NVR SDK编译使用指南

文件标识：RK-SYS1-NONE

发布版本：V1.0.0

日　　期：2021.8

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。

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

Rockchip NVR SDK编译使用指南。

产品版本

芯片名称	内核版本
RK356X	4.19

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

修订记录

版本号	作者	修改日期	修改说明
V0.1.0	肖亚鹏	2021-4-1	初始版本
V1.0.0	肖亚鹏	2021-8-5	发布版本，完善文档增加FAQ

目录

介绍

基于 rk356x_linux_release_v1.1.3_20210805.xml 356x 完整linux SDK裁剪而来，针对类NVR产品优化了多视频播放能力等，

sdk下载地址:

```
repo init --repo-url ssh://git@www.rockchip.com.cn/repo/rk/tools/repo \
-u ssh://git@www.rockchip.com.cn/linux/rockchip/platform/manifests \
-b linux -m rk356x_nvr_linux_lite.xml

.repo/repo/repo sync -c --no-tags
```

如遇问题请参考`Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf`

开发包目录说明

SDK目录包含有 kernel、u-boot、tools、docs、rkbin 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。

```
- SDK
  -- 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
  -- kernel          //存放 Kernel 4.19 开发的代码。
  -- rkbin           //存放 Rockchip 相关 Binary 和工具
  -- 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。
  -- IMAGE           //存放每次生成编译时间、XML、补丁和固件目录。
  -- rockdev         //存放编译输出固件。
  -- build           //存放编译脚本、rootfs以及toolchain编译工具链。
```

SDK编译说明

根目录下有两编译脚本`build_emmc.sh`以及`build_spi_nand.sh`分别用于编译emmc以及spi nand设备。
两个脚本编译指令都相同下面以`build_emmc.sh`为列。

查看编译命令

在根目录执行命令：./build_emmc.sh -h|help

```
rk356x$ ./build_emmc.sh -h
============Start check sdk env ============
Running check_env succeeded.
processing option: --help
Usage: build.sh [OPTIONS]
uboot               -build uboot
kernel              -build kernel
rootfs              -build default rootfs, currently build buildroot as default
all                 -build uboot, kernel, rootfs image
cleanall            -clean uboot, kernel, rootfs
update              -pack update image
env                 -check sdk env

Default option is 'all'.
```

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_emmc.sh all # 编译模块代码(u-Boot，kernel，Rootfs),并进行固件打包

./build_emmc.sh #同上
```

各模块编译及打包

U-Boot编译

```
### U-Boot编译命令
./build_emmc.sh uboot

### U-Boot配置参数
emmc配置
# 默认的编译配置，不需要修改
export RK_UBOOT_DEFCONFIG=rk3568
# 默认即可不修改
export RK_UBOOT_FORMAT_TYPE=fit

spi nand配置
# 根据硬件是否带pmic选择配置：rk3568-spi-nand/rk3568-spi-nand-pmic
export RK_UBOOT_DEFCONFIG=rk3568-spi-nand
# 默认即可不修改
export RK_UBOOT_FORMAT_TYPE=no-fit
# 单个uboot大小，最终uboot.img大小 = 单个uboot * count
export RK_UBOOT_TOTAL_SIZE=1024
# uboot.img包含count个uboot，默认为2个用于备份。
export RK_UBOOT_BACKUP_COUNT=2
# 单个trust大小，最终trust.img大小 = 单个trust * count
export RK_TRUST_TOTAL_SIZE=1024
# trust.img包含count个trust，默认为2个用于备份。
export RK_TRUST_BACKUP_COUNT=2
```

Kernel编译

```
### Kernel编译命令
./build_emmc.sh kernel

### kernel配置参数
emmc配置
# kernel默认的配置
export RK_KERNEL_DEFCONFIG=rockchip_linux_defconfig
# kernel nvr产品形态配置
export RK_KERNEL_DEFCONFIG_FRAGMENT=rk3568_nvr.config
# kernel的dts根据具体版型修改：rk3568-nvr-demo-v10-linux rk3568-evb1-ddr4-v10-linux
rk3568-nvr-demo-v12-linux
export RK_KERNEL_DTS=rk3568-nvr-demo-v12-linux
# 默认即可不需要修改
export RK_BOOT_IMG=zboot.img

spi nand配置
# kernel默认的配置
export RK_KERNEL_DEFCONFIG=rockchip_linux_defconfig
# kernel nvr产品形态配置
export RK_KERNEL_DEFCONFIG_FRAGMENT=rk3568_nvr.config
# kernel的dts根据具体版型修改：rk3568-nvr-demo-v10-linux   rk3568-evb1-ddr4-v10-
linux rk3568-nvr-demo-v12-linux
export RK_KERNEL_DTS=rk3568-nvr-demo-v12-linux-spi-nand
# 默认即可不需要修改
```

