

Rockchip RK3399Pro Linux SDK 发布说明

文档标识: RK-FB-CS-009

发布版本: V1.3.1

日期: 2020-07-22

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

免责声明

本文档按“现状”提供, 瑞芯微电子股份有限公司(“本公司”, 下同)不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因, 本文档将可能在未经任何通知的情况下, 不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标, 归本公司所有。

本文档可能提及的其他所有注册商标或商标, 由其各自所有者所有。

版权所有© 2020 瑞芯微电子股份有限公司

超越合理使用范畴, 非经本公司书面许可, 任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部, 并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址: 福建省福州市铜盘路软件园A区18号

网址: www.rock-chips.com

客户服务电话: +86-4007-700-590

客户服务传真: +86-591-83951833

客户服务邮箱: fae@rock-chips.com

前言

概述

文档主要介绍 Rockchip RK3399Pro Linux SDK发布说明，旨在帮助工程师更快上手RK3399Pro Linux SDK开发及相关调试方法。

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

各芯片系统支持状态

芯片名称	Buildroot	Debian 9	Debian 10	Yocto
RK3399Pro	Y	Y	N	Y

修订记录

日期	版本	作者	修改说明
2019-02-17	V0.0.1	Caesar Wang	初始 Beta 版本
2019-03-21	V0.0.2	Caesar Wang	修改5.1.3中./mkfirmware.sh生成image的方法。 更改8章节中 rknn_demo 用例，添加Debian 的说明。 更改8章节中SDK固件升级到v0.0.2。
2019-06-06	V1.0.0	Caesar Wang	正式发布版本，添加 NPU 相关说明。 增加 Yocto 的编译说明，增加 github 下载说明。
2019-06-21	V1.0.1	Caesar Wang	修改软件开发指南名字
2020-10-14	V1.1.2	Caesar Wang	修改 Debian 编译说明
2020-10-23	V1.1.3	Caesar Wang	支持 RK3399Pro EVB V13 编译
2020-12-03	V1.2.0	Caesar Wang	章节 3、4、6、7、8、9、10 内容更改
2020-03-24	V1.3.0	Caesar Wang	增加 RK3399Pro V14 的支持
2020-07-22	V1.3.1	Ruby Zhang	更新公司名称，文件名以及文档格式

目录

Rockchip RK3399Pro Linux SDK 发布说明

1. 概述
2. 主要支持功能
3. SDK 获取说明
4. 软件开发指南
 - 4.1 开发指南
 - 4.2 NPU 开发工具
 - 4.3 软件更新记录
5. 硬件开发指南
6. SDK 工程目录介绍
7. SDK 编译说明
 - 7.1 NPU 编译说明
 - 7.1.1 U-boot 编译
 - 7.1.2 Kernel 编译步骤
 - 7.1.3 Boot.img 以及 NPU 固件生成步骤
 - 7.1.4 全自动编译
 - 7.2 RK3399Pro 编译说明
 - 7.2.1 U-boot 编译
 - 7.2.2 Kernel 编译步骤
 - 7.2.3 Recovery 编译步骤
 - 7.2.4 Buildroot rootfs 及 APP 编译
 - 7.2.5 Debian rootfs 编译
 - 7.2.6 Yocto rootfs 编译
 - 7.2.7 全自动编译
 - 7.2.8 固件的打包
8. 刷机说明
 - 8.1 Windows 刷机说明
 - 8.2 Linux 刷机说明
 - 8.3 系统分区说明
9. RK3399Pro SDK 固件及简单 Demo 测试
 - 9.1 RK3399Pro SDK 固件
 - 9.2 RKNN_DEMO 测试
 - 9.3 N4 Camera 测试
10. SSH 公钥操作说明
 - 10.1 多台机器使用相同 SSH 公钥
 - 10.2 一台机器切换不同 SSH 公钥
 - 10.3 密钥权限管理
 - 10.4 参考文档

1. 概述

本 SDK 支持三个系统分别基于 Buildroot 2018.02-rc3, Yocto Thud 2.6, 和 Debian 9 上开发, 内核基于 Kernel 4.4, 引导基于 U-boot v2017.09, 适用于 RK3399Pro EVB 开发板及基于此开发板进行二次开发的所有 Linux 产品。本 SDK 支持 NPU TensorFlow/Caffe 模型、VPU 硬解码、GPU 3D、Wayland 显示、QT 等功能。具体功能调试和接口说明, 请阅读工程目录 docs/ 下文档。

2. 主要支持功能

功能	模块名
数据通信	Wi-Fi、以太网卡、USB、SD 卡、PCI-e 接口
应用程序	多媒体播放、设置、浏览器、文件管理

3. SDK 获取说明

SDK 通过瑞芯微代码服务器对外发布或者从 [Github](#) 开源网站上获取。其编译开发环境, 参考第 7 节 [SDK 编译说明](#)。

获取 SDK 方法一: 从瑞芯微代码服务器获取源码

获取 RK3399Pro Linux 软件包, 需要有一个帐户访问 Rockchip 提供的源代码仓库。客户向瑞芯微技术窗口申请 SDK, 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。关于瑞芯微代码服务器 SSH 公钥授权, 请参考第 10 节 [SSH 公钥操作说明](#)。

RK3399Pro_Linux_SDK 下载命令如下:

```
1 | repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo -u
  | ssh://git@www.rockchip.com.cn/linux/rk/platform/manifests -b linux -m
  | rk3399pro_linux_release.xml
```

repo 是 google 用 Python 脚本写的调用 git 的一个脚本, 主要是用来下载、管理项目的软件仓库, 其下载地址如下:

```
1 | git clone ssh://git@www.rockchip.com.cn/repo/rk/tools/repo
```

