release/platform.x509.pem
build/target/product/security/release/platform.pk8 ~/源.apk ~/签名后的.apk
```

- 预编译方式自动签名

```
vendor/sprd/proprieties-source/proprieties-prebuilt/Android.mk
```

新增需要预编译签名的 module 配置。如[图 3-11](#) 所示。

图3-11 预编译方式自动签名

```
#####
#customized properties
#####
#prebuild for USCPHOTOSProvider
include $(CLEAR_VARS)
LOCAL_MODULE := USCPHOTOSProvider
LOCAL_MODULE_TAGS := optional
LOCAL_MODULE_CLASS := APPS
LOCAL_CERTIFICATE := media
LOCAL_DEX_PREOPT := false
LOCAL_PRIVILEGED_MODULE := true
MODULE_DIR := ../platform/packages/providers/USCPHOTOSProvider/prebuilt
$(call proprietary-prebuild,$(MODULE_DIR))

MY_TARGET_MODULE :=
MY_MODULE_SOURCE_DIR :=
```

## 3.6.2 编译报错说明

正常编译时，如果编译报错看到如 No command 'mmm' found 等信息时，需要先确认是否执行 source build/envsetup.sh 脚本文件设置编译环境。

## 3.6.3 编译选项说明

平台默认开启 LOCAL\_DEX\_PREOPT。

开启 LOCAL\_DEX\_PREOPT，打包 package 时会生成 apk 跟 dex 文件，验证时需要同时 push。

## 3.6.4 库的依赖关系说明

使用如下命令查看对应的 so 文件，然后将编译环境下的这些 so 拷贝到/lib 目录下。

```
$ readelf -d helloworld | grep NEEDED
0x00000001 (NEEDED)             Shared library: [libstdc++.so.6]
0x00000001 (NEEDED)             Shared library: [libm.so.6]
0x00000001 (NEEDED)             Shared library: [libgcc_s.so.1]
0x00000001 (NEEDED)             Shared library: [libc.so.6]
```



# 4 新建 board

## 4.1 创建步骤

为方便创建一个新 board，建议选择一个已有 board 作参考，使用 `cust_board_create.sh` 脚本自动生成一个新 board。例如在 `device/sprd/sharkle` 下以 `sp9832e_1h10` 为参考 board 新建一个 board `sp9832e_1h10_cust`。

新 board 创建步骤如下：

- 步骤 1 将 `vendor/sprd/tools/board_configure_tool/cust_board_create.sh` 脚本文件拷贝到源码根目录（即 `/AndroidP/idh.code/`）。
- 步骤 2 执行 `chmod a+x cust_board_create.sh` 命令添加可执行权限。
- 步骤 3 在源码根目录下执行 `./cust_board_create.sh sharkle sp9832e_1h10 sp9832e_1h10_cust` 自动新建一个 board。

### 说明

- `sharkle` 为参考芯片名称，可使用 `ls device/sprd` 查看所有参考芯片。
- `sp9832e_1h10` 为参考 board 名称，以 `sharkle` 参考芯片为例，可通过 `ls device/sprd/sharkle` 查看该芯片的所有参考 board。
- `sp9832e_1h10_cust` 为目标 board 名称，可任意修改。

----结束

## 4.2 创建流程

新 board 创建流程包含以下子流程：

- 清理关联仓库
- 创建 device board
- 创建 kernel board
- 创建 uboot board
- 创建 chipram board
- 创建 wcn board

下面以 `sp9832e_1h10` 为参考 board，新建一个名为 `sp9832e_1h10_cust` 的 board 为例说明使用脚本自动创建新 board 的各子流程。

### 清理关联仓库

使用以下脚本清理关联仓库，避免修改冲突。

```
cd device/sprd/sharkle && git clean -df && git reset --hard HEAD && cd ->/dev/null
cd chipram && git clean -df && git reset --hard HEAD && cd ->/dev/null
cd u-boot15 && git clean -df && git reset --hard HEAD && cd ->/dev/null
cd kernel4.4 && git clean -df && git reset --hard HEAD && cd ->/dev/null
cd vendor/sprd/modules/wcn/connconfig && git clean -df && git reset --hard HEAD && cd ->/dev/null
```

## 说明

如确定关联仓库未做修改或未配置为 git 仓库，可跳过清理关联仓库子流程。

## 创建 device board

1. 使用以下脚本将 device/sprd/sharkle/sp9832e\_1h10 拷贝至 device/sprd/sharkle/sp9832e\_1h10\_cust。

```
cp -fr device/sprd/sharkle/sp9832e_1h10 device/sprd/sharkle/sp9832e_1h10_cust
```

2. 使用以下脚本将 device/sprd/sharkle/sp9832e\_1h10\_cust 下的所有名称包含 sp9832e\_1h10 的子目录及文件重命名。

