

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 [] 正在修改	文件标识：	Px3se-linux 开发说明
	当前版本：	V1.1
	作 者：	黄国椿
	完成日期：	2017-05-18
	审 核：	邓训金、王剑辉
	完成日期：	2017-05-18

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)

文档修改记录:

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

目录

1、概述	4
2、SDK 获取说明.....	4
2.1、repo 下载.....	4
2.2、SDK 下载和同步.....	4
3、交叉编译环境的配置.....	4
4、Boad config.....	5
5、SDK 编译说明.....	5
5.1、px3se emmc 大容量存储	5
5.2、px3se emmc 小容量存储	6
5.3、px3se sfc 小容量存储.....	6
5.4、px3se-slc 小容量存储.....	7
6、固件烧写说明.....	7
6.1、px3se emmc 大容量存储固件烧写	7
6.2、px3se 小容量存储固件烧写	8
7、工程目录介绍	8
附录 A 编译开发环境搭建.....	9
附录 B SSH 公钥操作说明.....	10

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
```

下载后执行即可获取到 repo：

```
cd repo
tar xvf repo.tar.gz
```

拷贝解压出来的 repo 目录到任意位置，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/user/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、交叉编译环境的配置

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/f/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、Boad config

product	kernel config	flash type
px3se-sdk.dts	px3se_linux_defconfig	emmc
px3se-emmc-minifs-sdk.dts	px3se_linux_emmc_minifs_defconfig	
px3se-sfc-sdk.dts	px3se_linux_sfc_defconfig	sfc
px3se-slc-sdk.dts	px3se_linux_slc_defconfig	slc

5、SDK 编译说明

5.1、px3se emmc 大容量存储

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
```

5.2、px3se emmc 小容量存储

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 rocking/Image-emmc:

```
rocking/Image-emmc/  
Firmware.img  
px3se_usb_boot_V1.22.bin
```

5.3、px3se sfc 小容量存储（128M 以下建议使用）

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 rocking/Image-sfc:

```
rocking/Image-sfc/  
Firmware.img  
px3se_usb_boot_V1.22.bin
```

5.4、px3se-slc 小容量存储（128M 以下建议使用）

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 rocking/Image-slc:

```
rocking/Image-slc/  
Firmware.img  
px3se_usb_boot_V1.22.bin
```