为方便客户快速获取 SDK 源码, 瑞芯微技术窗口通常会提供对应版本的 SDK 初始压缩包, 开发者可以通过这种方式, 获得 SDK 代码的初始压缩包, 该压缩包解压得到的源码, 进行同步后与通过 repo 下载的源码是一致的。以 rk3399pro_linux_sdk_release_v1.3.0_20200324.tgz 为例, 拷贝到该初始化包后, 通过如下命令可检出源码:

```
1 | mkdir rk3399pro
2 | tar xvf rk3399pro_linux_sdk_release_v1.3.0_20200324.tgz -C rk3399pro
3 | cd rk3399pro
4 | .repo/repo/repo sync -l
5 | .repo/repo/repo sync
```

后续开发者可根据 FAE 窗口定期发布的更新说明，通过 “.repo/repo/repo sync” 命令同步更新。

获取 SDK 方法二：从 Github 开源网站获取源码

下载 repo 工具：

```
1 | git clone https://github.com/rockchip-linux/repo.git
```

建立 rk3399pro linux 工作目录

```
1 | mkdir rk3399pro_linux
```

进入 rk3399pro linux 工作目录

```
1 | cd rk3399pro_linux/
```

初始化 repo 仓库

```
1 | ../repo/repo init --repo-url=https://github.com/rockchip-linux/repo -u
https://github.com/rockchip-linux/manifests -b master -m
rk3399pro_linux_release.xml
```

同步下载整个工程：

```
1 | ../repo/repo sync
```

注意：如果是已立项的项目请优先选择用方法一获取代码，不同于 Github 的是它会经过内部稳定测试和版本控制，方法二更多适用于爱好者和前期项目评估。

4. 软件开发指南

4.1 开发指南

RK3399Pro Linux SDK Kernel 版本是 Kernel 4.4，Rootfs 分别是 Buidlroot(2018.02-rc3)、Yocto(Thud 2.6) 和 Debian9，为帮助开发工程师更快上手熟悉 SDK 的开发调试工作，随 SDK 发布《Rockchip_Developer_Guide_Linux_Software_xx.pdf》。可在 docs/ 目录下获取，并会不断完善更新。

4.2 NPU 开发工具

本 SDK NPU 开发工具如下：

RKNN_DEMO (MobileNet SSD) : RKNN 的 Demo 请参考目录 external/rknn_demo/, 相关操作说明详见工程目录 docs/Soc_public/RK3399PRO/ Rockchip_Developer_Guide_Linux_RKNN_DEMO_CN.pdf。

RKNN-TOOLKIT : 开发工具在 external/rknn-toolkit 目录下, 主要用来实现模型转换, 模型推理, 模型性能评估功能等, 具体使用说明请参考当前 doc/ 的目录文档:

```
1 | └─ Rockchip_Developer_Guide_RKNN_Toolkit_Custom_OP_CN.pdf
2 | └─ Rockchip_Developer_Guide_RKNN_Toolkit_Custom_OP_EN.pdf
3 | └─ Rockchip_Quick_Start_RKNN_Toolkit_V1.3.0_CN.pdf
4 | └─ Rockchip_Quick_Start_RKNN_Toolkit_V1.3.0_EN.pdf
5 | └─ Rockchip_Trouble_Shooting_RKNN_Toolkit_V1.3_CN.pdf
6 | └─ Rockchip_Trouble_Shooting_RKNN_Toolkit_V1.3_EN.pdf
7 | └─ Rockchip_User_Guide_RKNN_Toolkit_V1.3.0_CN.pdf
8 | └─ Rockchip_User_Guide_RKNN_Toolkit_V1.3.0_EN.pdf
9 | └─ Rockchip_User_Guide_RKNN_Toolkit_Visualization_CN.pdf
10| └─ Rockchip_User_Guide_RKNN_Toolkit_Visualization_EN.pdf
```

RKNN-DRIVER: RKNN DRIVER 开发内容在工程目录 external/rknpu 下。

RKNPUTools: RKNN API的开发使用在工程目录 external/RKNPUTools 下。

NPU 软件启动说明: RK3399Pro 的 NPU 软件启动说明, 请参考工程目录 docs/Soc_public/RK3399PRO/ Rockchip_RK3399Pro_Instruction_Linux_NPU_CN.pdf。

4.3 软件更新记录

软件发布版本升级通过工程 xml 进行查看, 具体方法如下:

```
1 | .repo/manifests$ ls -l -h rk3399pro_linux_release.xml
```

软件发布版本升级更新内容通过工程文本可以查看, 具体方法如下:

```
1 | .repo/manifests$ cat rk3399pro_linux_v0.01/RK3399PRO_Release_Note.txt
```

或者参考工程目录:

```
1 | <SDK>/docs/SoC_public/RK3399PRO/RK3399PRO_Linux_SDK_Release_Note.pdf
```

5. 硬件开发指南

硬件相关开发可以参考用户使用指南, 在工程目录:

```
1 | <SDK>/docs/SoC_public/RK3399PRO/Rockchip_RK3399Pro_User_Guide_Hardware_xx.pdf
```

6. SDK 工程目录介绍

SDK目录包含有 buildroot、debian、recovery、app、kernel、u-boot、device、docs、external 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。

- app: 存放上层应用 APP，主要是 qcamera/qfm/qplayer/qsetting 等一些应用程序。
- buildroot: 基于 Buildroot (2018.02-rc3) 开发的根文件系统。
- debian: 基于 Debian 9 开发的根文件系统。
- device/rockchip: 存放各芯片板级配置以及一些编译和打包固件的脚步和预备文件。
- docs: 存放开发指导文件、平台支持列表、工具使用文档、Linux 开发指南等。
- distro: 基于 Debian 10 开发的根文件系统。
- IMAGE: 存放每次生成编译时间、XML、补丁和固件目录。
- external: 存放第三方相关仓库，包括音频、视频、网络、recovery 等。
- kernel: 存放 Kernel 4.4 开发的代码。
- npu: 存放 NPU 开发的代码。
- prebuilts: 存放交叉编译工具链。
- rkbin: 存放 Rockchip 相关 Binary 和工具。
- rockdev: 存放编译输出固件。
- tools: 存放 Linux 和 Window 操作系统下常用工具。
- u-boot: 存放基于 v2017.09 版本进行开发的 U-Boot 代码。
- yocto: 存放基于 YoctoThud 2.6 开发的根文件系统。