```
cd device/sprd/sharkle/sp9832e_1h10_cust
find . -type f -exec sed -i "s/sp9832e_1h10/sp9832e_1h10_cust/g;\
s/(TARGET_BOOTLOADER_BOARD_NAME := \).*/\1sp9832e_1h10_cust/g;\
s/(UBOOT_DEFCONFIG := \).*/\1sp9832e_1h10_cust/g;\
s/(UBOOT_TARGET_DTB := \).*/\1sp9832e_1h10_cust/g;\
s/(CHIPRAM_DEFCONFIG := \).*/\1sp9832e_1h10_cust/g" {} +
for file in `find . -type f -iname "*sp9832e_1h10*"`; do
mv $file ${file/sp9832e_1h10/sp9832e_1h10_cust}
done
cd ->/dev/null
```

## 创建 kernel board

1. 使用以下脚本将 kernel4.4/sprd-board-config/sharkle/sp9832e\_1h10 拷贝 kernel4.4/sprd-board-config/sharkle/sp9832e\_1h10\_cust。

```
cp -fr kernel4.4/sprd-board-config/sharkle/sp9832e_1h10 kernel4.4/sprd-board-config/sharkle/sp9832e_1h10_cust
```

2. 使用以下脚本将 kernel4.4/sprd-diffconfig/sharkle/arm64/sp9832e\_1h10\_diff\_config 拷贝至 kernel4.4/sprd-diffconfig/sharkle/arm64/sp9832e\_1h10\_cust\_diff\_config。

```
if [ -e kernel4.4/sprd-diffconfig/sharkle/arm64/sp9832e_1h10_diff_config ]; then
cp -fr kernel4.4/sprd-diffconfig/sharkle/arm64/sp9832e_1h10_diff_config kernel4.4/sprd-diffconfig/sharkle/arm64/sp9832e_1h10_cust_diff_config
```

3. 使用以下脚本将 kernel4.4/arch/arm64/boot/dts/sprd/sp9832e-1h10-native.dts 拷贝至 kernel4.4/arch/arm64/boot/dts/sprd/sp9832e-1h10-cust-native.dts。

```
cp -fr kernel4.4/arch/arm64/boot/dts/sprd/sp9832e-1h10-native.dts kernel4.4/arch/arm64/boot/dts/sprd/sp9832e-1h10-cust-native.dts
```

4. 使用以下脚本将 kernel4.4/arch/arm64/boot/dts/sprd/sp9832e-1h10-overlay.dts 拷贝至 kernel4.4/arch/arm64/boot/dts/sprd/sp9832e-1h10-cust-overlay.dts。

```
cp -fr kernel4.4/arch/arm64/boot/dts/sprd/sp9832e-1h10-overlay.dts kernel4.4/arch/arm64/boot/dts/sprd/sp9832e-1h10-cust-overlay.dts
```

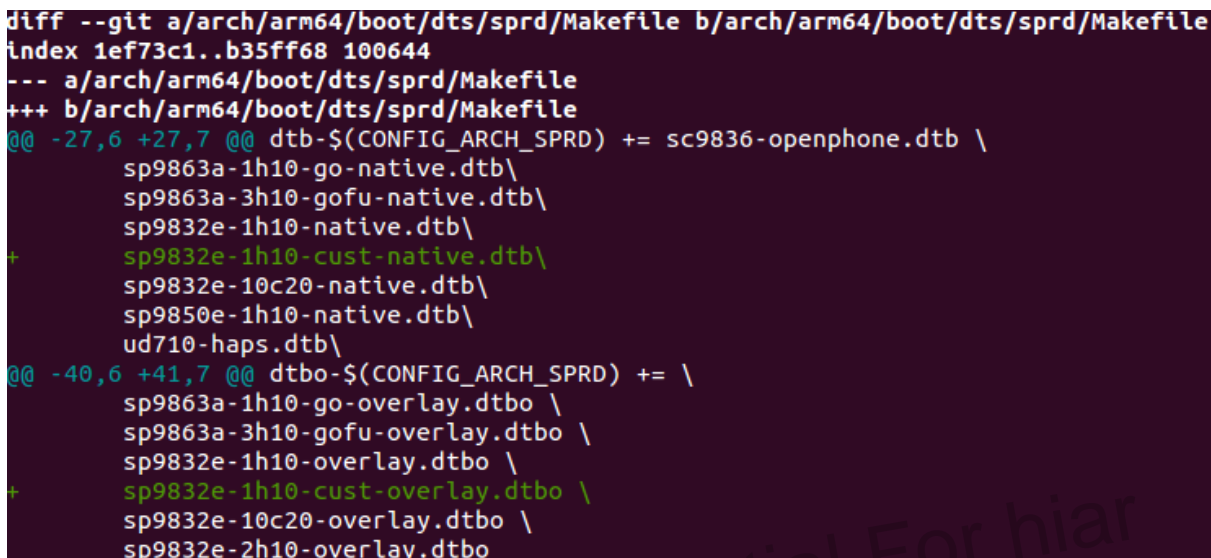
5. 使用以下脚本重命名。

```
sed -i "s/sp9832e_1h10/sp9832e_1h10_cust/" kernel4.4/arch/arm64/boot/dts/sprd/sp9832e-1h10-cust-native.dts
kernel4.4/arch/arm64/boot/dts/sprd/sp9832e-1h10-cust-overlay.dts
add config in kernel4.4/arch/arm64/boot/dts/sprd/Makefile
cd kernel4.4/sprd-board-config/sharkle/sp9832e_1h10_cust
```

