Rockchip RK3588 Linux NVR SDK Quick Start

ID: RK-JC-YF-543

Release Version: V1.7.0

Release Date: 2023-10-19

Security Level: □Top-Secret □Secret □Internal ■Public

DISCLAIMER

Trademark Statement

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: [fae@rock-chips.com]

Preface

Overview

Rockchip NVR SDK getting started and compilation guide.

Product Version

Chip Name	Kernel Version
RK3588	Linux 5.10

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	Author	Revision Date	Revision Description
V0.9.0	Mark.Huang	2021-12-01	Initial version
V1.0.0	Mark.Huang	2022-01-11	Adjust FAQ format and content
V1.1.0	Mark.Huang	2022-01-25	Reduce the size of boot FAQ
V1.2.0	Mark.Huang	2022-02-22	Reduce the size of boot FAQ
V1.3.0	Mark.Huang	2022-02-24	FAQ add the method of enabling HDMI-IN function FAQ adjust the printing instructions of cpu core FAQ add the method of reading SDK version information FAQ add the method of RAMBOOT
V1.4.0	Mark.Huang	2022-03-16	Add nvr new board type FAQ update the method of enabling HDMI-IN function FAQ add the method of enabling hardware security module
V1.5.0	Mark.Huang	2022-05-01	FAQ add the method of CPU accessing the DDR address space above 4G in uboot FAQ add the method of enabling MessageQueue FAQ add the method of checking HDMI/DP connect state FAQ add the method of modifying SATA3 from 6G to 3G
V1.5.1	Mark.Huang	2022-05-06	Update Application compiling
V1.5.2	Mark.Huang	2022-06-09	FAQ update HDMI-IN FAQ add the method of UVC FAQ add the method of AIQ
V1.5.3	Mark.Huang	2022-07-07	Update Some description
V1.5.4	Mark.Huang	2022-07-28	FAQ add multi-screen splicing interface instructions
V1.5.5	Mark.Huang	2022-10-12	FAQ add prompt egl initialization failure description Update part of the description
V1.5.6	XYP	2022-12-21	FAQ Update kernel packaging lzma description Added reserved CMA memory as a special description Add configuration download 575 firmware instructions
V1.6.0	XYP	2023-05-08	FAQ add vp to adjust the recommended frequency
V1.6.1	XYP	2023-05-11	Add description to FAQ
V1.7.0	XYP	2023-10-19	Add description to FAQ

Content

1. Introduction

The document aims to help developers get started with the development and debugging of the RK3588 Linux NVR SDK faster.

2. Set up an Development Environment

We recommend compiling with Ubuntu 18.04. Other Linux versions may require corresponding adjustments to the software packages. In addition to system requirements, there are other hardware and software requirements.

Hardware requirements: 64-bit system, hard disk space greater than 40G. If you do multiple builds, more hard drive space will be required.

Software requirements: Ubuntu 18.04 system:

The software package installation commands that are required to build the SDK environment are as follows:

```
sudo apt-get install repo git ssh make gcc libssl-dev liblz4-tool \
expect 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 python-pip ncurses-dev pyelftools \
```

It is suggested to use the Ubuntu 18.04 system or higher version for development. If an error is reported during compilation, you can install the corresponding software package according to the error message.

Consider the time cost of the customer to build the development environment, we also provide a cross-compiler docker mirror method for customer verification, which shortens the time required for building the compilation environment.

Refer to the document "Rockchip_User_Guide_SDK_Docker_EN.pdf".

The results of the compatibility test of the Docker compiled mirror system are as follows:

System Version	Load Firmware	Firmware Version
ubuntu14.04	PASS	PASS
ubuntu18.04	PASS	PASS
ubuntu16.04	PASS	PASS
fedora21	PASS	PASS
centos7.2	PASS	PASS
centos6.5+	PASS	PASS
centos7.4	PASS	PASS
centos7.5	PASS	PASS
centos7.6	PASS	PASS

3. SDK Directories Introduction

The SDK directory contains kernel, u-boot, tools, docs, rkbin and other directories. Each directory or its subdirectories corresponds to a git project, and submissions need to be made in their respective directories.

```
- SDK
  -- docs           //Save development guides, platform support lists, tool usage documents, Linux development guides, etc.
  -- kernel         //Save code developed by Kernel 5.10.
  -- rkbin          //Save Rockchip related binaries and tools.
  -- tools          //Save common tools under Linux and Window operating systems.
  -- u-boot         //Save the U-Boot code developed based on the v2017.09 version.
  -- IMAGE          //Save compile time, XML, patch and firmware directories for each build.
  -- rockdev        //Save compiled firmware.
  -- build          //Save the compilation script, rootfs and toolchain compilation toolchain.
```

