

密级状态：绝密() 秘密() 内部资料() 公开(☒)

PX3SE_Linux 开发说明

(技术部，第二系统产品部)

文件状态： [] 草稿 [] 正式发布 [<input checked="" type="checkbox"/>] 正在修改	当前版本：	V1.6
	作 者：	王剑辉
	完成日期：	2018-01-29
	审 核：	邓训金、陈海燕
	完成日期：	2018-01-30

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Semiconductor Co., Ltd

(版本所有, 翻版必究)

版 本 历 史

日期	修订版本	修订内容	修改人	核定人
2017-03-13	V1.0	初始版本	王剑辉	邓训金 张文平
2017-05-18	V1.1	支持 4 种存储方案	黄国椿	邓训金 王剑辉
2017-07-10	V1.2	添加补丁说明	王剑辉	邓训金 蓝斌元
2017-08-15	V1.3	去除补丁、sdk 兼容硬件 V10_20161124 版本	王剑辉	邓训金 蓝斌元
2017-09-19	V1.4	增加 sd 升级和 U 盘升级、添加 px3se linux 常见问题汇总	王剑辉	邓训金
2017-11-06	v1.5	增加 slc_nand 小容量多分区存储方案，以及多分区存储方案生产工具下载使用说明，添加分区表说明	黄国椿	邓训金 王剑辉
2018-01-29	V1.6	完善文档、fix 测试发现的 bug	王剑辉	邓训金 陈海燕

目 录

1 概述	1
2 SDK 获取说明	1
2.1 repo 下载	1
2.2 SDK 下载和同步	1
3 SDK 编译说明	1
3.1 px3se-slc 小容量存储（小容量推荐使用）	1
3.2 px3se emmc 大容量存储（大容量推荐使用）	2
3.3 px3se emmc 小容量存储（方便调试小容量，实际量产不常用）	2
3.4 px3se sfc 小容量存储（spi nor flash 使用）	3
3.5 px3se slc 小容量存储按多分区编译	3
4 固件烧写说明	5
4.1 px3se usb 下载口和串口	5
4.2 px3se 大容量存储固件烧写	6
4.3 px3se 小容量存储固件烧写	6
5 支持列表	7
5.1 DDR 支持列表	7
5.2 Flash 支持列表	8
5.2 Wifi 支持列表	8
6 兼容硬件 RK_SDK_MAIN_PX3SE_CAR_V10 版本	8
6.1 回退 wifi 提交	9
6.2 内置和外置 codec 适配介绍	9
7 分区表配置介绍	9
8 功能模块说明	9
8.1 SD 卡/U 盘升级	9
8.2 Q/A 文档	9
9 量产工具说明	9
10 工程目录介绍	9
附录 A 编译开发环境搭建	11
附录 B SSH 公钥操作说明	12

1 概述

本 SDK 是基于 buildroot 文件系统，内核基于 kernel 3.10，兼容多种存储方案，根据存储容量分别定制不同容量需求的文件系统和烧写方式。

2 SDK 获取说明

SDK 通过我司代码服务器对外发布。搭建编译开发环境参考[附录 A 编译开发环境搭建](#)。客户向我司技术窗口申请 SDK，需同步提供 SSH 公钥进行服务器认证授权，获得授权后即可同步代码。关于我司代码服务器 SSH 公钥授权，请参考[附录 B SSH 公钥操作说明](#)。

2.1 repo 下载

repo 是用来管理调用 git 的一个脚本，主要用来下载管理软件仓库，务必使用我司提供的 repo 进行初始化操作。repo 工程下载地址如下：

```
git clone
```

```
ssh://git@www.rockchip.com.cn:2222/repo-release/tools/repo.git
```