```
find . -type f -exec sed -i "s/sp9832e_1h10/sp9832e_1h10_cust/g" {} +
find . -type f -exec sed -i "s/sp9832e-1h10/sp9832e-1h10-cust/g" {} +
for file in `find . -type f -iname "*sp9832e_1h10*"`; do
    mv $file ${file/sp9832e_1h10/sp9832e_1h10_cust}
done
cd ->/dev/null
```

- 修改 kernel4.4/arch/arm64/boot/dts/sprd/Makefile 的配置，如图 4-1 所示。

图4-1 修改 kernel4.4/arch/arm64/boot/dts/sprd/Makefile



```
diff --git a/arch/arm64/boot/dts/sprd/Makefile b/arch/arm64/boot/dts/sprd/Makefile
index 1ef73c1..b35ff68 100644
--- a/arch/arm64/boot/dts/sprd/Makefile
+++ b/arch/arm64/boot/dts/sprd/Makefile
@@ -27,6 +27,7 @@ dtb=$(CONFIG_ARCH_SPRD) += sc9836-openphone.dtb \
    sp9863a-1h10-go-native.dtb\
    sp9863a-3h10-gofu-native.dtb\
    sp9832e-1h10-native.dtb\
+   sp9832e-1h10-cust-native.dtb\
    sp9832e-10c20-native.dtb\
    sp9850e-1h10-native.dtb\
    ud710-haps.dtb\
@@ -40,6 +41,7 @@ dtbo=$(CONFIG_ARCH_SPRD) += \
    sp9863a-1h10-go-overlay.dtbo \
    sp9863a-3h10-gofu-overlay.dtbo \
    sp9832e-1h10-overlay.dtbo \
+   sp9832e-1h10-cust-overlay.dtbo \
    sp9832e-10c20-overlay.dtbo \
    sp9832e-2h10-overlay.dtbo
```

## 创建 uboot board

- 使用如下脚本将 u-boot15/arch/arm/dts/sp9832e\_1h10.dts 拷贝至 u-boot15/arch/arm/dts/sp9832e\_1h10\_cust.dts。

```
cp -fr u-boot15/arch/arm/dts/sp9832e_1h10.dts u-boot15/arch/arm/dts/sp9832e_1h10_cust.dts
```

- 使用如下脚本将 u-boot15/board/spreadtrum/sp9832e\_1h10 拷贝至 u-boot15/board/spreadtrum/sp9832e\_1h10\_cust。

```
cp -fr u-boot15/board/spreadtrum/sp9832e_1h10 u-boot15/board/spreadtrum/sp9832e_1h10_cust
```

- 使用如下脚本将 u-boot15/configs/sp9832e\_1h10\_defconfig 拷贝至 u-boot15/configs/sp9832e\_1h10\_cust\_defconfig。

```
cp -fr u-boot15/configs/sp9832e_1h10_defconfig u-boot15/configs/sp9832e_1h10_cust_defconfig
```

- 使用以下脚本将 u-boot15/include/configs/sp9832e\_1h10.h 拷贝到 u-boot15/include/configs/sp9832e\_1h10\_cust.h。

```
cp -fr u-boot15/include/configs/sp9832e_1h10.h u-boot15/include/configs/sp9832e_1h10_cust.h
```

- 使用以下脚本重命名。

```
sed -i "s/sp9832e_1h10/sp9832e_1h10_cust/g;s/SP9832E_1H10/SP9832E_1H10_CUST/g" u-
boot15/arch/arm/dts/sp9832e_1h10_cust.dts u-boot15/configs/sp9832e_1h10_cust_defconfig u-
boot15/include/configs/sp9832e_1h10_cust.h
cd u-boot15/board/spreadtrum/sp9832e_1h10_cust
find . -type f -exec sed -i "s/sp9832e_1h10/sp9832e_1h10_cust/g;s/SP9832E_1H10/SP9832E_1H10_CUST/g" {} +
for file in `find . -type f -iname "*sp9832e_1h10*"`; do
```

```
mv $file ${file/sp9832e_1h10/sp9832e_1h10_cust}
done
cd ->/dev/null
```

6. 修改 u-boot15/arch/arm/dts/Makefile 和 u-boot15/board/spreadtrum/Kconfig 相关配置，如图 4-2 所示。

图4-2 修改 u-boot 相关配置

```
diff --git a/arch/arm/dts/Makefile b/arch/arm/dts/Makefile
index 92583c7..45f6074 100644
--- a/arch/arm/dts/Makefile
+++ b/arch/arm/dts/Makefile
@@ -172,6 +172,7 @@ dtb-$(CONFIG_VENDOR_SPRD) += sp9850s-1h10-native.dtb \
    pike2_vp.dtb \
    sp9850e_1h10.dtb \
    sp9832e_1h10.dtb \
+   sp9832e_1h10_cust.dtb \
    sp9832e_10c20.dtb \
    sp9832e_10c10_32b.dtb \
    sp9832e_1h10_32b.dtb \

diff --git a/board/spreadtrum/Kconfig b/board/spreadtrum/Kconfig
index 025460a..6e7e0f0 100644
--- a/board/spreadtrum/Kconfig
+++ b/board/spreadtrum/Kconfig
@@ -251,6 +251,12 @@ config TARGET_SP9832E_1H10
    help
        This is the Spreadtrum sharkle Board