7. SDK 编译说明

Ubuntu 16.04 系统: 编译 Buildroot 环境搭建所依赖的软件包安装命令如下:

```
1 | sudo apt-get install repo git-core gitk git-gui gcc-arm-linux-gnueabi u-  
boot-tools device-tree-compiler gcc-aarch64-linux-gnu mtools parted libudev-  
dev libusb-1.0-0-dev python-linaro-image-tools linaro-image-tools autoconf  
autotools-dev libsigsegv2 m4 intltool libdrm-dev curl sed make binutils  
build-essential gcc g++ bash patch gzip bzip2 perl tar cpio python unzip  
rsync file bc wget libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev  
libglade2-dev cvs git mercurial rsync openssh-client subversion asciidoc w3m  
dblatex graphviz python-matplotlib libc6:i386 libssl-dev texinfo liblz4-tool  
genext2fs expect patchelf xutils-dev
```

编译 Debian 环境搭建所依赖的软件包安装命令如下:

```
1 | sudo apt-get install repo git-core gitk git-gui gcc-arm-linux-gnueabi u-  
boot-tools device-tree-compiler gcc-aarch64-linux-gnu mtools parted libudev-  
dev libusb-1.0-0-dev python-linaro-image-tools linaro-image-tools gcc-arm-  
linux-gnueabi libssl-dev gcc-aarch64-linux-gnu g++ conf autotools-dev  
libsigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essential gcc  
g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc wget  
libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs git  
mercurial rsync openssh-client subversion asciidoc w3m dblatex graphviz  
python-matplotlib libc6:i386 libssl-dev texinfo liblz4-tool genext2fs xutils-  
dev
```

Ubuntu 17.04 或更高版本系统: 除了上述外还需如下依赖包:

```
1 | sudo apt-get install lib32gcc-7-dev g++-7 libstdc++-7-dev
```

建议使用 Ubuntu18.04 系统或更高版本开发，若编译遇到报错，可以视报错信息，安装对应的软件包。

注意：RK3399Pro 每次上电启动后会加载 NPU 固件。默认 NPU 固件都是预编好放到 Rootfs 的 /usr/share/npu_fw 目录下, NPU 固件烧写以及启动方式请参考文档

```
1 | <SDK>/docs/Soc_public/RK3399PRO/Rockchip_RK3399Pro_Instruction_Linux_NPU_CN.pdf。
```

下面分别对 NPU 和 RK3399Pro 固件编译方法进行介绍：

7.1 NPU 编译说明

7.1.1 U-boot 编译

进入工程 npu/u-boot 目录下执行 make.sh 来获取 rknpu_lion_loader_v1.03.103.bin trust.img uboot.img:

rk3399pro-npu:

```
1 | ./make.sh rknpu-lion
```

编译后生成文件在 u-boot 目录下：

```
1 | u-boot/  
2 | └─ rknpu_lion_loader_v1.03.103.bin  
3 | └─ trust.img  
4 | └─ uboot.img
```

7.1.2 Kernel 编译步骤

进入工程目录根目录执行以下命令自动完成 kernel 的编译及打包：

RK3399Pro EVB V10/V11/V12 开发板：

```
1 | cd npu/kernel  
2 | git checkout remotes/rk/stable-4.4-rk3399pro_npu-linux  
3 | make ARCH=arm64 rk3399pro_npu_defconfig  
4 | make ARCH=arm64 rk3399pro-npu-evb-v10.img -j12
```

RK3399Pro EVB V13/V14 开发板：

```
1 | cd npu/kernel  
2 | git checkout remotes/rk/stable-4.4-rk3399pro_npu-pcie-linux  
3 | make ARCH=arm64 rk3399pro_npu_pcie_defconfig  
4 | make ARCH=arm64 rk3399pro-npu-evb-v10-multi-cam.img -j12
```

7.1.3 Boot.img 以及 NPU 固件生成步骤

进入工程 npu 目录执行以下命令自动完成 boot.img 的打包：RK3399Pro EVB V10/V11/V12 板：


```
1 | cd npu
2 | ./build.sh ramboot
3 | ./mkfirmware.sh rockchip_rk3399pro-npu
```

RK3399Pro EVB V13/V14 开发板:

```
1 | cd npu/device/rockchip
2 | cp rk3399pro-npu-multi-cam/BoardConfig.mk .BoardConfig.mk
3 | cd - && cd npu
4 | ./build.sh ramboot
5 | ./mkfirmware.sh rockchip_rk3399pro-npu-multi-cam
```

7.1.4 全自动编译

上述 Kernel/U-Boot/Rootfs 各个部分的编译，进入工程目录根目录执行以下命令自动完成所有 的编译：
RK3399Pro EVB V10/V11/V12 开发板:

```
1 | cd npu/device/rockchip
2 | cp rk3399pro-npu/BoardConfig.mk .BoardConfig.mk
3 | cd - && cd npu
4 | ./build.sh uboot
5 | ./build.sh kernel
6 | ./build.sh ramboot
7 | ./mkfirmware.sh rockchip_rk3399pro-npu
```

RK3399Pro EVB V13/V14 开发板:

```
1 | cd npu/device/rockchip
2 | cp rk3399pro-npu-multi-cam/BoardConfig.mk .BoardConfig.mk
3 | cd ../../
4 | ./build.sh uboot
5 | ./build.sh kernel
6 | ./build.sh ramboot
7 | ./mkfirmware.sh rockchip_rk3399pro-npu-multi-cam
```

在 rockdev 目录下生成 boot.img, uboot.img, trust.img, MiniLoaderAll.bin 注意: rockdev下生成 NPU 固件需要存放在 Rootfs 指定位置/usr/share/npu_fw。

7.2 RK3399Pro 编译说明

7.2.1 U-boot 编译

进入工程 u-boot 目录下执行 make.sh 来获取 rk3399pro_loader_v1.24.119.bin trust.img uboot.img:
RK3399Pro EVB 板子:

```
1 | ./make.sh rk3399pro
```