下载后即可获取到 repo：

```
ls repo/repo
```

```
repo/repo
```

将 repo 拷贝到任意位置，以供下一步使用。建议放到（/home/bin/repo）。

2.2 SDK 下载和同步

使用步骤 2.1 获取的 repo 工程进行初始化。假如 repo 工程目录/home/bin/repo，那么 px3se 工程下载地址如下：

```
repo init --repo-url=ssh://git@www.rockchip.com.cn:2222/repo-release/tools/repo.git -u ssh://git@www.rockchip.com.cn:2222/px3-se/manifests.git -m px3_se_release.xml
```

```
repo sync
```

3 SDK 编译说明

3.1 px3se-slc 小容量存储（小容量推荐使用）

Building kernel

```
make ARCH=arm px3se_linux_slc_defconfig -j8
```

```
make ARCH=arm px3se-slc-sdk.img -j24
```

Building rootfs

```
cd buildroot/ && make rockchip_px3se_minifs_defconfig -j8 &&
```

```
cd .. && ./build_all.sh
```

Pack and compress the firmware

```
./mkfirmware_minifs.sh px3se-slc-sdk slc
```

编译的输出固件在工程目录 rockimg/Image-slc:

```
rockimg/Image-slc/
```

```
Firmware.img
```

```
px3se_usb_boot_V1.22.bin
```

3.2 px3se emmc 大容量存储（大容量推荐使用）

Building uboot

```
make px3se_linux_defconfig && make -j12
```

Building kernel

```
make ARCH=arm px3se_linux_defconfig -j8
```

```
make ARCH=arm px3se-sdk.img -j24
```

Build rootfs and app

```
cd buildroot && make rockchip_px3se_defconfig && cd ..
```

```
&& ./build_all.sh && ./mkfirmware.sh
```

编译的输出固件在工程目录的 rockimg/下:

```
rockimg/
```

```
Kernel.img
```

```
parameter-emmc.txt
```

```
Px3SeMiniLoaderAll_V2.32.bin
```

```
resource.img
```

```
Rootfs.img
```

```
Uboot.img
```

3.3 px3se emmc 小容量存储（方便调试小容量，实际量产不常用）

Building kernel

```
make ARCH=arm px3se_linux_emmc_minifs_defconfig -j8
```

```
make ARCH=arm px3se-emmc-minifs-sdk.img -j24
```

Building rootfs

```
cd buildroot/ && make rockchip_px3se_minifs_defconfig -j8 &&
```

```
cd .. && ./build_all.sh
```

Pack and compress the firmware

```
./mkfirmware_minifs.sh px3se-emmc-minifs-sdk emmc
```

编译的输出固件在工程目录 rockimg/Image-emmc:

```
rockimg/Image-emmc/
```

```
Firmware.img
```

```
px3se_usb_boot_V1.22.bin
```

3.4 px3se sfc 小容量存储（spi nor flash 使用）

Building kernel

```
make ARCH=arm px3se_linux_sfc_defconfig -j8
```

```
make ARCH=arm px3se-sfc-sdk.img -j24
```

Building rootfs

```
cd buildroot/ && make rockchip_px3se_minifs_defconfig -j8 &&
```

```
cd .. && ./build_all.sh
```

Pack and compress the firmware

```
./mkfirmware_minifs.sh px3se-sfc-sdk sfc
```

编译的输出固件在工程目录 rockimg/Image-sfc:

```
rockimg/Image-sfc/
```

```
Firmware.img
```

```
px3se_usb_boot_V1.22.bin
```

3.5 px3se slc 小容量存储按多分区编译

为了适应产品缩短开机时间，对 px3se 的 slc nand 提供多分区的编译方式：

编译之前做如下修改：

- 在工程的 kernel/下对描述 nand 的设备信息节点做如下修改：

```
diff --git a/arch/arm/boot/dts/px3se-sdk.dtsi b/arch/arm/boot/dts/px3se-sdk.dtsi
index 02efa6b..fef01f 100644
--- a/arch/arm/boot/dts/px3se-sdk.dtsi
+++ b/arch/arm/boot/dts/px3se-sdk.dtsi
@@ -118,12 +118,10 @@
 };

 &nandc {
-    compatible = "rockchip,nandc";
    status = "okay"; // used nand set "okay", used emmc set "disabled"
 };

 &nandc0reg {
-    compatible = "rockchip,nandc";
    status = "disabled"; // used nand set "disabled", used emmc set "okay"
 };
```

