

Rockchip RK3566_RK3568 Linux SDK 快速入门

文档标识: RK-FB-YF-942

发布版本: V1.5.0

日期: 2024-06-20

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，由其各自拥有者所有。

版权所有© 2024 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

本文主要描述了RK3566/RK3568 Linux SDK的基本使用方法，旨在帮助开发者快速了解并使用RK3566/RK3568 SDK开发包。

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 | Uboot版本 | Kernel版本 | Debian版本 | Buildroot版本 |
|---------------|---------|----------|----------|-----------------|
| RK3566、RK3568 | 2017.9 | 5.10、6.1 | 11、12 | 2021.11、2024.02 |

修订记录

| 日期 | 版本 | 作者 | 修改说明 |
|------------|--------|-------------|----------------|
| 2022-06-20 | V1.0.0 | Caesar Wang | 初始版本。 |
| 2022-09-20 | V1.0.1 | Caesar Wang | 增加Linux5.10支持。 |
| 2022-11-20 | V1.0.2 | Caesar Wang | 更新刷机说明。 |
| 2023-04-20 | V1.1.0 | Caesar Wang | 适配新版本SDK。 |
| 2023-05-20 | V1.1.1 | Caesar Wang | 更新SDK到V1.1.1。 |
| 2023-06-20 | V1.2.0 | Caesar Wang | 更新SDK到V1.2.0。 |
| 2023-09-20 | V1.3.0 | Caesar Wang | 更新SDK到V1.3.0。 |
| 2023-12-20 | V1.4.0 | Caesar Wang | 更新SDK到V1.4.0。 |
| 2024-06-20 | V1.5.0 | Caesar Wang | 更新SDK到V1.5.0。 |

目录

Rockchip RK3566_RK3568 Linux SDK 快速入门

1. SDK预编译镜像
2. 开发环境搭建
 - 2.1 准备开发环境
 - 2.2 安装库和工具集
 - 2.2.1 检查和升级主机的 `python` 版本
 - 2.2.2 检查和升级主机的 `make` 版本
 - 2.2.3 检查和升级主机的 `lz4` 版本
3. Docker环境搭建
4. 软件开发指南
 - 4.1 开发向导
 - 4.2 芯片资料
 - 4.3 Buildroot开发指南
 - 4.4 Debian开发指南
 - 4.5 第三方OS移植
 - 4.6 RKNPU开发指南
 - 4.7 DPDK开发指南
 - 4.8 实时性Linux开发说明
 - 4.8.1 压力测试
 - 4.8.1.1 PREEMPT_RT Patch
 - 4.8.1.2 Xenomai Cobalt Mode
 - 4.9 软件更新记录
 5. 硬件开发指南
 6. IO电源设计注意事项
 7. SDK 配置框架说明
 - 7.1 SDK 工程目录介绍
 8. SDK交叉编译工具链介绍
 - 8.1 U-Boot 及Kernel编译工具链
 - 8.2 Buildroot工具链
 - 8.2.1 配置编译环境
 - 8.2.2 打包工具链
 - 8.3 Debian工具链
 - 8.4 Yocto工具链
 9. SDK 编译说明
 - 9.1 SDK编译命令查看
 - 9.2 SDK板级配置
 - 9.3 SDK定制化配置
 - 9.4 全自动编译
 - 9.5 各模块编译
 - 9.5.1 U-Boot编译
 - 9.5.2 Kernel编译
 - 9.5.3 Recovery编译
 - 9.5.4 Buildroot编译
 - 9.5.4.1 Buildroot模块编译
 - 9.5.5 Debian编译
 - 9.5.6 Yocto 编译
 - 9.6 固件的打包
 10. 刷机说明
 - 10.1 Windows 刷机说明
 - 10.2 Linux 刷机说明
 - 10.3 系统分区说明

1. SDK预编译镜像

使用RK3566_RK3568 Linux SDK预编译镜像，开发人员可以省去从源代码编译整个操作系统的过 程，直接将预编译的镜像刷入RK3566_RK3568开发板，从而快速启动开发和进行相关评估、对比，可减少因编译问题导致的开发时间和成本浪费。

可从公开地址下载 [SDK固件下载点击这里](#)

Linux5.10固件路径：通用 **Linux SDK 固件-> Linux5.10 -> RK3566_RK3568**

Linux6.1固件路径：通用 **Linux SDK 固件-> Linux6.1 -> RK3566_RK3568**

如果需要修改SDK代码或快速入门，请参考下面章节。

2. 开发环境搭建

2.1 准备开发环境

我们推荐使用 Ubuntu 22.04 或更高版本的系统进行编译。其他的 Linux 版本可能需要对软件包做相应调整。除了系统要求外，还有其他软硬件方面的要求。

硬件要求：64 位系统，硬盘空间大于 40G。如果您进行多个构建，将需要更大的硬盘空间。

软件要求：Ubuntu 22.04 或更高版本系统。

2.2 安装库和工具集

使用命令行进行设备开发时，可以通过以下步骤安装编译SDK需要的库和工具。

使用如下apt-get命令安装后续操作所需的库和工具：

```
sudo apt-get update && sudo apt-get install git ssh make gcc libssl-dev \
liblz4-tool expect expect-dev g++ patchelf chrpath gawk texinfo chrpath \
diffstat binfmt-support qemu-user-static live-build bison flex fakeroot \
cmake gcc-multilib g++-multilib unzip device-tree-compiler ncurses-dev \
libgucharmap-2-90-dev bzip2 expat gpgv2 cpp-aarch64-linux-gnu libgmp-dev \
libmpc-dev bc python-is-python3 python2
```