+config TARGET_SP9832E_1H10_CUST
+    bool "Spreadtrum sharkle"
+    select ARM64
+    help
+        This is the Spreadtrum sharkle Board
+
config TARGET_SP9832E_10C20
    bool "Spreadtrum sharkle"
    select ARM64
@@ -481,6 +487,7 @@ source "board/spreadtrum/sp9850ka_2c20/Kconfig"
source "board/spreadtrum/sp9835_haps/Kconfig"
source "board/spreadtrum/sp9850e_1h10/Kconfig"
source "board/spreadtrum/sp9832e_1h10/Kconfig"
+source "board/spreadtrum/sp9832e_1h10_cust/Kconfig"
source "board/spreadtrum/sp9832e_10c20/Kconfig"
source "board/spreadtrum/sp9832e_1h10_32b/Kconfig"
source "board/spreadtrum/sp9832e_2c10_32b/Kconfig"
```

## 创建 chipram board

1. 使用如下脚本将 chipram/include/configs/sp9832e\_1h10.h 拷贝至 chipram/include/configs/sp9832e\_1h10\_cust.h。

```
cp -fr chipram/include/configs/sp9832e_1h10.h chipram/include/configs/sp9832e_1h10_cust.h
```

2. 修改 chipram/board.def 的配置，如图 4-3 所示。

图4-3 修改 board.def 配置

```
diff --git a/board.def b/board.def
index 3532551..2ab9974 100644
--- a/board.def
+++ b/board.def
@@ -67,6 +67,9 @@ sp9820e_2c10_config : preconfig
    sp9832e_1h10_config : preconfig
        @$(MKCONFIG) $@ arm armv8 sp9832e_1h10 spreadtrum sharkle

+sp9832e_1h10_cust_config : preconfig
++    @$(MKCONFIG) $@ arm armv8 sp9832e_1h10_cust spreadtrum sharkle
++
    sp9832e_10c20_config : preconfig
        @$(MKCONFIG) $@ arm armv8 sp9832e_10c20 spreadtrum sharkle
```

## 创建 wcn board

使用如下脚本将 vendor/sprd/modules/wcn/connconfig/sharkle/sp9832e\_1h10 拷贝至 vendor/sprd/modules/wcn/connconfig/sharkle/sp9832e\_1h10\_cust。

```
cp -fr vendor/sprd/modules/wcn/connconfig/sharkle/sp9832e_1h10
vendor/sprd/modules/wcn/connconfig/sharkle/sp9832e_1h10_cust
```

Unisoc Confidential For hiar

# 5 OTA 整体及差分升级包制作

## 5.1 OTA 整体升级包制作

以 UDS710 平台的 ROC1 项目为例说明 OTA 整体升级包制作步骤。

步骤 1 下载项目 AP 部分的代码。

步骤 2 设置编译环境，命令如下：