- 在工程 buildroot/下对 rootfs 的配置文件做如下修改:

```
+++ b/configs/rockchip_px3se_defconfig
@@ -114,12 +114,7 @@ BR2_PACKAGE_WPA_SUPPLICANT_WPS=y
 BR2_PACKAGE_WPA_SUPPLICANT_CLI=y
 BR2_PACKAGE_WPA_SUPPLICANT_WPA_CLIENT_SO=y
 BR2_PACKAGE_WPA_SUPPLICANT_PASSPHRASE=y
-BR2_TARGET_ROOTFS_EXT2=y
-BR2_TARGET_ROOTFS_EXT2_4=y
+BR2_TARGET_ROOTFS_SQUASHFS=y
 # BR2_TARGET_ROOTFS_TAR is not set
 BR2_PACKAGE_HOST_VBOOT_UTILS=y
 BR2_PACKAGE_LIBXML2=y
-BR2_PACKAGE_STRESS=y
-BR2_PACKAGE_STRESS_NG=y
-BR2_PACKAGE_GLMARK2=y
-BR2_PACKAGE_MEMTESTER=y
```

- 在工程根目录下对 mkfirmware.sh 做如下修改:

```
+++ b/mkfirmware.sh
@@ -4,15 +4,19 @@ IMAGE_RELEASE_PATH=$(pwd)/rockimg/Image-release
 KERNEL_PATH=$(pwd)/kernel
 UBOOT_PATH=$(pwd)/u-boot
 PRODUCT_PATH=$(pwd)/device/rockchip/px3-se/
+BUILDROOT_TARGET_PATH=$(pwd)/buildroot/output/target/
+
+cp $PRODUCT_PATH/sdcard-udisk-udev/rules.d/add-sdcard-udisk-nand.rules $BUILDROOT_TARGET_PATH/etc/udev/rules.d/add-sdcard-udisk.rules
+cp $PRODUCT_PATH/sdcard-udisk-udev/rules.d/remove-sdcard-udisk-nand.rules $BUILDROOT_TARGET_PATH/etc/udev/rules.d/remove-sdcard-udisk.rules
+
 #cd buildroot && make && cd -
 rm -rf $IMAGE_OUT_PATH
 mkdir -p $IMAGE_OUT_PATH
 echo "package rootfs.img now"
 source $PRODUCT_PATH/mkrootfs.sh
-cp $(pwd)/buildroot/output/images/rootfs.ext4 $IMAGE_OUT_PATH/rootfs.img
+cp $(pwd)/buildroot/output/images/rootfs.squashfs $IMAGE_OUT_PATH/rootfs.img
-cp $PRODUCT_PATH/rockimg/parameter-emmc.txt $IMAGE_OUT_PATH/
+cp $PRODUCT_PATH/rockimg/parameter-slc.txt $IMAGE_OUT_PATH/
```

- 在工程 device/rockchip/px3-se/下, 把 rockimg/parameter-emmc.txt 改为 rockimg/parameter-slc.txt, 并做如下修改:

```
root=/dev/rknand_boot rootfstype=squashfs
```

固件编译方式如下:

Building uboot

```
make px3se_linux_defconfig && make -j12
```

Building kernel

```
make ARCH=arm px3se_linux_defconfig -j8
```

```
make ARCH=arm px3se-sdk.img -j24
```

Build rootfs and app

```
cd buildroot && make rockchip_px3se_defconfig && cd ..
```

```
&& ./build_all.sh && ./mkfirmware.sh slc
```