说明：

安装命令适用于Ubuntu22.04，其他版本请根据安装包名称采用对应的安装命令，若编译遇到报错，可以视报错信息，安装对应的软件包。其中：

- python要求安装python 3.6及以上版本，此处以python 3.6为例。
- make要求安装 make 4.0及以上版本，此处以 make 4.2为例。
- lz4要求安装 lz4 1.7.3及以上版本。
- 编译Yocto需要VPN网络。

2.2.1 检查和升级主机的 `python` 版本

检查和升级主机的 `python` 版本方法如下：

- 检查主机 `python` 版本

```
$ python3 --version
Python 3.10.6
```

如果不满足 `python>=3.6` 版本的要求，可通过如下方式升级：

- 升级 `python 3.6.15` 新版本

```
PYTHON3_VER=3.6.15
echo "wget
https://www.python.org/ftp/python/${PYTHON3_VER}/Python-${PYTHON3_VER}.tgz"
echo "tar xf Python-${PYTHON3_VER}.tgz"
echo "cd Python-${PYTHON3_VER}"
echo "sudo apt-get install libsqlite3-dev"
echo "./configure --enable-optimizations"
echo "sudo make install -j8"
```

2.2.2 检查和升级主机的 `make` 版本

检查和升级主机的 `make` 版本方法如下：

- 检查主机 `make` 版本

```
$ make -v
GNU Make 4.2
Built for x86_64-pc-linux-gnu
```

- 升级 `make 4.2` 新版本

```
$ sudo apt update && sudo apt install -y autoconf autopoint

git clone https://gitee.com/mirrors/make.git
cd make
git checkout 4.2
git am $BUILDROOT_DIR/package/make/*.patch
autoreconf -f -i
./configure
make make -j8
sudo install -m 0755 make /usr/bin/make
```

2.2.3 检查和升级主机的 `lz4` 版本

检查和升级主机的 `lz4` 版本方法如下：

- 检查主机 `lz4` 版本

```
$ lz4 -v
*** LZ4 command line interface 64-bits v1.9.3, by Yann Collet ***
```

- 升级 lz4 新版本

```
git clone https://gitee.com/mirrors/LZ4_old1.git
cd LZ4_old1

make
sudo make install
sudo install -m 0755 lz4 /usr/bin/lz4
```

3. Docker环境搭建

为帮助开发者快速完成上面复杂的开发环境准备工作，我们也提供了交叉编译器Docker镜像，以便客户可以快速验证，从而缩短编译环境的构建时间。

使用Docker环境前，可参考如下文档进行操作

`<SDK>/docs/cn/Linux/Docker/Rockchip_Developer_Guide_Linux_Docker_Deploy_CN.pdf`。

已验证的系统如下：

| 发行版本 | Docker 版本 | 镜像加载 | 固件编译 |
|--------------|-----------|------|--------------|
| ubuntu 22.04 | 24.0.5 | pass | pass |
| ubuntu 21.10 | 20.10.12 | pass | pass |
| ubuntu 21.04 | 20.10.7 | pass | pass |
| ubuntu 18.04 | 20.10.7 | pass | pass |
| fedora35 | 20.10.12 | pass | NR (not run) |

Docker镜像可从网址 [Docker镜像](#) 获取。

4. 软件开发指南

4.1 开发向导

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_CN.pdf》，可在 `docs/cn/RK3566_RK3568` 目录下获取，并会不断完善更新

4.2 芯片资料

为帮助开发工程师更快上手熟悉 RK3566、RK3568 的开发调试工作，随 SDK 发布《Rockchip_RK3566_Datasheet_V1.4-20240621.pdf》和《Rockchip_RK3568_Datasheet_V2.1-20240621.pdf》芯片手册。可在 `docs/cn/RK3566_RK3568/Datasheet` 目录下获取，并会不断完善更新。

4.3 Buildroot开发指南

为帮助开发工程师更快上手熟悉Buildroot系统的开发调试，随 SDK 发布《Rockchip_Developer_Guide_Buildroot_CN.pdf》开发指南，可在 `docs/cn/Linux/System` 目录下获取，并会不断完善更新。

4.4 Debian开发指南

为帮助开发工程师更快上手熟悉 RK3566、RK3568 Debian的开发调试，随 SDK 发布《Rockchip_Developer_Guide_Debian_CN.pdf》开发指南，可在 `docs/cn/Linux/System` 下获取，并会不断完善更新。

4.5 第三方OS移植

为帮助开发工程师更快上手熟悉 RK3566、RK3568 第三方OS的移植适配，随 SDK 发布《Rockchip_Developer_Guide_Third_Party_System_Adaptation_CN.pdf》开发向导，可在 `docs/cn/Linux/System` 下获取，并会不断完善更新。