编译后生成文件在 u-boot 目录下:

```
1 | u-boot/  
2 | └─ rk3399pro_loader_v1.24.119.bin  
3 | └─ trust.img  
4 | └─ uboot.img
```

7.2.2 Kernel 编译步骤

进入工程目录根目录执行以下命令自动完成 kernel 的编译及打包：RK3399Pro EVB V10 开发板：

```
1 | cd kernel  
2 | make ARCH=arm64 rockchip_linux_defconfig  
3 | make ARCH=arm64 rk3399pro-evb-v10-linux.img -j12
```

RK3399Pro EVB V11/V12 开发板：

```
1 | cd kernel  
2 | make ARCH=arm64 rockchip_linux_defconfig  
3 | make ARCH=arm64 rk3399pro-evb-v11-linux.img -j12
```

RK3399Pro EVB V13 开发板：

```
1 | cd kernel  
2 | make ARCH=arm64 rockchip_linux_defconfig  
3 | make ARCH=arm64 rk3399pro-evb-v13-linux.img -j12
```

RK3399Pro EVB V14 开发板：

```
1 | cd kernel  
2 | make ARCH=arm64 rockchip_linux_defconfig  
3 | make ARCH=arm64 rk3399pro-evb-v14-linux.img -j12
```

编译后在 kernel 目录生成 boot.img，此 boot.img 就是包含 Kernel 的 Image 和 DTB。

7.2.3 Recovery 编译步骤

进入工程目录根目录执行以下命令自动完成 Recovery 的编译及打包：RK3399Pro EVB 开发板：

```
1 | ./build.sh recovery
```

编译后在 Buildroot 目录 output/rockchip_rk3399pro_recovery/images 生成 recovery.img。

7.2.4 Buildroot rootfs 及 APP 编译

进入工程目录根目录执行以下命令自动完成 Rootfs 的编译及打包：RK3399Pro EVB V10/V11/V12 开发板：

```
1 | cd device/rockchip/rk3399pro
2 | cp BoardConfig_rk3399pro_usb.mk ../BoardConfig.mk
3 | cd - && ./build.sh rootfs
```

RK3399Pro EVB V13 开发板:

```
1 | cd device/rockchip/rk3399pro
2 | cp BoardConfig_rk3399pro_multi_cam_pcie.mk ../BoardConfig.mk
3 | cd - && ./build.sh rootfs
```

RK3399Pro EVB V14 开发板:

```
1 | ./build.sh rootfs
```

编译后在 Buildroot 目录 output/rockchip_rk3399pro_combine/images 下生成 rootfs.ext4。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/rockchip_rk3399pro_combine/host/usr 目录下, 需要将工具的 bin/ 目录和 aarch64-buildroot-linux-gnu/bin/ 目录设为环境变量, 在顶层目录执行自动配置环境变量的脚本 (只对当前控制台有效):

```
1 | source envsetup.sh
```

输入命令查看:

```
1 | aarch64-linux-gcc --version
```

此时会打印如下信息:

```
1 | aarch64-linux-gcc.br_real (Buildroot 2018.02-rc3-01797-gcd6c508) 6.5.0
```

7.2.5 Debian rootfs 编译

```
1 | ./build.sh debian
```

或进入 debian/ 目录:

```
1 | cd debian/
```

后续的编译和 Debian 固件生成请参考当前目录 readme.md。

(1) Building base Debian system

```
1 | sudo apt-get install binfmt-support qemu-user-static live-build
2 | sudo dpkg -i ubuntu-build-service/packages/*
3 | sudo apt-get install -f
```

编译 64 位的 Debian:

```
1 | RELEASE=stretch TARGET=desktop ARCH=arm64 ./mk-base-debian.sh
```

编译完成会在 debian/ 目录下生成: linaro-stretch-alip-xxxxx-1.tar.gz (xxxxx 表示生成时间戳)。

FAQ:

- 上述编译如果遇到如下问题情况:

```
1 noexec or nodev issue /usr/share/debootstrap/functions: line 1450:
2 .... /rootfs/ubuntu-build-service/stretch-desktop-armhf/chroot/test-dev-null:
Permission denied E: Cannot install into target
'/home/foxlue/work3/rockchip/rk_linux/rk3399_linux/rootfs/ubuntu-build-
service/stretch-desktop-armhf/chroot' mounted with noexec or nodev
```

解决方法:

```
1 mount -o remount,exec,dev xxx (xxx 是工程目录), 然后重新编译
```

另外如果还有遇到其他编译异常, 先排除使用的编译系统是 ext2/ext4 的系统类型。

- 编译 Base Debian 由于访问国外网站, 国内网络会经常出现下载失败的情况:

Debian 9 使用 live build, 镜像源改为国内可以这样配置:

```
1 +++ b/ubuntu-build-service/stretch-desktop-arm64/configure
2 @@ -11,6 +11,11 @@ set -e
3 echo "I: create configuration"
4 export LB_BOOTSTRAP_INCLUDE="apt-transport-https gnupg"
5 lb config \
6 + --mirror-bootstrap "http://mirrors.163.com/debian" \
7 + --mirror-chroot "http://mirrors.163.com/debian" \
8 + --mirror-chroot-security "http://mirrors.163.com/debian-security" \
9 + --mirror-binary "http://mirrors.163.com/debian" \
10 + --mirror-binary-security "http://mirrors.163.com/debian-security" \
11 --apt-indices false \
12 --apt-recommends false \
13 --apt-secure false \
```

如果其他网络原因不能下载包, 有预编生成的包分享在[百度云网盘](#)

(2) Building rk-debian rootfs

编译 64 位的 Debian:

```
1 VERSION=debug ARCH=arm64 ./mk-rootfs-stretch.sh
```

(3) Creating the ext4 image(linaro-rootfs.img)

```
1 ./mk-image.sh
```

此时会生成 linaro-rootfs.img。

7.2.6 Yocto rootfs 编译

进入工程目录根目录执行以下命令自动完成 Rootfs 的编译及打包: RK3399Pro EVB 开发板:

```
1 | ./build.sh yocto
```

