

R16

Quick Start Guide

目录

R16.....	1
1. SDK 下载指南.....	4
2. SDK 目录结构.....	4
3. 编译环境搭建指南.....	5
4. 编译指南.....	5
4.1. 编译内核.....	5
4.1.1. 方法一.....	5
4.1.2. 方法二.....	5
4.2. 编译 brandy(本节可选).....	5
4.2.1. 编译 u-boot.....	5
4.3. 编译 Android.....	6
5. 打包指南.....	6
5.1. Android 固件打包.....	6
5.2. Dragonboard/Linux 固件打包.....	7
5.2.1. 方法一.....	7
5.2.2. 方法二.....	7
6. 烧写指南.....	7
7. Declaration.....	7

1. SDK 下载指南

暂无。

2. SDK 目录结构

└── android	
│ ├── abi	
│ ├── art	
│ ├── bionic	
│ ├── bootable	
│ ├── build	
│ ├── cts	
│ ├── dalvik	
│ ├── developers	
│ ├── development	
│ ├── device	
│ │ └── softwinner	
│ │ ├── polaris-common	平台公共目录
│ │ └── astar-evb	方案定制目录
│ ├── docs	
│ ├── external	
│ ├── frameworks	
│ ├── hardware	
│ ├── libcore	
│ ├── libnativehelper	
│ ├── Makefile	
│ ├── ndk	
│ ├── packages	
│ ├── pdk	
│ ├── prebuilts	
│ ├── sdk	
│ ├── system	
│ └── tools	
└── lichee	
├── brandy	boot 相关源码
├── buildroot	编译脚本、交叉编译工具链
├── build.sh	Top level 编译脚本
├── linux-3.4	内核目录
├── README	
└── tools	打包脚本、工具和方案配置

3. 编译环境搭建指南

请查看 Android 网站 <http://source.android.com/> 说明。

4. 编译指南

4.1. 编译内核

4.1.1. 方法一

推荐使用。

1. 配置（开启新的终端必须执行一次）

```
$ source buildroot/scripts/mksetup.sh
```

2. 编译 lichee

```
$ mklichee
```

3. 单独编译 buildroot

```
$ mkbr
```

4. 单独编译内核

```
$ mkkernel
```

4.1.2. 方法二

1. 配置（如果已经配置可以省略这个步骤）

```
$ ./build.sh config
```

2. 编译

```
$ ./build.sh
```

注意: *android*, *dragonboard* 和 *linux* 固件打包都需要编译内核（在配置时, *All available platforms* 时对应选择 *android*, *dragonboard* 或者 *linux* 区分）。

4.2. 编译 **brandy**(本节可选)

brandy 目录中存放的是 R16 平台的 bootloader, 该目录为 R16 启动代码, 默认不编译。

4.2.1. 编译 u-boot

方法一、brandy 目录下，可以快速完成 uboot 编译动作。

```
cd lichee/brandy/
./build.sh -p sun8iw5p1
```

方法二，最常用。

```
cd lichee/brandy/u-boot-2011.09
make distclean && make sun8iw5p1 -j32      #-j 开启多核编译，服务器开发一般为服务器 cpu 数量
的一半
```

当编译成功，生成的 u-boot-sun8iw5p1.bin 文件会自动拷贝到对应的 tools 目录下，这时候可以直接打包或者其它操作。u-boot-sun8iw5p1.bin 是启动时 uboot 核心可执行程序。

注意：

R16 的 boot 阶段涉及的 lcd 驱动位置是：u-boot-2011.09/drivers/video_sunxi/sunxi_v2。

4.3. 编译 Android

编译 android 前请先编译内核。

1. 选择方案

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

2. 拷贝 kernel 和 modules

```
$ extract-bsp
```

3. 编译 android

```
$ make -j?
```

?表示启用几个进程编译，一般情况下进程个数不用超过 cpu 核数。

5. 打包指南

5.1. Android 固件打包

cd 到 android 根目录

1. 打包 release 固件：

```
$ pack
```

2. 打包 debug 固件：

```
$ pack -d
```

3. 打包签名固件：

```
$ pack -s
```

5.2. Dragonboard/Linux 固件打包

注意: *dragonboard* 和 *linux* 固件打包前必须编译 *lichee* (配置 *All available platforms* 时对应选择 *dragonboard* 或者 *linux*)，参考 [4.1 编译内核](#)。

5.2.1. 方法一

cd 到 lichee 根目录

1. 配置 (开启新的终端必须执行一次)

```
$ source buildroot/scripts/mksetup.sh
```

2. 打包

```
$ mkpack [-d [card0|uart0]] [-s [none|sig]] [-m [normal|dump]] [-f [android|prvt]]
```

参数:

- d card0|uart0, card0 表示 debug 固件, uart0 表示 release 固件 (默认)
- s none|sig, none 表示非签名固件 (默认), sig 表示签名固件
- m normal|dump, normal 表示普通固件 (默认), dump 表示读取机器分区数据固件
- f android|prvt, android 表示普通固件 (默认), prvt 表示烧写 private 分区的固件

5.2.2. 方法二

cd 到 lichee 根目录

1. 配置 (如果已经配置可以省略这个步骤)

```
$ ./build.sh config
```

2. 打包

- a. 打包 release 固件

```
$ ./build.sh pack
```

- b. 打包 debug 固件

```
$ ./build.sh pack_debug
```

- c. 打包读取机器分区数据固件

```
$ ./build.sh pack_dump
```

- d. 打包烧写 private 分区的固件

```
$ ./build.sh pack_prvt
```

6. 烧写指南

安装 PhoenixSuit, 使用指南请查看《PhoenixSuit 使用说明文档》。

7. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner.

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.