4.6 RKNPU开发指南

SDK提供了RKNPU相关开发工具，具体如下：

RKNN-TOOLKIT2：

RKNN-Toolkit2是在PC上进行RKNN模型生成及评估的开发套件：

开发套件在 `external/rknn-toolkit2` 目录下，主要用来实现模型转换、优化、量化、推理、性能评估和精度分析等一系列功能。

基本功能如下：

| 功能 | 说明 |
|------|---------------------------------------------------------------------------------------------------------------------------------------------|
| 模型转换 | 支持Pytorch / TensorFlow / TFLite / ONNX / Caffe / Darknet的浮点模型 支持Pytorch / TensorFlow / TFLite的量化感知模型（QAT） 支持动态输入模型（动态化/原生动态） 支持大模型 |
| 模型优化 | 常量折叠/ OP矫正/ OP Fuse&Convert / 权重稀疏化/ 模型剪枝 |
| 模型量化 | 支持量化类型：非对称i8/ fp16 支持Layer / Channel量化方式； Normal / KL/ MMSE量化算法 支持混合量化以平衡性能和精度 |
| 模型推理 | 支持在PC上通过模拟器进行模型推理 支持将模型传到NPU硬件平台上完成模型推理（连板推理） 支持批量推理，支持多输入模型 |
| 模型评估 | 支持模型在NPU硬件平台上的性能和内存评估 |
| 精度分析 | 支持量化精度分析功能（模拟器/ NPU） |
| 附加功能 | 支持版本/设备查询功能等 |

具体使用说明请参考当前 `doc/` 的目录文档：

```

├── 01_Rockchip_RKNPU_Quick_Start_RKNN_SDK_V2.0.0beta0_CN.pdf
├── 01_Rockchip_RKNPU_Quick_Start_RKNN_SDK_V2.0.0beta0_EN.pdf
...
├── RKNNToolKit2_API_Difference_With_Toolkit1-V2.0.0beta0.md
└── RKNNToolKit2_OP_Support-v2.0.0-beta0.md

```

RKNN API:

RKNN API的开发说明在工程目录 `external/rknpu2` 下，用于推理RKNN-Toolkit2生成的rknn模型。

具体使用说明请参考当前 `doc/` 的目录文档：

```

...
├── 02_Rockchip_RKNPU_User_Guide_RKNN_SDK_V2.0.0beta0_CN.pdf
├── 02_Rockchip_RKNPU_User_Guide_RKNN_SDK_V2.0.0beta0_EN.pdf
├── 03_Rockchip_RKNPU_API_Reference_RKNN_Toolkit2_V2.0.0beta0_CN.pdf
├── 03_Rockchip_RKNPU_API_Reference_RKNN_Toolkit2_V2.0.0beta0_EN.pdf
├── 04_Rockchip_RKNPU_API_Reference_RKNNRT_V2.0.0beta0_CN.pdf
└── 04_Rockchip_RKNPU_API_Reference_RKNNRT_V2.0.0beta0_EN.pdf

```

4.7 DPDK开发指南

为帮助开发工程师更快上手熟悉DPDK的开发调试，随 SDK 发布

《Rockchip_Developer_Guide_Linux_DPDK_CN.pdf》和
《Rockchip_Developer_Guide_Linux_GMAC_DPDK_CN.pdf》

开发向导，分别可在 `<SDK>/external/dpdk/rk_docs` 目录下获取，并会不断完善更新。

4.8 实时性Linux开发说明

随着产品对实时性能要求的提高，标准Linux的实时性已经满足不了很多产品的需求，需要对标准Linux进行一定的优化，来提高实时性能，比如：PREEMPT_RT，Xenomai实时系统。

下面是进行是基于RK3566_RK3568 Buildroot上分别基于PREEMPT_RT，Xenomai进行压力测试的延迟情况，具体如下：

4.8.1 压力测试

```
stress-ng -c 4 --io 2 --vm 1 --vm-bytes 4M --timeout 1000000s
```

4.8.1.1 PREEMPT_RT Patch

```
rk3568_t:/ # /data/cyclictest -c 0 -m -n -t 4 -p 99
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 9.36 9.67 9.84 8/1152 3678

T: 0 ( 3482) P:99 I:1000 C:5892687 Min: 9 Act: 20 Avg: 25 Max: 123
T: 1 ( 3483) P:99 I:1500 C:3928444 Min: 9 Act: 20 Avg: 25 Max: 116
T: 2 ( 3484) P:99 I:2000 C:2946323 Min: 9 Act: 13 Avg: 26 Max: 120
T: 3 ( 3485) P:99 I:2500 C:2357050 Min: 10 Act: 20 Avg: 26 Max: 126
(测试2小时)
```

4.8.1.2 Xenomai Cobalt Mode

```
rk3568_t:/ # /data/cyclictest -c 0 -m -n -t 4 -p 99
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 7.65 7.80 7.75 8/1118 3998

T: 0 ( 3492) P:99 I:1000 C:21069118 Min: 2 Act: 4 Avg: 13 Max: 85
T: 1 ( 3493) P:99 I:1500 C:14046070 Min: 2 Act: 13 Avg: 15 Max: 76
T: 2 ( 3494) P:99 I:2000 C:10534550 Min: 2 Act: 6 Avg: 16 Max: 70
T: 3 ( 3495) P:99 I:2500 C:8427637 Min: 2 Act: 15 Avg: 15 Max: 96
(测试5小时)
```

详情请参考 [docs/Patches/Real-Time-Performance/](#) 开发补丁包和说明。

4.9 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：

```
.repo/manifests$ realpath rk356x_linux5.10_release.xml
# 例如：打印的版本号为v1.5.0，更新时间为20240620
# <SDK>/.repo/manifests/rk356x_linux/rk356x_linux5.10_release_v1.5.0_20240620.xml
```