```
source build/envsetup.sh
lunch
kheader
```

步骤 3 通过 make 命令编译整个工程。

步骤 4 进入 device/sprd/roc1/ud710\_xx（board 对应目录），手动建立 modem\_bins 子目录。

步骤 5 将展锐发布的对应 AP 版本的 modem bins 按照 device/sprd/roc1/ud710\_xx/AndroidBoard.mk 中规定的名称重命名，然后拷贝到 device/sprd/roc1/ud710\_xx/modem\_bins 目录下。

步骤 6 使用 make otapackage 命令编译 OTA 整体升级包，在 out 目录下生成 OTA 整包，整包路径：  
out/target/product/ud710\_xx/ud710\_xx-ota-\*.zip。

---结束

### 说明

- lte 平台包括 ltemodem.bin、ltedgsp.bin 等相关 bin，非 lte 平台命名是不相同的，需要仔细确认。
- 不同平台的 modem bins 重命名前后名称变化参见表 5-1 到表 5-12。
- 由于平台众多，modem bins 的多少和名字也都会有一些细微区别，如果不确定如何改名字，可以联系展锐 FAE，展锐 FAE 会给出指导。

表5-1 UDS710 改名映射

| modem bin 原名称               | 重命名后名称          |
|-----------------------------|-----------------|
| SC9600_roc1_pubcp_modem.dat | ltemodem.bin    |
| roc1_pubcp_LTEA_DSP.bin     | ltedsp.bin      |
| roc1_pubcp_DM_DSP.bin       | ltedgsp.bin     |
| roc1_cm4.bin                | pmsys.bin       |
| gnssbdmodem.bin             | gnssbdmodem.bin |
| gnssmodem.bin               | gnssmodem.bin   |

| modem bin 原名称                      | 重命名后名称         |
|------------------------------------|----------------|
| PM_roc1_cm4.bin                    | wcnmodem.bin   |
| roc1_pubcp_nvitem.bin              | ltenvitem.bin  |
| roc1_pubcp_Smart_Phone_deltanv.bin | ltedeltanv.bin |

表5-2 SC7731/7731C 改名映射

| modem bin 原名称                  | 重命名后名称       |
|--------------------------------|--------------|
| SC8800G_pike_wcn_dts_modem.bin | wcnmodem.bin |
| nvitem_wcn.bin                 | wcnfdl.bin   |
| DSP_DM_G2.bin                  | wdsp.bin     |
| SC7702_pike_modem_AndroidM.dat | wmodem.bin   |
| nvitem.bin                     | wnvitem.bin  |

表5-3 SC9832A 改名映射

| modem bin 原名称                        | 重命名后名称        |
|--------------------------------------|---------------|
| SC9600_sharkls_3593.dat              | ltemodem.bin  |
| SHARKL_DM_DSP.bin                    | ltegdsp.bin   |
| PM_sharkls_arm7.bin                  | pmsys.bin     |
| LTE_DSP.bin                          | ltedsp.bin    |
| SC9600_sharkl_wphy_5mod_volte_zc.bin | ltewarm.bin   |
| EXEC_KERNEL_IMAGE0.bin               | wcnmodem.bin  |
| fdl_wcn.bin                          | wcnnvitem.bin |
| nvitem.bin                           | ltenvitem.bin |

表5-4 SC9820A 改名映射

| modem bin 原名称                                | 重命名后名称       |
|--|--------------|
| SC9600_pikel_tddcsfb_3592_V11_RTM7910V31.bin | ltemodem.bin |
| PM_pikel_arm7.bin                            | pmsys.bin    |
| LTE_DSP.bin                                  | ltedsp.bin   |
| PIKEL_DM_DSP.bin                             | ltegdsp.bin  |



| modem bin 原名称                     | 重命名后名称        |
|-----------------------------------|---------------|
| pikel_tddcsfb_3592_V11_nvitem.bin | ltenvitem.bin |

表5-5 SC9850K 改名映射

| modem bin 原名称                  | 重命名后名称        |
|--------------------------------|---------------|
| SC9600_sharkl2_pubcp_modem.dat | ltemodem.bin  |
| sharkl2_pubcp_DM_DSP.bin       | ltegdsp.bin   |
| sharkl2_pubcp_LTEA_DSP.bin     | ltedsp.bin    |
| sharkl2_arm7.bin               | pmsys.bin     |
| sharkl2_pubcp_nvitem.bin       | ltenvitem.bin |
| EXEC_KERNEL_IMAGE.bin          | wcnmodem.bin  |

表5-6 SC9853I 改名映射

| modem bin 原名称                         | 重命名后名称         |
|---------------------------------------|----------------|
| SC9600_isharkl2_pubcp_volte_modem.dat | ltemodem.bin   |
| isharkl2_pubcp_volte_LTEA_DSP.bin     | ltedsp.bin     |
| isharkl2_pubcp_volte_DM_DSP.bin       | ltetgdsp.bin   |
| isharkl2_pubcp_volte_deltanv.bin      | ltedeltanv.bin |
| iSharkl2_pubcp_volte_V1_nvitem.bin    | ltenvitem.bin  |
| isharkl2_cm4.bin                      | pmsys.bin      |
| EXEC_KERNEL_IMAGE.bin                 | wcnmodem.bin   |

表5-7 SC9820E 改名映射

| modem bin 原名称                                | 重命名后名称          |
|--|-----------------|
| SC9600_sharkle_pubcp_Feature_Phone_modem.dat | ltemodem.bin    |
| SharkLE_FEATURE_PHONE_LTEA_DSP.bin           | ltedsp.bin      |