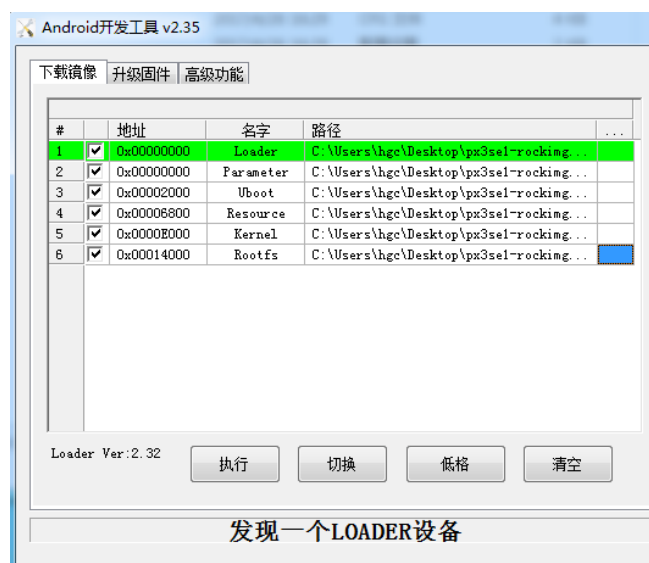
6、固件烧写说明

SDK 提供固件烧写工具，位于工程目录 tool/windows/AndroidTool/

（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）

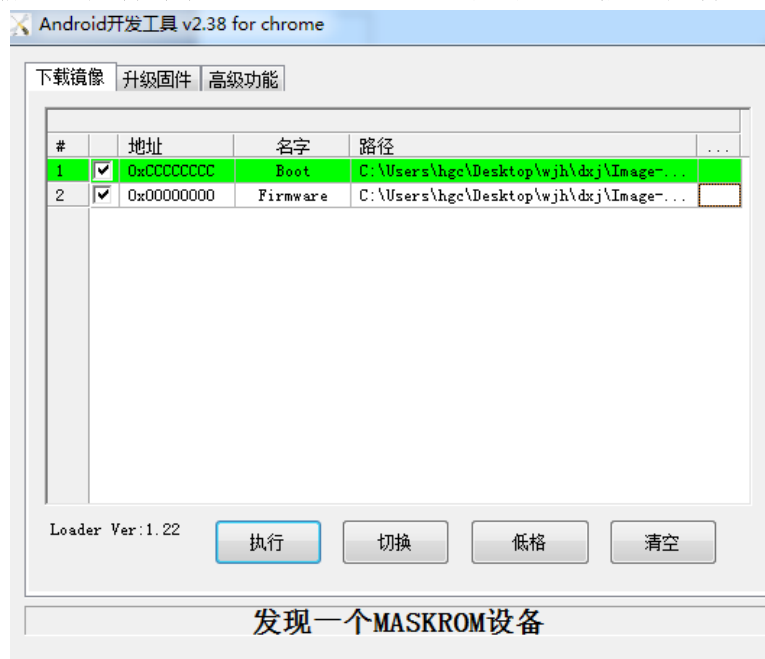
6.1、px3se emmc 大容量存储固件烧写

出入 usb 下载线按住 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。



6.2、px3se 小容量存储固件烧写

将编译输出的固件路径加载进入 MASKROM 后就可以烧写固件。



7、工程目录介绍

进工程目录下有 buildroot、app、kernel、u-boot、device、docs、external、prebuilts、rocking、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) docs/patches/kernel/0001-vehicle-add-vehicle-driver-for-px3se.patch: 快速倒车的补丁;
- 8) prebuilts: 存放编译 kernel 需要的 gcc 和交叉编译工具 toolchain;
- 9) rocking: 存放编译输出固件;
- 10) 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 Mac OS 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。

在 Ubuntu 或者 MacOS 下，通常您需要安装如下工具：

- A. Python 2.6 — 2.7, 您可以从 python.org 下载.
- B. GNU Make 3.81 — 3.82, 您可以从 gnu.org 下载.
- C. Git 1.7 or newer. 您可以从 git-scm.com 下载.

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
$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gn
u/libGL.so
```

4. 配置 USB 接入

在 GUN/Linux 系统中（特别是在 Ubuntu 系统中），在默认配置下用户不能直接接入 USB 设备。系统需要配置为可允许接入。推荐方法：创建一个/etc/udev/rules.d/51-android.rules（使用 root 用户），然后复制下列内容到文件中。<username>必须是经授权的用 USB 连接设备的用户的实际用户名。

```
# adb protocol on passion (Rockchip products)SUBSYSTEM=="usb",
ATTR{idVendor}=="2207",ATTR{idProduct}=="0010",MODE="0600",
OWNER="<username>"
```

新的配置将在下一次设备接入时生效，因此必须重新拔插设备。本方法支持 Ubuntu Hardy Heron (8.04.x LTS) 以及 Lucid Lynx(10.04.x LTS)。其他版本的 Ubuntu 或者其他类型的 GNU/linux 可能需要不同的配置。

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

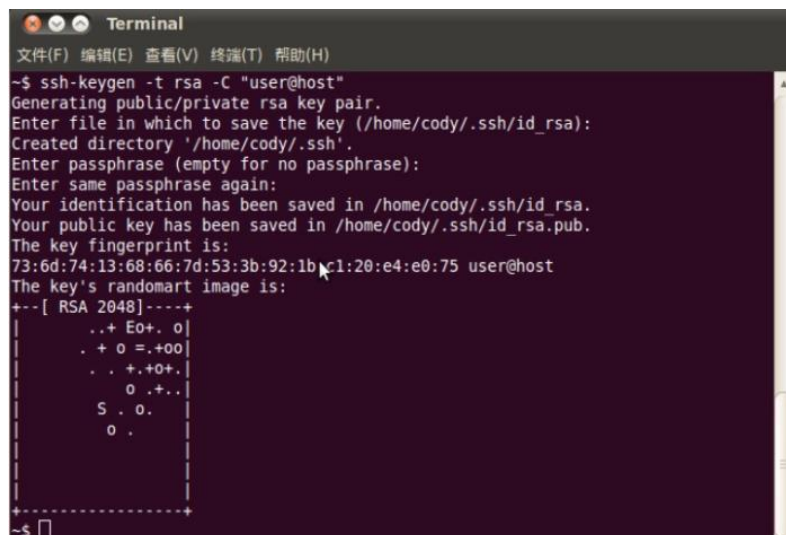
附录 B SSH 公钥操作说明

附录 B-1 SSH 公钥生成

使用如下命令生成：

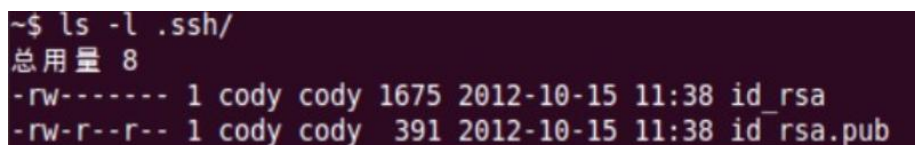
```
ssh-keygen -t rsa -C "user@host"
```

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 |
|      . + 0 =. +00 |
|      . . +. +0+. |
|      o  . +.. |
|      S . 0. |
|      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。