- 软件发布版本升级更新内容通过工程文本可以查看，参考工程目录：

```
<SDK>/.repo/manifests/rk356x_linux/RK3566_RK3568_Linux5.10_SDK_Note.md
```

或者

```
<SDK>/docs/cn/RK3566_RK3568/RK3566_RK3568_Linux5.10_SDK_Note.md
```

- 板端可通过如下方式获取版本信息

```
buildroot:/# cat /etc/os-release
NAME=Buildroot
VERSION=linux-5.10-gen-rkr8
ID=buildroot
VERSION_ID=2021.11
PRETTY_NAME="Buildroot 2021.11"
OS="buildroot"
BUILD_INFO="xxx Thur Jun 20 23:12:04 CST 2024 - rockchip_rk3568"
KERNEL="5.10 - rockchip_linux_defconfig"
```

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：

RK3566_RK3568 硬件设计指南：

```
<SDK>/docs/cn/RK3566_RK3568/Hardware/Rockchip_RK3566_Hardware_Design_Guide_V1.1_2
0220206_CN.pdf
<SDK>/docs/cn/RK3566_RK3568/Hardware/Rockchip_RK3568_Hardware_Design_Guide_V1.2_2
0211107_CN.pdf
```

RK3566_RK3568 EVB 硬件用户指南：

```
<SDK>/docs/cn/RK3566_RK3568/Hardware/Rockchip_RK3568_EVB_User_Guide_V1.2_CN.pdf
<SDK>/docs/cn/RK3566_RK3568/Hardware/Rockchip_RK3566_EVB2_User_Guide_V1.1_CN.pdf
```

6. IO电源设计注意事项

主控电源域的IO电平要与对接外设芯片的IO电平保持一致，还要注意软件的电压配置要跟硬件的电压一致，否则可能会导致GPIO的损坏。



注意

关于GPIO电源域IO电平匹配问题：

GPIO的电源域PMUIO0_VDD, PMUIO1_VDD, VCCIO1_VDD, VCCIO2_VDD, VCCIO3_VDD, VCCIO4_VDD, VCCIO5_VDD, VCCIO6_VDD, VCCIO7_VDD, 这些电源的电压要跟所接的外设的IO电平的电压保持一致，否则可能会导致GPIO的损坏。

还要注意软件的电压配置要跟硬件的电压一致：比如硬件IO电平接1.8V，软件的电压配置也要相应的配成1.8V；硬件IO电平接3.3V，软件的电压配置也要用3.3V，否则也可能导致GPIO的损坏。

更多信息参考：

```
<SDK>/docs/cn/RK3566_RK3568/Rockchip_RK356X_Introduction_IO_Power_Domains_Configuration.pdf  
<SDK>/docs/cn/Common/IO-DOMAIN/Rockchip_Developer_Guide_Linux_IO_DOMAIN_CN.pdf
```

7. SDK 配置框架说明

7.1 SDK 工程目录介绍

SDK 工程目录包含有 buildroot、debian、rtos、app、kernel、u-boot、device、docs、external 等目录。采用manifest来管理仓库，用repo工具来管理每个目录或其子目录会对应的 git 工程。

- app: 存放上层应用 APP，主要是一些应用Demo。
- buildroot: 基于 Buildroot (2021或2024) 开发的根文件系统。
- debian: 基于 Debian bullseye(11或12) 开发的根文件系统。
- device/rockchip: 存放芯片板级配置以及一些编译和打包固件的脚本和文件。
- 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。
- external: 存放第三方相关仓库，包括显示、音视频、摄像头、网络、安全等。
- kernel: 存放 Kernel 开发的代码。
- hal: 存放基于RK HAL硬件抽象层的裸机开发库，用于AMP方案。
- output: 存放每次生成的固件信息、编译信息、XML、主机环境等。
- prebuilts: 存放交叉编译工具链。
- rkbin: 存放 Rockchip 相关二进制和工具。
- rockdev: 存放编译输出固件,实际软链接到 output/firmware。
- rtos: 存放基于RT-Thread 4.1开发的根文件系统。
- 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。
- uefi: 存放基于 edk2 V2.7版本进行开发的 UEFI 代码。
- yocto: 存放基于 Yocto 4.0或5.0开发的根文件系统。

8. SDK交叉编译工具链介绍

鉴于Rockchip Linux SDK目前只在Linux PC环境下编译，我们也仅提供了Linux下的交叉编译工具链。prebuilt目录下的预置的工具链是给U-Boot和Kernel使用。具体Rootfs需要用各自对应的工具链，或者使用第三方工具链进行编译。

8.1 U-Boot 及 Kernel编译工具链

SDK prebuilts目录预置交叉编译目前用于U-Boot 及 Kernel编译，如下：

| 目录 | 说明 |
|------------------------------------------------------------------------------------|-----------------------|
| prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu | gcc arm 10.3.1 64位工具链 |
| prebuilts/gcc/linux-x86/arm/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf | gcc arm 10.3.1 32位工具链 |

可从以下地址下载工具链：

[点击这里](#)

8.2 Buildroot工具链

8.2.1 配置编译环境

若需要编译单个模块或者第三方应用，需交叉编译环境进行配置。比如RK3568其交叉编译工具位于
buildroot/output/rockchip_rk3568/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：

```
source buildroot/envsetup.sh rockchip_rk3568
```