备注：

若需要编译单个模块或者第三方应用，需对交叉编译环境进行配置。

交叉编译工具位于 `buildroot/output/host/usr` 目录下，需要将工具的 `bin/` 目录和 `arm-rockchip-linux-gnueabi/hf/bin/` 目录设为环境变量，在顶层目录自动配置环境变量的脚本（只对当前控制台有效）：

```
source envsetup.sh
```

输入命令查看：

```
arm-linux-gcc --version
```

此时会打印出以下 log 即标志为配置成功：

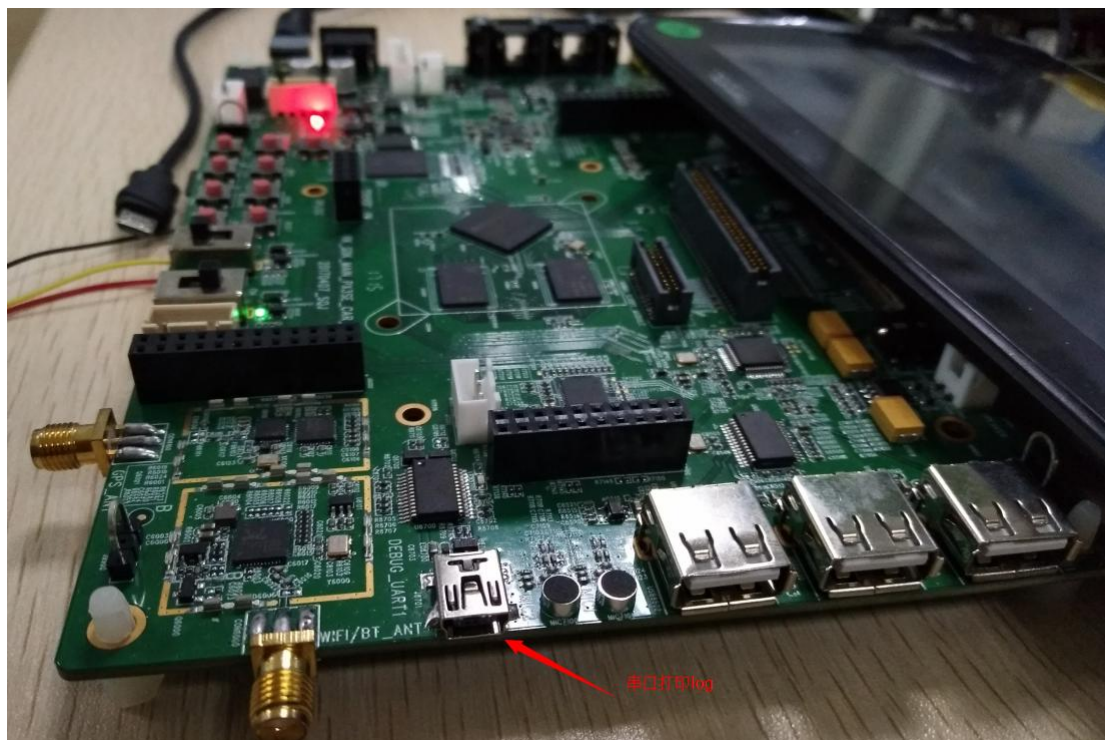
```
arm-linux-gcc.br_real(Buildroot 2016.08.1)
```