| SHARKLE1_DM_DSP.bin                          | ltegdsp.bin     |
| sharkle_cm4.bin                              | ltegdsp.bin     |
| gnssbdmodem.bin                              | gnssbdmodem.bin |
| gnssmodem.bin                                | gnssmodem.bin   |



| modem bin 原名称                               | 重命名后名称        |
|---|---------------|
| PM_sharkle_cm4.bin(1)                       | wcnmodem.bin  |
| sharkle_pubcp_Feature_Phone_kois_nvitem.bin | ltenvitem.bin |

表5-8 SC9832E 改名映射

| modem bin 原名称                              | 重命名后名称          |
|--|-----------------|
| SC9600_sharkle_pubcp_Smart_Phone_modem.dat | ltemodem.bin    |
| SharkLE_LTEA_DSP.bin                       | ltedsp.bin      |
| SHARKLE2_DM_DSP.bin                        | ltegdsp.bin     |
| PM_sharkle_cm4.bin                         | wcnmodem.bin    |
| sharkle_cm4.bin                            | pmsys.bin       |
| sharkle_pubcp_Smart_Phone_nvitem.bin       | ltenvitem.bin   |
| harkle_pubcp_Smart_Phone_deltanv.bin       | ltedeltanv.bin  |
| gnssbdmodem.bin                            | gnssbdmodem.bin |
| gnssmodem.bin                              | gnssmodem.bin   |

表5-9 SC9863A 改名映射

| modem bin 原名称                         | 重命名后名称          |
|---------------------------------------|-----------------|
| SC9600_sharkl3_pubcp_modem.dat        | ltemodem.bin    |
| sharkl3_pubcp_LTEA_DSP.bin            | ltedsp.bin      |
| sharkl3_pubcp_DM_DSP.bin              | ltegdsp.bin     |
| sharkl3_cm4.bin                       | pmsys.bin       |
| gnssbdmodem.bin                       | gnssbdmodem.bin |
| gnssmodem.bin                         | gnssmodem.bin   |
| PM_sharkl3_cm4.bin                    | wcnmodem.bin    |
| sharkl3_pubcp_nvitem.bin              | ltenvitem.bin   |
| sharkl3_pubcp_Smart_Phone_deltanv.bin | ltedeltanv.bin  |

表5-10 SC9850E 改名映射

| modem bin 原名称                           | 重命名后名称          |
|---|-----------------|
| SC9600_sharkle_pubcp_SharkleP_modem.dat | ltemodem.bin    |
| sharkle_pubcp_SharkleP_LTEA_DSP.bin     | ltedsp.bin      |
| sharkle_pubcp_SharkleP_DM_DSP.bin       | ltegdsp.bin     |
| PM_sharkle_cm4.bin                      | wcnmodem.bin    |
| sharkle_cm4.bin                         | pmsys.bin       |
| sharkle_pubcp_Smart_Phone_nvitem.bin    | ltenvitem.bin   |
| gnssbdmodem.bin                         | gnssbdmodem.bin |
| gnssmodem.bin                           | gnssmodem.bin   |

表5-11 SC7731E 改名映射

| modem bin 原名称                        | 重命名后名称          |
|--------------------------------------|-----------------|
| SC9600_pike2_pubcp_uncache_modem.dat | wmodem.bin      |
| PIKE2_DM_DSP.bin                     | wgdsp.bin       |
| pike2_pubcp_v1_nvitem.bin            | wnvitem.bin     |
| pike2_cm4.bin                        | pmsys.bin       |
| PM_pike2_cm4.bin                     | wcnmodem.bin    |
| gnssmodem.bin                        | gnssmodem.bin   |
| gnssbdmodem.bin                      | gnssbdmodem.bin |

表5-12 UMS312 改名映射

| modem bin 原名称                  | 重命名后名称          |
|--------------------------------|-----------------|
| SC9600_sharkl5_pubcp_modem.dat | ltemodem.bin    |
| sharkl5_pubcp_LTEA_DSP.bin     | ltedsp.bin      |
| sharkl5_pubcp_DM_DSP.bin       | ltegdsp.bin     |
| sharkl5_cm4.bin                | pmsys.bin       |
| gnssbdmodem.bin                | gnssbdmodem.bin |
| gnssmodem.bin                  | gnssmodem.bin   |
| PM_sharkl5_cm4.bin             | wcnmodem.bin    |

| modem bin 原名称                         | 重命名后名称         |
|---------------------------------------|----------------|
| sharkl5_pubcp_nvitem.bin              | ltenvitem.bin  |
| sharkl5_pubcp_Smart_Phone_deltanv.bin | ltedeltanv.bin |

## 5.2 OTA 差分升级包制作

OTA 差分升级包的制作步骤如下：

步骤 1 下载 A 版本代码，执行 5.1 OTA 整体升级包制作的所有步骤，然后保存此版本对应的 target 包 A-target.zip。

步骤 2 下载 B 版本代码，执行 5.1 OTA 整体升级包制作的所有步骤，然后保存此版本对应的 target 包 B-target.zip。

步骤 3 执行命令制作差分包。

- 正常差分包制作