输入命令查看：

```
cd buildroot/output/rockchip_rk3568/host/usr/bin  
.aarch64-linux-gcc --version
```

此时会打印如下信息：

```
aarch64-linux-gcc.br_real (Buildroot -gc307c95550) 12.3.0
```

8.2.2 打包工具链

Buildroot 支持将内置工具链打包为压缩包，以供第三方应用单独编译使用。有关如何打包工具链的详细信息，请参阅 Buildroot 官方文档：

在 SDK 中，可以直接运行以下命令来生成工具链包：

```
./build.sh bmake: sdk
```

生成的工具链包位于 `buildroot/output/*/images/` 目录，名为 `aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz`，供有需求的用户使用。解压后，`gcc` 的路径将是：

```
./aarch64-buildroot-linux-gnu_sdk-buildroot/bin/aarch64-buildroot-linux-gnu-gcc
```

8.3 Debian工具链

使用 docker 机器端，`gcc` 或者 `dpkg-buildpackage` 进行相关编译。

8.4 Yocto工具链

参考如下：

```
https://wiki.yoctoproject.org/wiki/Building_your_own_recipes_from_first_principle  
s#Adding_new_recipes_to_the_build_system  
https://docs.yoctoproject.org/dev/dev-manual/new-recipe.html
```

9. SDK 编译说明

SDK 可通过 `make` 或 `./build.sh` 加目标参数进行相关功能的配置和编译。

具体参考 `device/rockchip/common/README.md` 编译说明。

9.1 SDK 编译命令查看

`make help`，例如：

```
$ make help
  menuconfig          - interactive curses-based configurator
  oldconfig           - resolve any unresolved symbols in .config
  syncconfig          - Same as oldconfig, but quietly, additionally update
  deps
  olddefconfig        - Same as syncconfig but sets new symbols to their
  default value
  savedefconfig       - Save current config to RK_DEFCONFIG (minimal config)
  ...
```

`make` 实际运行是 `./build.sh`

即也可运行 `./build.sh <target>` 来编译相关功能，具体可通过 `./build.sh help` 查看具体编译命令。

```
#####
# Rockchip Linux SDK #####
#####

Manifest: rk356x_linux5.10_release_v1.5.0_20240620.xml

Log colors: message notice warning error fatal

Usage: build.sh [OPTIONS]
Available options:
chip[:<chip>[:<config>]]          choose chip
defconfig[:<config>]                  choose defconfig
config                                modify SDK defconfig
...
updateimg                            build update image
otapackage                           build A/B OTA update image
all                                    build images
release                               release images and build info
save                                   alias of release
all-release                           build and release images
allsave                               alias of all-release
shell                                 setup a shell for developing
cleanall                             cleanup
clean[:module[:module]]...           cleanup modules
post-rootfs <rootfs dir>           trigger post-rootfs hook scripts
help                                 usage

Default option is 'all'.
```

9.2 SDK板级配置

进入工程 `<SDK>/device/rockchip/rk3566_rk3568` 目录:

| 板级配置 | 说明 |
|-----------------------------------------------|------------------------------------------------|
| rockchip_rk3566_evb2_lp4x_v10_32bit_defconfig | 适用于 RK3566 EVB 搭配 LPDDR4 开发板，运行32位根文件系统 |
| rockchip_rk3566_evb2_lp4x_v10_defconfig | 适用于 RK3566 EVB 搭配 LPDDR4 开发板 |
| rockchip_rk3568_evb1_ddr4_v10_32bit_defconfig | 适用于 RK3568 EVB 搭配 LPDDR4 开发板，运行32位根文件系统 |
| rockchip_rk3568_evb1_ddr4_v10_defconfig | 适用于 RK3568 EVB 搭配 DDR4 开发板 |
| rockchip_rk3568_evb8_lp4_v10_32bit_defconfig | 适用于 RK3568 EVB 搭配 LPDDR4/RK860X 开发板，运行32位根文件系统 |
| rockchip_rk3568_evb8_lp4_v10_defconfig | 适用于 RK3568 EVB8 搭配 LPDDR4/RK860X 开发板 |
| rockchip_rk3568_pcie_ep_lp4x_v10_defconfig | 适用于RK3568 PCIe EP标准卡开发验证 |
| rockchip_defconfig | 默认配置, 具体会软链接到默认一个板级配置 |

方法1

`./build.sh` 后面加上板级配置文件，例如：

选择**RK3566 EVB** 搭配 **LPDDR4** 开发板，运行32位根文件系统的板级配置：

```
./build.sh  
device/rockchip/rk3566_rk3568/rockchip_rk3566_evb2_lp4x_v10_32bit_defconfig
```

选择**RK3566 EVB** 搭配 **LPDDR4** 开发板的板级配置：

```
./build.sh device/rockchip/rk3566_rk3568/rockchip_rk3566_evb2_lp4x_v10_defconfig
```

选择**RK3568 EVB** 搭配 **DDR4** 开发板，运行32位根文件系统的板级配置：

```
./build.sh  
device/rockchip/rk3566_rk3568/rockchip_rk3568_evb1_ddr4_v10_32bit_defconfig
```

选择**RK3568 EVB** 搭配 **DDR4** 开发板的板级配置：

```
./build.sh device/rockchip/rk3566_rk3568/rockchip_rk3568_evb1_ddr4_v10_defconfig
```