4 固件烧写说明

（注：烧写前，需要安装最新的 `usb` 驱动，驱动在工程目录 `tools/windows/DriverAssitant_v4.2`）

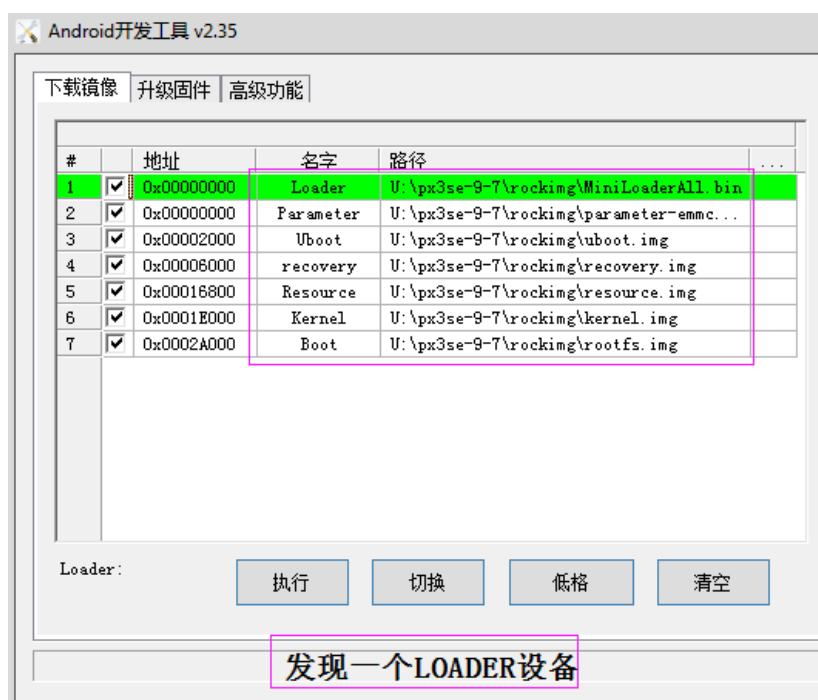
4.1 px3se usb 下载口和串口





4.2 px3se 大容量存储固件烧写

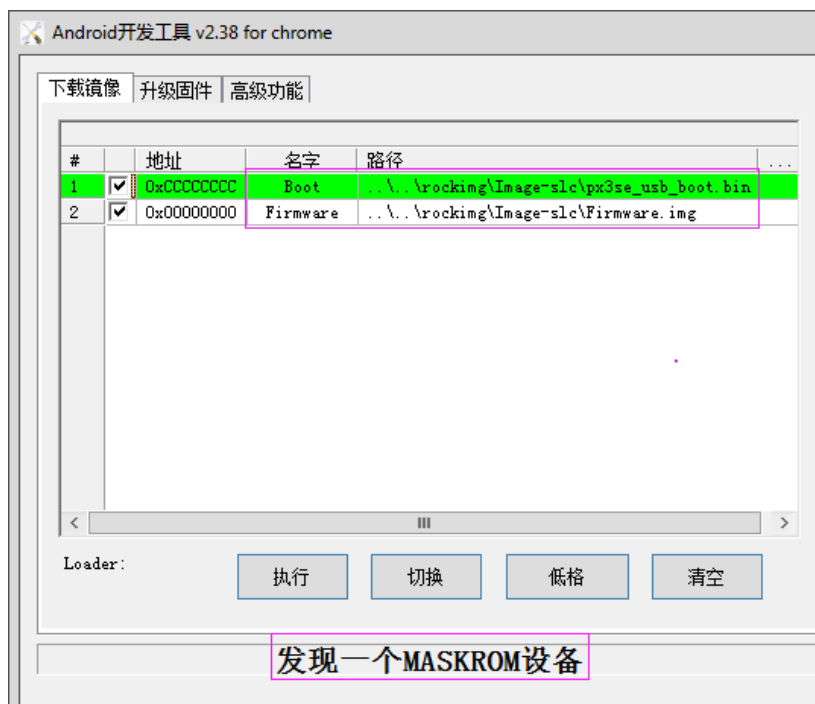
px3se 大容量存储固件烧写工具位于工程目录 `tool/windows/AndroidTool`。烧写操作步骤：插入 usb 下载线并按住 **RESET+ VOL+** 按键组合进入下载模式(工具底下会提示“发现一个 LOADER 设备”)，将固件路径加载后就可以执行烧写。



4.3 px3se 小容量存储固件烧写

px3se 小容量存储固件烧写工具位于 `tool/windows/Minifs_Tools_Release_v1.0`

0. 烧写操作步骤: 插入 usb 下载线并按住 RESET + UPDATE 按键组合进入下载模式(工具底下会提示“发现一个 MASKROM 设备”), 将固件路径加载后就可以执行:



5 支持列表

5.1 DDR 支持列表

PX3-SE DDR 目前选型列表支持 DDR3、DDR3L、LPDDR2

1.1.4 External Memory or Storage device

- Dynamic Memory Interface (DDR3/DDR3L/LPDDR2)
 - Compatible with JEDEC standard DDR3/DDR3L/LPDDR2 SDRAM
 - Data rates up to 800Mbps(400MHz) for DDR3/DDR3L/LPDDR2
 - Support up to 2 ranks (chip selects) for each channel, totally 4GB(max) address space.
 - 16bits/32bits data width is software programmable
 - 7 host ports with 64bits/128bits AXI bus interface for system access, AXI bus clock is asynchronous with DDR clock
 - Programmable timing parameters to support DDR3/DDR3L/LPDDR2 SDRAM from various vendor
 - Advanced command reordering and scheduling to maximize bus utilization
 - Low power modes, such as power-down and self-refresh for DDR3/LPDDR2 SDRAM; clock stop and deep power-down for LPDDR2 SDRAM
 - Compensation for board delays and variable latencies through programmable pipelines
 - Programmable output and ODT impedance with dynamic PVT compensation

5.2 Flash 支持列表

- Nand Flash Interface
 - Support 8bits async/toggle/syncnandflash, up to 4 banks
 - Support LBA nandflash
 - 16bits, 24bits, 40bits, 60bits hardware ECC
 - For DDR nandflash, support DLL bypass and 1/4 or 1/8 clock adjust, maximum clock rate is 66.5MHz
 - For async/togglenandflash, support configurable interface timing , maximum data rate is 16bit/cycle
 - Embedded AHB master interface to do data transfer by DMA method
 - Also support data transfer by AHB slave interface together with external DMAC
- eMMC Interface
 - Compatible with standard iNAND interface
 - Support MMC4.41 protocol
 - Provide eMMC boot sequence to receive boot data from external eMMC device
 - Support FIFO over-run and under-run prevention by stopping card clock automatically
 - Support CRC generation and error detection
 - Embedded clock frequency division control to provide programmable baud rate
 - Support block size from 1 to 65535Bytes
 - 8bits data bus width
- SD/MMC Interface
 - Compatible with SD3.0, MMC ver4.41
 - Support FIFO over-run and under-run prevention by stopping card clock automatically
 - Support CRC generation and error detection
 - Embedded clock frequency division control to provide programmable baud rate
 - Support block size from 1 to 65535Bytes
 - Data bus width is 4bits

5.2 Wifi 支持列表

PX3SE Wi-Fi Situation												
WiFi Chip	IFACE	IEEE 802.11 Standard	2.4GHz Band	5.0GHz Band	BT	GPS	NFC	IIAC	SDIO3.0	MIMO	BT4.0	BT4.2
RTL8723CS/DS	SDIO	IEEE 802.11B/G/N	✓	×	✓	×	×	✓	×	×	×	✓
1. ✓：支持 ×：不支持 注：空的表示没调过												
2. 该列表仅适用kernel3.10												

6 兼容硬件 RK_SDK_MAIN_PX3SE_CAR_V10 版本

注意：目前我们有两个版本硬件，旧 sdk 板（旧板 RK_SDK_MAIN_PX3SE_CAR_V10 蓝色板子），新 sdk 板（新板 RK_SDK_MAIN_PX3SE_CAR_V11 绿色板子）。服务器上代码默认是支持新板 RK_SDK_MAIN_PX3SE_CAR_V11 绿色板子。假如您手上的 sdk 板是旧板，即 RK_SDK_MAIN_PX3SE_CAR_V10 蓝色板子，请做以下两个操作（新板忽略以下两点）：