```
./build/tools/releasetools/ota_from_target_files --block -k sign_key_dir -i A-target.zip B-target.zip A-B_update.zip
```

- 降级差分包制作

降级差分包是指从较新的版本反向升级到较老的版本，命令中 A 版本必须要比 B 版本更新。

```
./build/tools/releasetools/ota_from_target_files --downgrade --block -k sign_key_dir -i A-target.zip B-target.zip A-B_downgrade_update.zip
```

----结束

### 📖 说明

- A\_target.zip 与 B\_target.zip 分别是升级前和要升级到版本的 OTA target 包。
- -k 后面的 sign\_key\_dir 为实际版本的 key 的放置路径，user 版本该路径为“build/target/product/security/release/releasekey”，userdebug 版本该路径为“build/target/product/security/testkey”。
- 块升级时差分包制作命令中需加入--block。

# 6

## pac 制作及下载

### 6.1 pac 制作

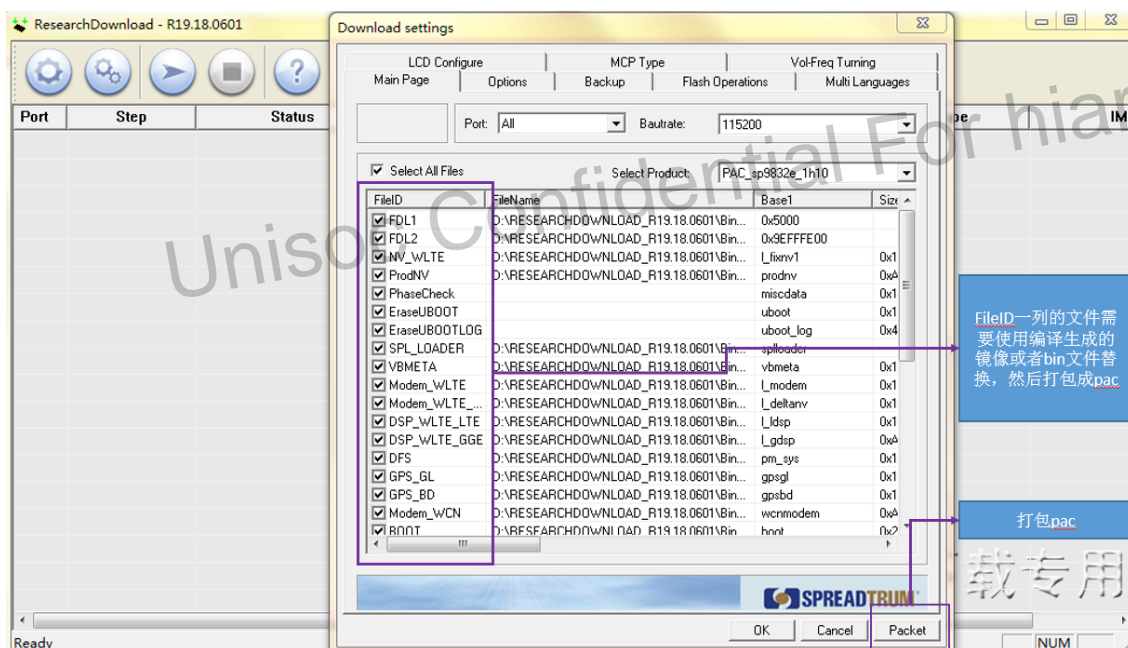
pac 制作有两种方式：

- 工具制作
- 脚本打包制作

#### 6.1.1 工具制作

工具制作方式如图 6-1 所示。

图6-1 工具制作 pac



不足：需要先 load 一个 pac，然后替换，而且替换前后的生成文件大小差异不能太大，太大可能因分配的分區大小 size 不匹配，导致烧录失败。

当分区 size 大小不匹配时，需要修改配置文件，修改分区大小。如：

RESEARCHDOWNLOA\_R19.18.0601\Bin\ImageFiles\DownloadFiles460679189: s9832e1h10.xml 文件，找到对应的分区大小设置的地方，修改 size 大小。如图 6-2 所示。

图6-2 修改 size 大小

```
<Partition id="pm_sys" size="1"/>
<Partition id="boot" size="35"/>
<Partition id="system" size="2150"/>
<Partition id="cache" size="150"/>
<Partition id="vendor" size="400"/>
<Partition id="vbmeta" size="1"/>
<Partition id="vbmeta_bak" size="1"/>
<Partition id="userdata" size="0xFFFFFFFF"/>
```

## 6.1.2 脚本制作

UNISOC 的打包脚本由版本编译与集成团队统一维护，如果没有打包脚本，可以向展锐 FAE 申请获取。

### 6.1.2.1 Git 推送版本

1. 脚本需要在 ./vendor/sprd/release/IDH/Script 目录下执行，并添加执行权限。
2. 执行 build\_pac.sh 脚本，脚本执行命令格式如下：

```
./build_pac.sh -a "参数 a" -b "参数 b" -c "参数 c"
```

以 sp9832e 平台为例，执行如下命令进行编译和打包。

```
./build_pac.sh -a sp9832e_1h10_gofu-userdebug-gms -b ALL
```

#### 说明

- 脚本参数 a 和参数 b 必填，参数 c 可选。
- 参数 a 是 ./vendor/sprd/release/IDH 目录下面 "sp\*" 目录名称，如 sp9832e\_1h10\_gofu-userdebug-gms。
- 参数 b 的所有参数选项及其含义：
  - ALL|all: 编译和打包
  - BUILD|build: 只编译
  - PAC|pac: 只打包
  - OTA|ota: 只 OTA 编译
  - BOTA|bota: 编译和 OTA 编译
- 参数 c 用于编译运营商工程，如果有多个用英文","隔开，只对编译有影响，对打包没有影响。
- 生成的 pac 包在参数 a 指定的目录的子目录（除 out 目外）下。
- 编译 OTA 生成的 target\_files 压缩包和 ota 压缩包在 ./out/target/product/项目工程/"OTA\_\*" 目录下。如 ./out/target/product/sp7731e\_1h10/OTA\_userdebug\_gms/sp7731e\_1h10\_native-ota-PIKE2\_9\_Uncache.zip 和 ./out/target/product/sp7731e\_1h10/OTA\_userdebug\_gms/sp7731e\_1h10\_native-target\_files-PIKE2\_9\_Uncache.zip。

想了解更多详情，请 ./vendor/sprd/release/IDH/Script 下的客户编译简易操作文档。

### 6.1.2.2 非 Git 推送版本

编译脚本会有 CPM 单独提供针对特定项目 (pike2/sharkle/sharkl3)，脚本内容结构如图 6-3 所示：

图6-3 脚本内容结构

|                     |                  |        |
|---------------------|------------------|--------|
| avb_sign_modem.sh   | 2018/1/25 16:57  | SH 文件  |
| pac.sh              | 2019/3/21 13:26  | SH 文件  |
| pac_via_conf.pl     | 2018/10/17 13:48 | PL 文件  |
| readme.txt          | 2018/10/17 14:57 | 文本文档   |
| sprd_720p.bmp       | 2018/7/27 17:16  | BMP 图像 |
| UpdatedPacCRC_Linux | 2017/6/13 10:59  | 文件     |

1. 将如图 6-3 所示文件拷贝到 `idh.code/out/target/product/<name>` 下，如 `idh.code/out/target/product/sp9832e_1h10`。
2. 赋予脚本可执行权限。