编译后在 yocto 目录 build/lastest 下生成 rootfs.img。

FAQ: 上面编译如果遇到如下问题情况:

```
1 | Please use a locale setting which supports UTF-8 (such as LANG=en_US.UTF-8).
2 | Python can't change the filesystem locale after loading so we need a UTF-8
3 | when Python starts or things won't work.
```

解决方法:

```
1 | locale-gen en_US.UTF-8
2 | export LANG=en_US.UTF-8 LANGUAGE=en_US.en LC_ALL=en_US.UTF-8
```

或者参考 [setup-locale-python3](#) 编译后生成的 image 在 yocto/build/lastest/rootfs.img, 默认用户名登录是 root。Yocto 更多信息请参考 [Rockchip Wiki](#)。

7.2.7 全自动编译

完成上述 Kernel/U-Boot/Recovery/Rootfs 各个部分的编译后, 进入工程目录根目录执行以下命令自动完成所有的编译:

```
1 | $./build.sh all
```

默认是 Buildroot, 可以通过设置环境变量 RK_ROOTFS_SYSTEM 指定 rootfs。比如需要 Yocto 可以通过以下命令进行生成:

```
1 | $export RK_ROOTFS_SYSTEM=yocto
2 | $./build.sh all
```

具体参数使用情况, 可 help 查询, 比如:

```
1 | rk3399pro$ ./build.sh --help
2 | Usage: build.sh [OPTIONS]
3 | Available options:
4 | BoardConfig*.mk    -switch to specified board config
5 | uboot               -build uboot
6 | spl                 -build spl
7 | kernel              -build kernel
8 | modules             -build kernel modules
9 | toolchain           -build toolchain
10 | rootfs              -build default rootfs, currently build buildroot as
    default
11 | buildroot           -build buildroot rootfs
12 | ramboot             -build ramboot image
13 | multi-npu_boot     -build boot image for multi-npu board
14 | yocto               -build yocto rootfs
15 | debian              -build debian9 stretch rootfs
16 | distro              -build debian10 buster rootfs
17 | pcba                -build pcba
18 | recovery            -build recovery
```

```

19 all -build uboot, kernel, rootfs, recovery image
20 cleanall -clean uboot, kernel, rootfs, recovery
21 firmware -pack all the image we need to boot up system
22 updateimg -pack update image
23 otapackage -pack ab update otapackage image
24 save -save images, patches, commands used to debug
25 allsave -build all & firmware & updateimg & save
26
27 Default option is 'allsave'.

```

每个板子的板级配置需要在 /device/rockchip/rk3399pro/Boardconfig.mk 进行相关配置。RK3399Pro EVB 主要配置如下：

```

1 # Target arch
2 export RK_ARCH=arm64
3 # Uboot defconfig
4 export RK_UBOOT_DEFCONFIG=rk3399pro
5 # Kernel defconfig
6 export RK_KERNEL_DEFCONFIG=rockchip_linux_defconfig
7 # Kernel dts
8 export RK_KERNEL_DTS=rk3399pro-evb-v14-linux
9 # boot image type
10 export RK_BOOT_IMG=boot.img
11 # kernel image path
12 export RK_KERNEL_IMG=kernel/arch/arm64/boot/Image
13 # parameter for GPT table
14 export RK_PARAMETER=parameter.txt
15 # Buildroot config
16 export RK_CFG_BUILDRROOT=rockchip_rk3399pro_combine
17 # Recovery config
18 export RK_CFG_RECOVERY=rockchip_rk3399pro_recovery

```

7.2.8 固件的打包

上面 Kernel/U-Boot/Recovery/Rootfs 各个部分的编译后，进入工程目录根目录执行以下命令自动完成所有固件打包到 rockdev 目录下：