6.1 回退 wifi 提交

在 kernel 目录下面，回退这个提交：

```
commit b9ebb0d3858fd324225358a72a19b6de924081ad
Author: wjh <wjh@rock-chips.com>
Date:   Fri Aug 4 12:37:10 2017 +0800

    arm: dts: px3se: wifi module use rtl8723ds

    Change-Id: Icf3dd9863d3bdedf5eb5b3ba17dd29861d2dd57

    Signed-off-by: wjh <wjh@rock-chips.com>
```

操作命令如下：

```
git revert b9ebb0d3858fd324225358a72a19b6de924081ad
```

6.2 内置和外置 codec 适配介绍

Codec 适配的介绍，请参考 docs 目录下面《PX3-SE 音频开发文档.pdf》。服务器上默认的代码是适配 **RK_SDK_MAIN_PX3SE_CAR_V11** 绿色板子。

7 分区表配置介绍

多分区和 gpt 分区大小配置，请参考 docs 目录下面《PX3-SE 分区表配置说明_V1.0_20171102.pdf》

8 功能模块说明

8.1 SD 卡/U 盘升级

SD 卡升级和 U 盘升级，请参考 docs 目录下面《PX3-SE_Recovery 开发说明_V1.0_20170908.pdf》和《PX3-SE 大容量 EMMC 升级固件制作说明.pdf》。

8.2 Q/A 文档

开发过程中，会遇到一些常见的问题，docs 目录下面《PX3SE_Linux 常见问题解决方法_V1.0.pdf》记录了一些常见的问题。

9 量产工具说明

量产固件生成和烧写请参考 docs 目录下面《PX3SE_Linux 量产固件生成和下载说明》。

10 工程目录介绍

工程目录下有 buildroot、app、kernel、u-boot、device、docs、external、prebuilts、rockimg、tools 等目录。每个目录或其子目录会对应一个 git 工程，提交需

要在各自的目录下进行。

- 1) **buildroot**: 用于生成根文件系统、交叉编译工具以及相关工具和应用的管理;
- 2) **app**: 存放上层应用 **app**, **carmachin** 是 SDK 的 Demo 应用;
- 3) **external**: 相关库, 包括音频、视频、网络等;
- 4) **kernel**: **kernel** 代码;
- 5) **device/rockchip/px3-se**: 存放开机初始化脚本, 存放第三方库、**bin**、**alsa/wifi** 等配置文件; 另还存放编译脚本, 系统根目录的几个 **sh** 脚本都是在 **repo sync** 的时候, 从这里拷贝出来的, 所以若要提交修改的脚本, 必须在 **device/rockchip/px3-se** 目录下进行;

build_all.sh: 编译所有第三方库和应用;

mkfirmware.sh: 打包最终烧写的固件;

mkfirmware_mini.sh: 打包用于压缩成 **Firmware** 固件的合成文件;

envsetup.sh: 终端环境变量设置的脚本;

- 6) **docs**: 存放工程帮助文件;
- 7) **prebuilts**: 存放编译 **kernel** 需要的 **gcc** 和交叉编译工具 **toolschain**;
- 8) **rockimg**: 存放编译输出固件;
- 9) **tools**: 存放平台工具。

附录 A 编译开发环境搭建

1. 初始化开发环境

本部分内容包括如何搭建用于 px3se Linux 开发的本地环境。您需要在 Linux 或者 MacOS 环境下搭建，建议使用 Ubuntu14.04 64bit 开发，与我司的开发环境统一，避免出现环境问题。

2. 配置一个 Linux 开发环境

本创建步骤是基于最新的 Ubuntu LTS(14.04)版本，但是大部分发行版本必须保证所需的工具可以运行。

注意：您也可以在虚拟机中搭建环境。如果您在虚拟机中运行 Linux，您需要至少 2GB 的 RAM/swap，或者 30GB 以上的磁盘空间来创建编译环境。

3. 安装所需的安装包（基于 Ubuntu 14.04）

您需要一个 64 位版本的 Ubuntu，推荐使用 Ubuntu14.04。注意：使用老版本 Ubuntu 可能会有兼容性问题。用下面命令来安装 Ubuntu 所需的包：

```
$ sudo apt-get install git gnupg flex bison gperf build-essential \  
zip tar curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \  
libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \  
libgl1-mesa-dev g++-multilib mingw32 cmake tofrodos \  
python-markdown libxml2-utils xsltproc zlib1g-dev:i386 lzop
```