选择**RK3568 EVB** 搭配 **LPDDR4/RK860X**开发板的板级配置：

```
./build.sh device/rockchip/rk3566_rk3568/rockchip_rk3568_evb8_lp4_v10_defconfig
```

选择**RK3568 EVB** 搭配 **LPDDR4/RK860X**开发板，运行32位根文件系统的板级配置：

```
./build.sh  
device/rockchip/rk3566_rk3568/rockchip_rk3568_evb8_lp4_v10_32bit_defconfig
```

选择**RK3568 PCIe EP**标准卡开发验证的板级配置：

```
./build.sh  
device/rockchip/rk3566_rk3568/rockchip_rk3568_pcie_ep_lp4x_v10_defconfig
```

方法2

```
rk3566_rk3568$ ./build.sh lunch  
Pick a defconfig:  
1. rockchip_defconfig  
2. rockchip_rk3566_evb2_lp4x_v10_32bit_defconfig  
3. rockchip_rk3566_evb2_lp4x_v10_defconfig  
4. rockchip_rk3568_evb1_ddr4_v10_32bit_defconfig  
5. rockchip_rk3568_evb1_ddr4_v10_defconfig  
6. rockchip_rk3568_evb8_lp4_v10_32bit_defconfig  
7. rockchip_rk3568_evb8_lp4_v10_defconfig  
8. rockchip_rk3568_pcie_ep_lp4x_v10_defconfig  
9. rockchip_rk3568_uvc_evb1_ddr4_v10_defconfig  
Which would you like? [1]:
```

9.3 SDK定制化配置

SDK可通过 `make menuconfig` 进行相关配置，目前可配组件主要如下：

```
(rockchip_rk3568_evb1_ddr4_v10_defconfig) Name of defconfig to save
[*] Rootfs (Buildroot|Debian|Yocto) --->
[*] Loader (U-Boot) --->
[ ] AMP (Asymmetric Multi-Processing System)
[*] Kernel (Embedded in an Android-style boot image) --->
    Boot (Android-style boot image) --->
[*] Recovery (based on Buildroot) --->
    *** Security feature depends on buildroot rootfs ***
    Extra partitions (oem, userdata, etc.) --->
    Firmware (partition table, misc image, etc.) --->
[*] Update (Rockchip update image) --->
    Others configurations --->
```

- Rootfs: 这里的Rootfs代表“根文件系统”，在这里可选择Buildroot、Yocto、Debian等不同的根文件系统配置。
- Loader (u-boot): 这是引导加载器的配置，通常是u-boot，用于初始化硬件并加载主操作系统。
- AMP: 多核异构启动方案，适用于需要实时性能的应用场景。
- Kernel: 这里配置内核选项，定制适合自己的硬件和应用需求的Linux内核。
- Boot: 这里配置Boot分区支持格式。
- Recovery (based on buildroot): 这是基于buildroot的recovery环境的配置，用于系统恢复和升级。
- PCBA test (based on buildroot): 这是一个基于buildroot的PCBA（印刷电路板组装）测试环境的配置。
- Security: 安全功能开启，包含Secureboot开启方式、Optee存储方式、烧写Key等。
- Extra partitions: 用于配置额外的分区。
- Firmware: 在这里配置固件相关选项。
- Update (Rockchip update image): 用于配置Rockchip完整固件的选项。
- Others configurations: 其他额外的配置选项。

通过 `make menuconfig` 配置界面提供了一个基于文本的用户界面来选择和配置各种选项。

配置完成后，使用 `make savedefconfig` 命令保存这些配置，这样定制化编译就会根据这些设置来进行。

通过以上config，可选择不同Rootfs/Loader/Kernel等配置，进行各种定制化编译，这样就可以灵活地选择和配置系统组件以满足特定的需求。

9.4 全自动编译

进入工程根目录执行以下命令自动完成所有的编译：

```
./build.sh all # 只编译模块代码 (u-Boot, kernel, Rootfs, Recovery)
                # 需要再执行`./build.sh ./mkfirmware.sh`进行固件打包

./build.sh      # 编译模块代码 (u-Boot, kernel, Rootfs, Recovery)
                # 打包成update.img完整升级包
                # 所有编译信息复制和生成到out目录下
```

默认是 Buildroot，可以通过设置环境变量 `RK_ROOTFS_SYSTEM` 指定不同 rootfs。`RK_ROOTFS_SYSTEM` 目前可设定三种系统：buildroot、debian、yocto。

比如需要 debain 可以通过以下命令进行生成：

```
export RK_ROOTFS_SYSTEM=debian
./build.sh
或
RK_ROOTFS_SYSTEM=debian ./build.sh
```

注意：

SDK每次更新建议都清理之前的编译产物，直接运行 `./build.sh cleanall` 即可。

9.5 各模块编译

9.5.1 U-Boot编译

```
./build.sh uboot
```

9.5.2 Kernel编译

- 方法一

```
./build.sh kernel
```

- 方法二

```
cd kernel
export CROSS_COMPILE=../prebuilts/gcc/linux-x86/aarch64/gcc-arm-10.3-2021.07-
x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu-
make ARCH=arm64 rockchip_linux_defconfig

make ARCH=arm64 rk3566-evb2-1p4x-v10-linux.img -j16
或
make ARCH=arm64 rk3568-evb1-ddr4-v10-linux.img -j16
或
make ARCH=arm64 rk3568-evb8-1p4-v10-linux.img -j16
```