固件生成：

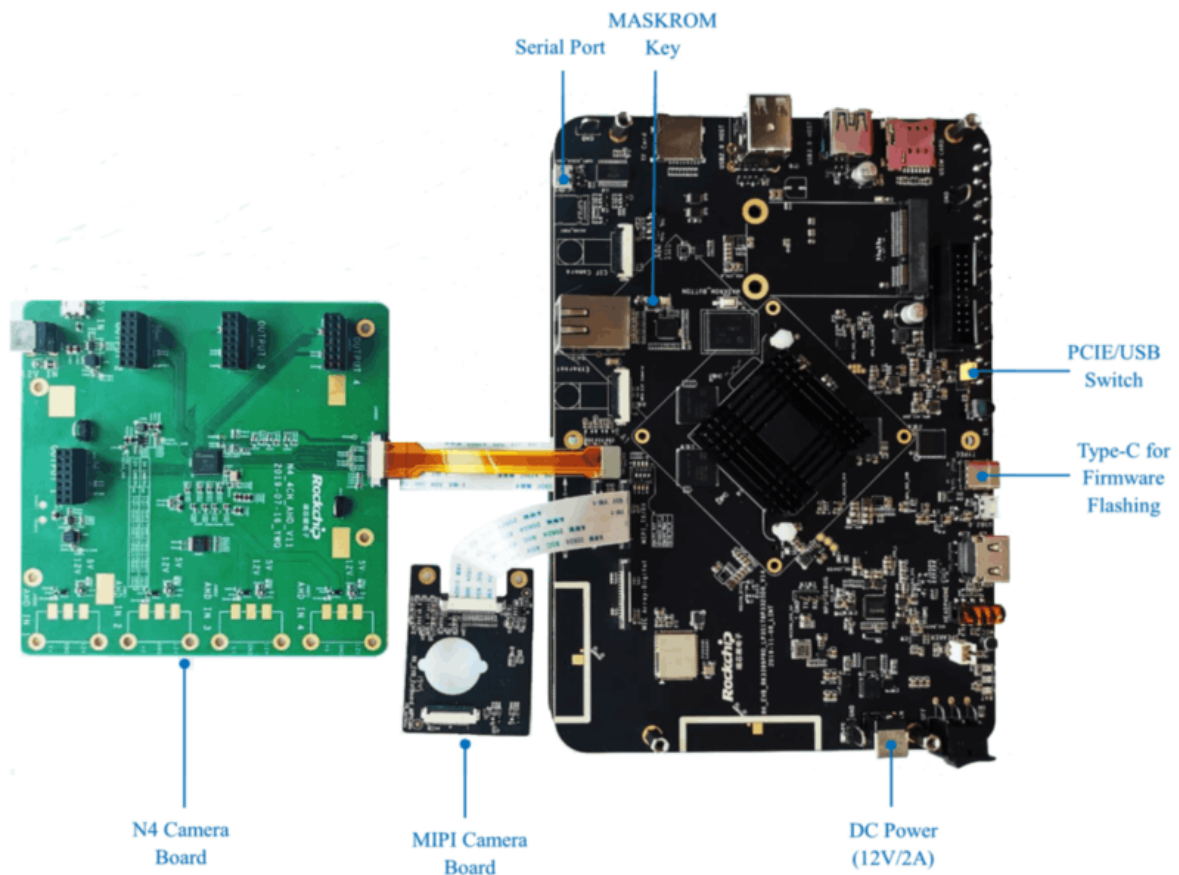
```

1 | ./mkfirmware.sh

```

8. 刷机说明

目前 RK3399Pro EVB 有 V10/V11/V12/V13/V14，5个版本，绿色板子是 V10 版本，黑色板子是 V11/V12/V13/V14 版本。板子功能位置是一样，下面以 RK3399Pro EVB V14 板子做介绍，如下图说明。



8.1 Windows 刷机说明

SDK 提供 Windows 烧写工具(工具版本需要 V2.55 或以上), 工具位于工程根目录:

```
1 | tools/  
2 | └─ windows/AndroidTool
```

如下图, 编译生成相应的固件后, 设备烧写需要进入 MASKROM 或 BootROM 烧写模式, 连接好 USB 下载线后, 按住按键“MASKROM”不放并按下复位键“RST”后松手, 就能进入 MASKROM 模式, 加载编译生成固件的相应路径后, 点击“执行”进行烧写, 也可以按“recovery”按键不放并按下复位键“RST”后松手进入 loader 模式进行烧写, 下面是 MASKROM 模式的分区偏移及烧写文件。(注意: Window PC 需要在管理员权限运行工具才可执行)



注：烧写前，需安装最新 USB 驱动，驱动详见：

```
1 | <SDK>/tools/windows/DriverAssitant_v4.8.zip
```

8.2 Linux 刷机说明

Linux 下的烧写工具位于 tools/linux 目录下(Linux_Upgrade_Tool 工具版本需要 V1.33 或以上)，请确认你的板子连接到 MASKROM/loader rockusb。比如编译生成的固件在 rockdev 目录下，升级命令如下：

```
1 | sudo ./upgrade_tool ul rockdev/MiniLoaderAll.bin
2 | sudo ./upgrade_tool di -p rockdev/parameter.txt
3 | sudo ./upgrade_tool di -u rockdev/uboot.img
4 | sudo ./upgrade_tool di -t rockdev/trust.img
5 | sudo ./upgrade_tool di -misc rockdev/misc.img
6 | sudo ./upgrade_tool di -b rockdev/boot.img
7 | sudo ./upgrade_tool di -recovery rockdev/recovery.img
8 | sudo ./upgrade_tool di -oem rockdev/oem.img
9 | sudo ./upgrade_tool di -rootfs rockdev/rootfs.img
10 | sudo ./upgrade_tool di -userdata rockdev/userdata.img
11 | sudo ./upgrade_tool rd
```

或升级整个 firmware 的 update.img 固件：

```
1 | sudo ./upgrade_tool uf rockdev/update.img
```

或在根目录，机器在 MASKROM 状态运行如下升级：

```
1 | ./rkflash.sh
```

8.3 系统分区说明

默认分区说明 (下面是 RK3399Pro EVB 分区参考)

Number	Start (sector)	End (sector)	Size	Name
1	16384	24575	4096K	uboot
2	24576	32767	4096K	trust
3	32768	40959	4096K	misc
4	40960	106495	32M	boot
5	106496	303104	96M	recovery
6	303104	368639	32M	bakcup
7	368640	499711	64M	oem
8	499712	13082623	6144M	rootfs
9	12082624	30535646	8521M	userdata

- uboot分区: 供 uboot 编译出来的 uboot.img。
- trust 分区: 供 uboot 编译出来的 trust.img。
- misc 分区: 供 misc.img, 给 recovery 使用。
- boot 分区: 供 kernel 编译出来的 boot.img。
- recovery 分区: 供 recovery 编译出的 recovery.img。
- backup 分区: 预留, 暂时没有用, 后续跟 Android 一样作为 recovery 的 backup 使用。
- oem 分区: 给厂家使用, 存放厂家的 APP 或数据。挂载在 /oem 目录。
- rootfs 分区: 供 buildroot、debian 或 yocto 编出来的 rootfs.img。
- userdata 分区: 供 APP 临时生成文件或给最终用户使用, 挂载在 /userdata 目录下。

9. RK3399Pro SDK 固件及简单 Demo 测试

9.1 RK3399Pro SDK 固件

RK3399PRO_LINUX_SDK_V1.3.0_20200324 固件下载链接如下 (包含 Buildroot/Debian 9/Yocto 的固件)

Buildroot: [V10\(绿色\)板子](#) [V11/V12\(黑色\)板子](#) [V13\(黑色\)板子](#) [V14\(黑色\)板子](#)

Debian 9: [适配所有板子](#)

Yocto: [适配所有板子](#)