4.gcc 和 g++版本要求

编译环境的 gcc 和 g++工具版本要求 4.8 以上版本

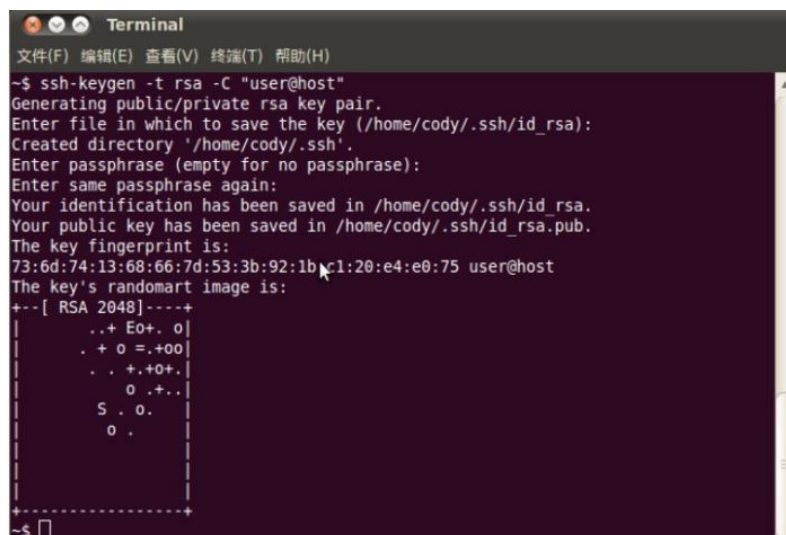
附录 B SSH 公钥操作说明

附录 B-1 SSH 公钥生成

使用如下命令生成：

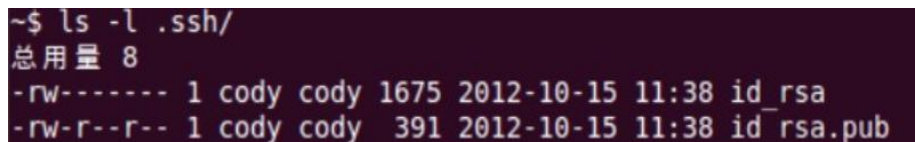
```
ssh-keygen -t rsa -C "user@host"
```

请将 **user@host** 替换成您的邮箱地址。



```
Terminal
文件(F) 编辑(E) 查看(V) 终端(T) 帮助(H)
~$ ssh-keygen -t rsa -C "user@host"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cody/.ssh/id_rsa):
Created directory '/home/cody/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cody/.ssh/id_rsa.
Your public key has been saved in /home/cody/.ssh/id_rsa.pub.
The key fingerprint is:
73:6d:74:13:68:66:7d:53:3b:92:1b:c1:20:e4:e0:75 user@host
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      ..+ Eo+. o |
|      . + o =. +oo |
|      . . +. +o+ . |
|      o . +. . |
|      S . o . |
|      o . |
+-----+
~$
```

命令运行完成会在您的目录下生成公钥（id_rsa.pub）和私钥（id_rsa）。



```
~$ ls -l .ssh/
总用量 8
-rw----- 1 cody cody 1675 2012-10-15 11:38 id_rsa
-rw-r--r-- 1 cody cody 391 2012-10-15 11:38 id_rsa.pub
```

请妥善保管生成的私钥文件 **id_rsa** 和密码，并将公钥 **id_rsa.pub** 发邮件给 SDK 发布服务器的管理员。

附录 B-2 Git 权限申请说明

参考上述章节，生成公钥文件，发邮件至 fae@rock-chips.com，申请开通 SDK 代码下载权限。