```
chmod a+x pac.sh
chmod a+x UpdatedPacCRC_Linux
```

3. 执行脚本 `pac.sh`。

脚本会自动对 modem 相关文件的签名，以及对生成的 pac 进行 CRC 校验。如图 6-4 所示。

图6-4 脚本使用说明

```
/home8/chunlei.liu/AndroidP/idh.code/out/target/product/sp9832e_1h10/../../../../target/product/sp9832e_1h10/SharkLE_LTEA_DSP_ews_on.bin matched: 1_ldsp
size = 20
sizeInByte = 20971520
check dat cp
not packed image, return.
start sign
normal bin, direct sign
Sign /home8/chunlei.liu/AndroidP/idh.code/out/target/product/sp9832e_1h10/../../../../target/product/sp9832e_1h10/SharkLE_LTEA_DSP_ews_on.bin succeed

Linux os
CalcCRC,pac file [sp9832e_1h10_gofu-userdebug-native_SHARKLE_9832E_9.pac].
crc first part...
wCRC1=0xb589

crc second part...
wCRC2=0x5f0c

-----
sp9832e_1h10_gofu-userdebug-native_SHARKLE_9832E_9.pac
do packet success

total size: 3517827930, total time: 21s
```

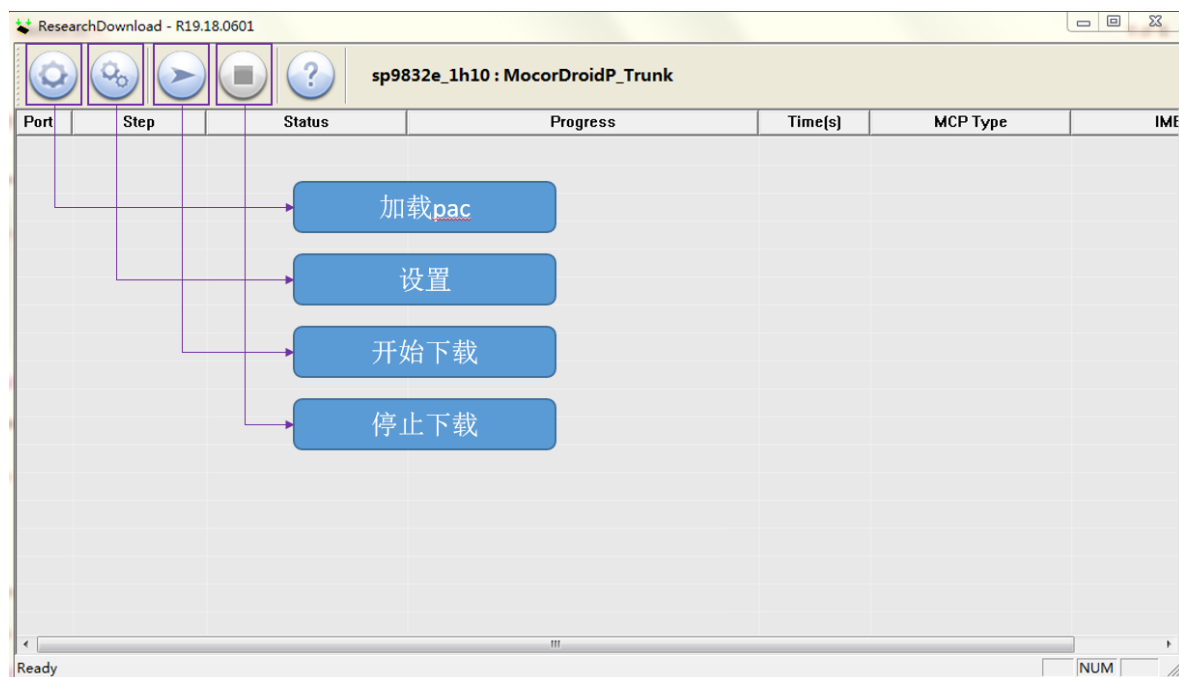
### 注意

打包脚本默认编译“`sp9832e_1h10_go-userdebug-native_SHARKLE_9832E_GO_1H10_9.pac`”工程，如果需要打包其它工程，请将 `pac.sh` 中变量 `PAC_NAME` 的值改成需要打包的工程名。

## 6.2 pac 下载

ResearchDownload 下载工具使用说明如图 6-5 所示。

图6-5 ResearchDownload 下载工具使用说明



工具下载的具体步骤如下：

- 步骤 1 在 PC 端安装手机驱动。
- 步骤 2 在 ResearchDownload 工具上加载 pac。
- 步骤 3 点击开始下载按钮。
- 步骤 4 按手机音量下键，同时将 PC 端的 USB 线连接手机，并插入电池。
- 步骤 5 完成下载后插拔电池。
- 步骤 6 重启手机。

----结束