```
export RK_BOOT_IMG=zboot.img
```

Rootfs编译

执行后会把`build/rootfs/`目录打包成特定格式的img固件，格式为根目录下build.sh的配置`RK_ROOTFS_TYPE`

```
### Rootfs编译命令
./build_emmc.sh rootfs
emmc配置
# parameter分区表，增删分区可以修改这个文件,build/parameter-nvr-emmc.txt
export RK_PARAMETER=parameter-nvr-emmc.txt
# rootfs格式，默认支持ext4 squashfs ubi
export RK_ROOTFS_TYPE=ext4
# 默认即可不需要修改
export RK_ROOTFS_IMG=rootfs.${RK_ROOTFS_TYPE}
# update打包文件，增删分区后需要修改这个问题以便打包正确的update.img。
tools/linux/Linux_Pack_Firmware/rockdevrk356x-package-file-nvr-emmc.txt
export RK_PACKAGE_FILE=rk356x-package-file-nvr-emmc

spi nand配置
# parameter分区表，增删分区可以修改这个文件,build/parameter-nvr-spinand.txt
export RK_PARAMETER=parameter-nvr-spinand.txt
# rootfs格式，默认支持ext4 squashfs ubi，spi nand只支持squashfs ubi
export RK_ROOTFS_TYPE=ubi
# 默认即可不需要修改
export RK_ROOTFS_IMG=rootfs.${RK_ROOTFS_TYPE}
# update打包文件，增删分区后需要修改这个问题以便打包正确的update.img。
tools/linux/Linux_Pack_Firmware/rockdevrk356x-package-file-nvr-spi-nand.txt
export RK_PACKAGE_FILE=rk356x-package-file-nvr-spi-nand
```

客户可以自行在`build/rootfs/`增删根文件系统内容。

固件打包

上面 Kernel/U-Boot/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 目录下并且生成编译时间、XML、补丁和固件到IMAGE目录：

```
### 固件打包命令
./build_emmc.sh update
```

应用编译

目前的编译方式只支持CMake脚本，其它编译系统可以参考build/app/build/build.sh脚本配置编译工具链。

这里以我们发布的RKMPI为例

1. 使能sdk环境

```
在根目录执行 ./build_emmc.sh env
```

2. 编译

```
cd build/app/build
./build.sh ../RKMPI_Release/SDK/rockit/
```

编译生成的bin文件会在 `build/app/RKMPI_Release/SDK/bin` 目录下。

编译出来的bin文件可以通过以下两种方式放到板端运行：

1. 可以放到 `build/rootfs/usr/bin` 目录下然后执行 `./build_emmc.sh rootfs` 重新生成 `rootfs.img` 然后烧写。
2. 板端挂载nfs设备 `mount -t nfs -o nolock 169.254.210.33:/opt/rootfs /mnt/nfs`

RKMPI媒体包

RKMPI是Rockchip 多媒体处理平台接口，在 `build/app/RKMPI_Release` 目录下有带一份发布的RKMPI包

相关使用文档参考： `build/app/RKMPI_Release/README.md`

固件烧写

烧写文档请参考 `docs/RK356x/Rockchip_RK356X_Linux_SDK_*_V*_*_CN.pdf`

SecureBoot功能

安全启动功能文档清参考
`docs/Linux/Security/Rockchip_Developer_Guide_Linux_Secure_Boot_CN.pdf`

FAQ

Q: 把资源放到build/rootfs/usr/bin目录下编译rootfs后烧写提示rootfs分区太小。
A: 需要修改parameter分区表，文件存放在build目录下。
emm: parameter-nvr-emmc.txt
spi-nand: parameter-nvr-spinand.txt
分区大小算法：比如rootfs要配置200M，则 200M * 2048 == 0x64000

0x00064000	**0x0000a800**	**rootfs**
分区大小	分区起始地址	分区名字

Q: rootfs设置为ext4格式下，烧写成功后根目录剩余空间太小。
A: 因为rootfs是根据build/rootfs目录大小打包的没有剩余太多空间，可以按照如下修改生成新的img大小，总大小不能超过parameter配置的分区大小。

```
+++ b/tools/build_emmc.sh
@@ -154,7 +154,7 @@ function build_rootfs(){
                        SRC_DIR=rootfs/
                        tar cf $TEMP $SRC_DIR &>/dev/null
                        SIZE=$(du -m $TEMP|grep -o "^[0-9]*")
-                       let SIZE+=10 #fix out of space
+                       let SIZE+=50 #fix out of space
                        rm -rf $TEMP
                        echo "Making $SRC_DIR (auto size:${SIZE}M)"
                        rm -rf rootfs.img rootfs.ext4
```