9.2 RKNN_DEMO 测试

首先插入 usb camera, 然后在 Buildroot 系统中运行 rknn_demo 或 Debian 系统中运行 test_rknn_demo.sh。具体参考工程文档

docs/Soc_public/RK3399PRO/Rockchip_Developer_Guide_Linux_RKNN_DEMO_CN.pdf。在 Buildroot中运行结果如下:

```

1 [root@rk3399pro:/]# rknn_demo
2 librga:RGA_GET_VERSION:3.02,3.020000
3 ctx=0x2e834c20,ctx->rgaFd=3
4 Rga built version:version:+2017-09-28 10:12:42
5 Success build
6 size = 12582988, g_bo.size = 13271040
7 size = 12582988, cur_bo->size = 13271040
8 size = 12582988, cur_bo->size = 13271040
9 ...
10 read model:/usr/share/rknn_demo/mobilenet_ssd.rknn, len:32002449
11 Please configure uvc...
12 D RKNNAPI: =====
13 D RKNNAPI: RKNN VERSION:
14 D RKNNAPI: API: 1.3.0 (933b767 build: 2019-11-27 14:43:32)
15 D RKNNAPI: DRV: 1.3.0 (c4f8c23 build: 2019-11-25 10:39:29)
16 D RKNNAPI: =====

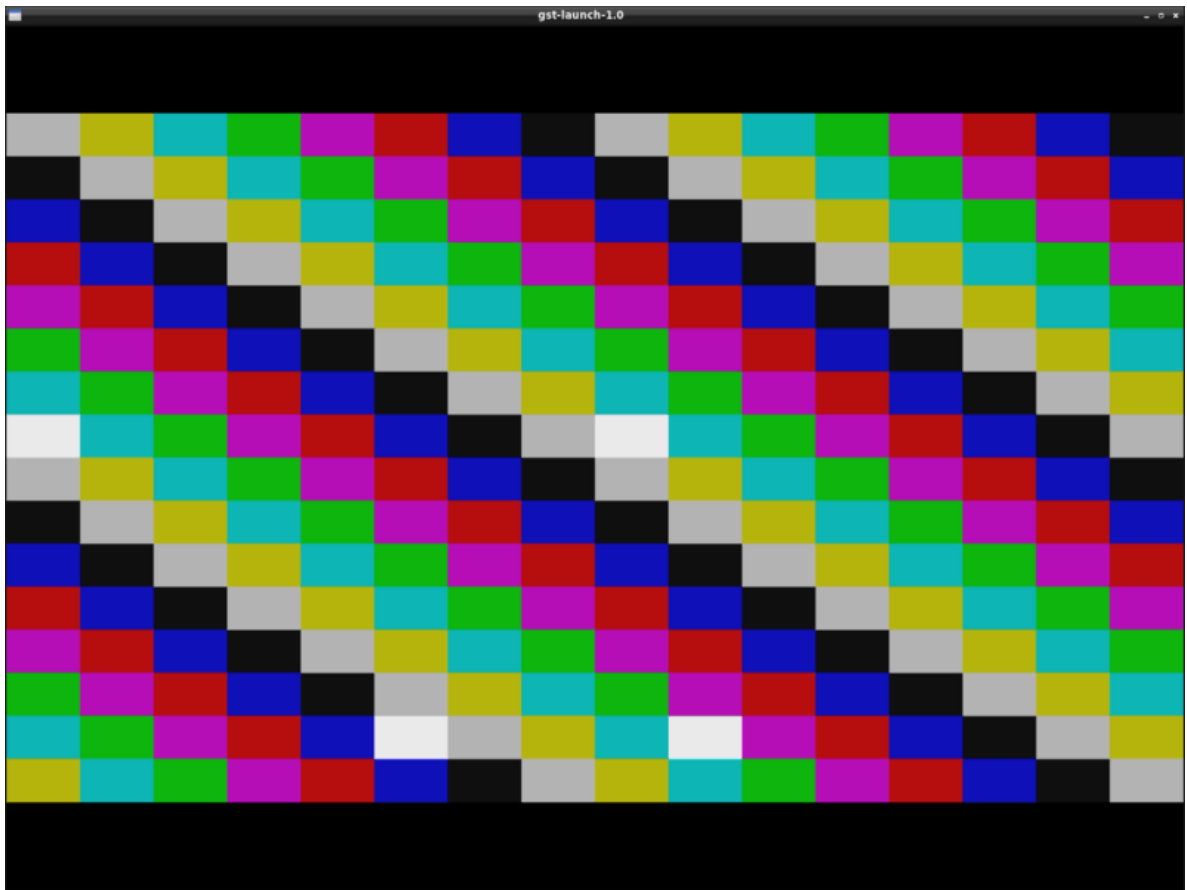
```

最终在屏幕显示效果如下：



9.3 N4 Camera 测试

首先接入 N4 Camera 模块（需要接入 12V 电源），然后在 Buildroot 系统中直接打开 Camera 应用或在 Debian 系统中运行 test_camera-rkisp1.sh。在 Buildroot 中运行结果如下：（没有接入具体 Camera Sensor）



10. SSH 公钥操作说明

请根据《Rockchip SDK 申请及同步指南》文档说明操作，生成 SSH 公钥，发邮件至fae@rock-chips.com，申请开通 SDK 代码。该文档会在申请开通权限流程中，释放给客户使用。

10.1 多台机器使用相同 SSH 公钥

在不同机器使用，可以将你的 SSH 私钥文件 `id_rsa` 拷贝到要使用的机器的“`~/.ssh/id_rsa`”即可。在使用错误的私钥会出现如下提示，请注意替换成正确的私钥

```
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
git@172.16.10.211's password: █
```

添加正确的私钥后，就可以使用 `git` 克隆代码，如下图。

```
~$ cd tmp/
~/tmp$ git clone git@172.16.10.211:rk292x/mid/4.1.1_r1
Initialized empty Git repository in /home/cody/tmp/4.1.1_r1/.git/
The authenticity of host '172.16.10.211 (172.16.10.211)' can't be established.
RSA key fingerprint is fe:36:dd:30:bb:83:73:e1:0b:df:90:e2:73:e4:61:46.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.211' (RSA) to the list of known hosts.
remote: Counting objects: 237923, done.
remote: Compressing objects: 100% (168382/168382), done.
Receiving objects: 9% (21570/237923), 61.52 MiB | 11.14 MiB/s
```