- 方法三

```
cd kernel
export CROSS_COMPILE=aarch64-linux-gnu-
make ARCH=arm64 rockchip_linux_defconfig

make ARCH=arm64 rk3566-evb2-1p4x-v10-linux.img -j16
或
make ARCH=arm64 rk3568-evb1-ddr4-v10-linux.img -j16
或
make ARCH=arm64 rk3568-evb8-1p4-v10-linux.img -j16
```

9.5.3 Recovery编译

进入工程根目录执行以下命令自动完成 Recovery 的编译及打包。

```
<SDK># ./build.sh recovery
```

编译后在 Buildroot 目录 `output/rockchip_rk3568_recovery/images` 生成 `recovery.img`。

注: `recovery.img` 是包含内核, 所以每次 Kernel 更改, Recovery 是需要重新打包生成。Recovery 重新打包的方法如下:

```
<SDK># source buildroot/envsetup.sh
<SDK># cd buildroot
<SDK># make recovery-reconfigure
<SDK># cd -
<SDK># ./build.sh recovery
```

注: Recovery是非必需的功能, 有些板级配置不会设置

9.5.4 Buildroot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:

```
./build.sh rootfs
```

编译后在Buildroot目录 `output/rockchip_rk3568/images` 下生成不同格式的镜像, 默认使用 `rootfs.ext4` 格式。

具体可参考Buildroot开发文档参考:

```
<SDK>/docs/cn/Linux/System/Rockchip_Developer_Guide_Buildroot_CN.pdf
```

9.5.4.1 Buildroot模块编译

可通过 `source buildroot/envsetup.sh` 来设置不同芯片和目标功能的配置

```
$ source buildroot/envsetup.sh
Top of tree: rk3566_rk3568

You're building on Linux
Lunch menu...pick a combo:

57. rockchip_rk3566
58. rockchip_rk3566_32
59. rockchip_rk3566_recovery
60. rockchip_rk3566_rk3568_base
61. rockchip_rk3566_rk3568_ramboot
62. rockchip_rk3568
63. rockchip_rk3568_32
64. rockchip_rk3568_recovery

...
```

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

默认选择57, `rockchip_rk3566`。然后进入RK3566的Buildroot目录, 开始相关模块的编译。

其中 `rockchip_rk3566_32` 是用来编译32位根文件系统, `rockchip_rk3566_recovery` 是用来编译 Recovery模块。

比如编译 `rockchip-test` 模块, 常用相关编译命令如下:

进入 buildroot 目录

```
<SDK># cd buildroot
```

- 删除并重新编译 `rockchip-test`

```
buildroot# make rockchip-test-recreate
```

- 重编 `rockchip-test`

```
buildroot# make rockchip-test-rebuild
```

- 删除 `rockchip-test`

```
buildroot# make rockchip-test-dirclean
```

或者

```
buildroot# rm -rf output/rockchip_rk3566/build/rockchip-test-master/
```

9.5.5 Debian 编译

```
./build.sh debian
```

编译后在 `debian` 目录下生成 `linaro-rootfs.img`。

说明: 需要预先安装相关依赖包

```
sudo apt-get install binfmt-support qemu-user-static live-build
```

```
sudo dpkg -i ubuntu-build-service/packages/*
```

```
sudo apt-get install -f
```

具体可参考Debian开发文档参考:

```
<SDK>/docs/cn/Linux/System/Rockchip_Developer_Guide_Debian_CN.pdf
```

9.5.6 Yocto 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:

```
./build.sh yocto
```

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

默认用户名登录是 `root`。Yocto 更多信息请参考 [Rockchip Wiki](#)。

FAQ:

- 上面编译如果遇到如下问题情况:

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

解决方法:

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

或者参考 [setup-locale-python3](#)

- 如果遇到git权限问题，导致编译出错。

由于git新版本增加了CVE-2022-39253安全检测补丁，而如果旧版本的Yocto就需要poky包含如下才可修复:

```
commit ac3eb2418aa91e85c39560913c1ddfd2134555ba  
Author: Robert Yang <liezhi.yang@windriver.com>  
Date:   Fri Mar 24 00:09:02 2023 -0700  
  
bitbake: fetch/git: Fix local clone url to make it work with repo  
  
The "git clone /path/to/git/objects_symlink" couldn't work after the  
following  
change:  
  
https://github.com/git/git/commit/6f054f9fb3a501c35b55c65e547a244f14c38d56
```

或者通过回退PC的git版本到V2.38或早期版本也行，比如下:

```
$ sudo apt update && sudo apt install -y libcurl4-gnutls-dev  
  
git clone https://gitee.com/mirrors/git.git --depth 1 -b v2.38.0  
cd git  
make git -j8  
make install  
sudo install -m 0755 git /usr/bin/git
```

9.6 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 `output/firmware` 目录下:

固件生成:

```
./build.sh firmware
```

10. 刷机说明

RK3566 EVB2 接口分布图如下：



RK3568 EVB1 开发板接口分布图如下：



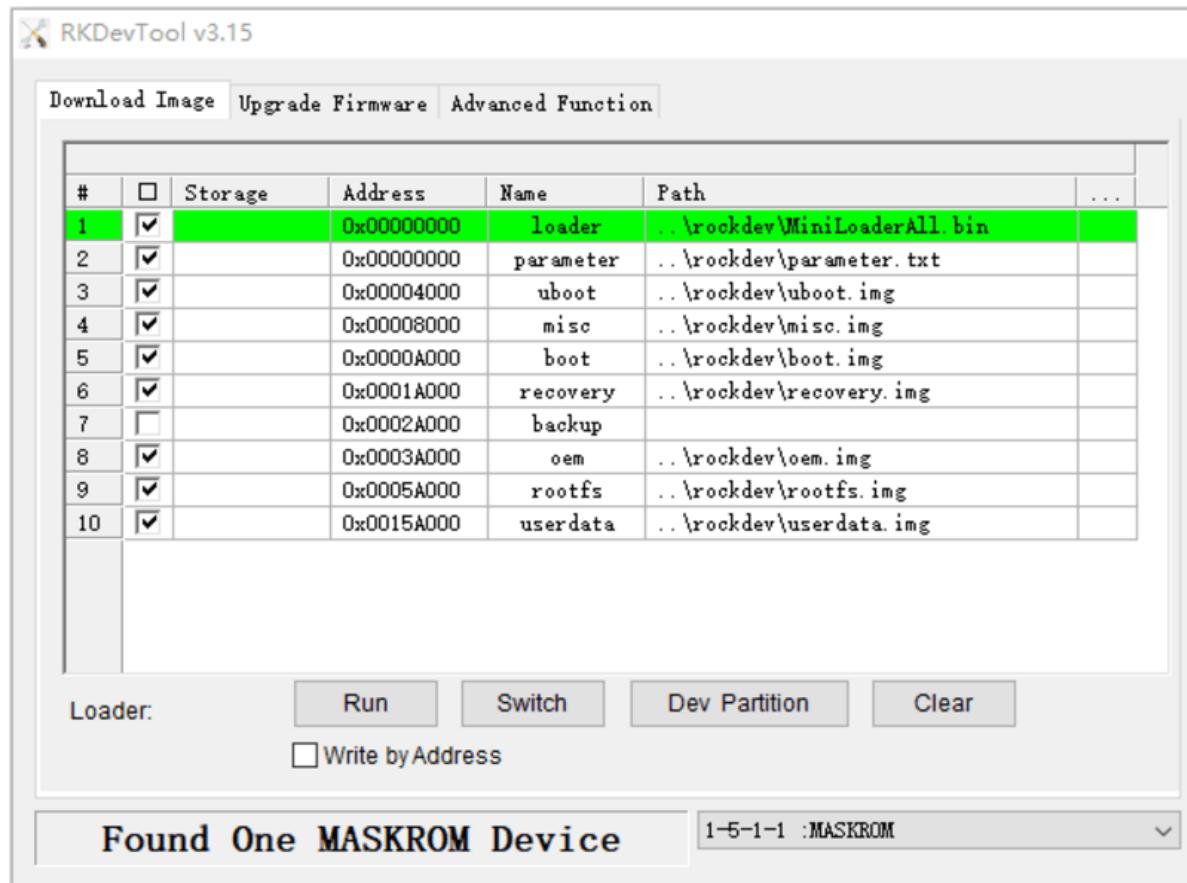
10.1 Windows 刷机说明

SDK 提供 Windows 烧写工具(工具版本需要 V3.31或以上), 工具位于工程根目录:

```
tools/  
└─ windows/RKDevTool
```

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： Windows 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：

<SDK>/tools/windows/DriverAssitant_v5.13.zip

10.2 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 V2.36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：

```
sudo ./upgrade_tool ul rockdev/MiniLoaderAll.bin -noreset
sudo ./upgrade_tool di -p rockdev/parameter.txt
sudo ./upgrade_tool di -u rockdev/uboot.img
sudo ./upgrade_tool di -misc rockdev/misc.img
sudo ./upgrade_tool di -b rockdev/boot.img
sudo ./upgrade_tool di -recovery rockdev/recovery.img
sudo ./upgrade_tool di -oem rockdev/oem.img
sudo ./upgrade_tool di -rootfs rockdev/rootfs.img
sudo ./upgrade_tool di -userdata rockdev/userdata.img
sudo ./upgrade_tool rd
```

或升级打包后的完整固件：

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

或在根目录，机器在 MASKROM 状态运行如下升级：

```
./rkflash.sh
```

10.3 系统分区说明

默认分区说明 (下面是 RK3568 EVB 分区参考)

| Number | Start (sector) | End (sector) | Size | Name |
|--------|----------------|--------------|-------|----------|
| 1 | 16384 | 24575 | 4M | uboot |
| 2 | 24576 | 32767 | 4M | misc |
| 3 | 32768 | 163839 | 64M | boot |
| 4 | 163840 | 425983 | 128M | recovery |
| 5 | 425984 | 491519 | 32M | bakcup |
| 6 | 491520 | 13074431 | 6144M | rootfs |
| 7 | 13074432 | 13336575 | 128M | oem |
| 8 | 13336576 | 61120478 | 22.7G | userdata |

- uboot 分区：供 uboot 编译出来的 uboot.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。
- rootfs 分区：供 buildroot、debian 或 yocto 编出来的 rootfs.img。
- oem 分区：给厂家使用，存放厂家的 APP 或数据。挂载在 /oem 目录。
- 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。