添加 ssh 私钥可能出现如下提示错误。

```
1 | Agent admitted failure to sign using the key
```

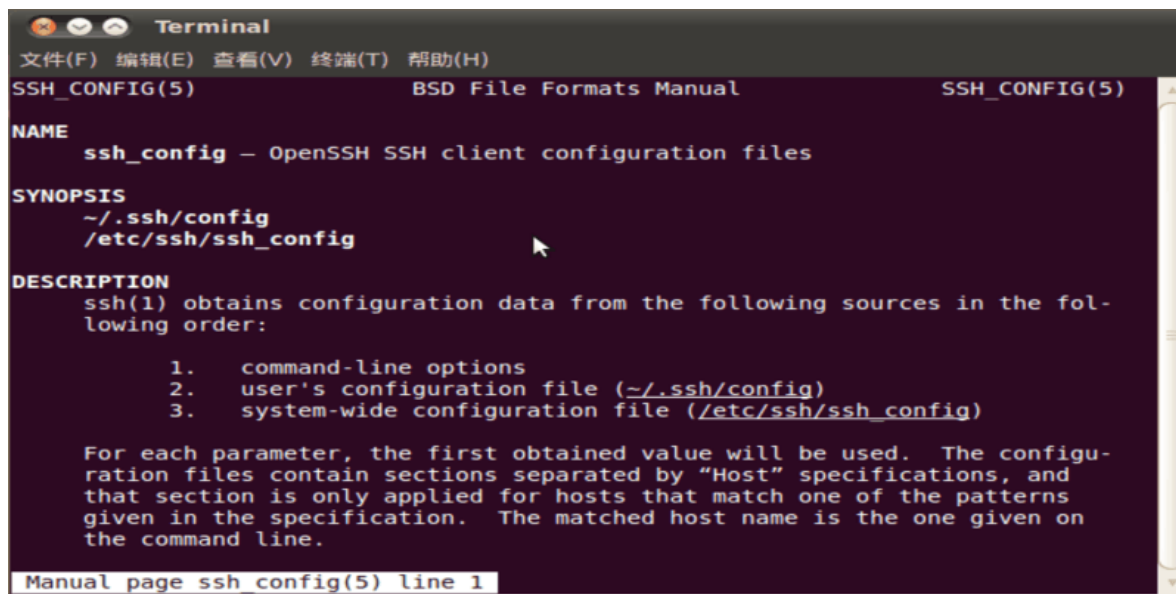
在 console 输入如下命令即可解决。

```
1 | ssh-add ~/.ssh/id_rsa
```

10.2 一台机器切换不同 SSH 公钥

可以参考 ssh_config 文档配置 SSH。

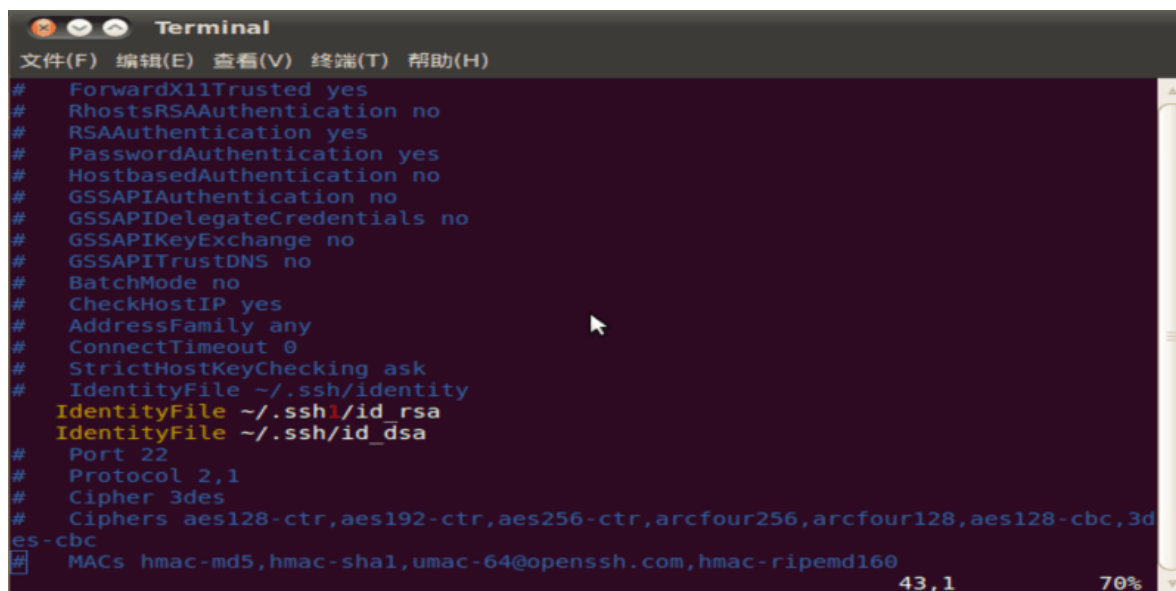
```
1 | ~$ man ssh_config
```



通过如下命令，配置当前用户的 SSH 配置。

```
1 | ~$ cp /etc/ssh/ssh_config ~/.ssh/config
2 | ~$ vi ~/.ssh/config
```

如图，将 SSH 使用另一个目录的文件“~/.ssh1/id_rsa”作为认证私钥。通过这种方法，可以切换不同的密钥。



10.3 密钥权限管理

服务器可以实时监控某个 key 的下载次数、IP 等信息，如果发现异常将禁用相应的 key 的下载权限。请妥善保管私钥文件。并不要二次授权与第三方使用。

10.4 参考文档

更多详细说明，可参考文档 [sdk/docs/RKTools manuals/Rockchip SDK Kit 申请指南 V1.6- 201905.pdf](#)。