4. Development Document Index

Development guides, platform support lists, tool usage documents, Linux development guides, etc. are placed in the docs directory by default:

```
├── docs
│    ├── Common (General kernel driver and module related documents, DDR/Flash/eMMC/Camera/WiFi/Bluetooth compatibility list)
│    │
```

```
|    ├── Linux (Rockchip Linux system general documents, RK3588 platform can refer)
|    |   ├── ApplicationNote
|    |   ├── Camera
|    |   ├── Graphics
|    |   ├── Multimedia
|    |   ├── Profile
|    |   ├── Recovery
|    |   └── Security (Encrypted documents)
|    |
|    ├── Others
|    |   ├── Rockchip_User_Guide_Bug_System_EN.pdf
|    |   └── Rockchip_User_Guide_SDK_Application_And_Synchronization_EN.pdf
|    |
|    └── RK3588
|        ├── Rockchip_RK3588_Linux_NVR_SDK_xxx_V1.0.0_xxxxxx_EN.pdf (SDK release instruction)
|        └── Rockchip_RK3588_Quick_Start_Linux_EN.pdf (Quick development guide)
|
build/app/RKMPI_Release/doc（RKMPI multimedia framework API documents）
├── README.pdf
├── Rockchip_Developer_Guide_MPI_AUDIO_EN.pdf
├── Rockchip_Developer_Guide_MPI_AVS_EN.pdf
├── Rockchip_Developer_Guide_MPI_DUMP_EN.pdf
├── Rockchip_Developer_Guide_MPI_GDC_EN.pdf
├── Rockchip_Developer_Guide_MPI_MMZ_EN.pdf
├── Rockchip_Developer_Guide_MPI_RGN_EN.pdf
├── Rockchip_Developer_Guide_MPI_SYS_EN.pdf
├── Rockchip_Developer_Guide_MPI_TDE_EN.pdf
├── Rockchip_Developer_Guide_MPI_VDEC_EN.pdf
├── Rockchip_Developer_Guide_MPI_VENC_EN.pdf
├── Rockchip_Developer_Guide_MPI_VGS_EN.pdf
├── Rockchip_Developer_Guide_MPI_VI_EN.pdf
├── Rockchip_Developer_Guide_MPI_VO_EN.pdf
├── Rockchip_Developer_Guide_MPI_VPSS_EN.pdf
├── Rockchip_FAQ_MPI_EN.pdf
└── Rockchip_Release_Note_MPI_EN.pdf
```

5. SDK Compilation Instruction

There are two compilation scripts `build_emmc.sh` and `build_spi_nand.sh` in the root directory for compiling emmc and spi nand devices respectively.

The compilation instructions of the two scripts are the same. The following is an example of `build_emmc.sh`.

5.1 Check compile command

Run the command in the root directory: /build_emmc.sh -h|help

```
rk3588$ ./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'.
```

5.2 Automatic compiling

Enter the project root directory and run the following command to automatically complete all compilations:

```
./build_emmc.sh all # Compile module code (u-Boot, kernel, Rootfs), and
package firmware
./build_emmc.sh # Same as above
```

5.3 Compile and package each module

5.3.1 U-Boot compiling

```
### U-Boot compile command
./build_emmc.sh uboot

### U-Boot configuration parameters
emmc configuration
# Default compile configuration, no need to modify
export RK_UBOOT_DEFCONFIG=rk3588
# The default can not be modified
export RK_UBOOT_FORMAT_TYPE=fit

spi nand configuration
# Uboot defconfig
export RK_UBOOT_DEFCONFIG=rk3588
# Uboot image format type: fit(flattened image tree)
export RK_UBOOT_FORMAT_TYPE=fit
# Uboot update loader (spl)
# Compile uboot with the default spl-new parameter to ensure that spl, etc.
are generated with the latest code
export RK_LOADER_UPDATE_SPL=true
```

5.3.2 Kernel compiling

```
### Kernel compile command
./build_emmc.sh kernel

### Kernel configuration parameters
emmc configuration
# Kernel default configuration
export RK_KERNEL_DEFCONFIG=rockchip_linux_defconfig
# Kernel nvr product configuration
export RK_KERNEL_DEFCONFIG_FRAGMENT=rk3588_nvr.config
#export RK_KERNEL_DTS=rk3588-evb1-lp4-v10-linux  #Select this if the
hardware board type is EVB1
#export RK_KERNEL_DTS=rk3588-nvr-demo1-v21
export RK_KERNEL_DTS=rk3588-nvr-demo-v10
# The default can not be modified
export RK_BOOT_IMG=zboot.img

spi nand configuration
# Kernel defconfig
export RK_KERNEL_DEFCONFIG=rockchip_linux_defconfig
# Kernel defconfig fragment
export RK_KERNEL_DEFCONFIG_FRAGMENT=rk3588_nvr.config
# Kernel dts
export RK_KERNEL_DTS=rk3588-nvr-demo-v10-spi-nand
# boot image type
export RK_BOOT_IMG=zboot.img
```

By default, the kernel is currently adapted to 3 types of boards. Customers need to select the corresponding dts configuration to compile according to the hardware board type to avoid exceptions caused by mismatches. The corresponding relationships are as follows:

Hardware Board Type	Corresponding DTS	Corresponding Hardware Guide
RK3588 EVB1 LP4 V10 Board	rk3588-evb1-lp4-v10-linux	Rockchip_RK3588_EVB_User_Guide_V1.0_CN
RK3588 NVR DEMO V10 Board	rk3588-nvr-demo-v10	
RK3588 NVR DEMO1 V21 Board	rk3588-nvr-demo1-v21	

5.3.3 Rootfs compiling

After execution, the `build/rootfs/` directory will be packaged into img firmware in a specific format, the format is the configuration `RK_ROOTFS_TYPE` of build.sh in the root directory.

```
### Rootfs compile command
./build_emmc.sh rootfs
emmc configuration
# Parameter partition table, adding or deleting partitions can modify this
file, build/parameter-nvr-emmc.txt
```

```
export RK_PARAMETER=parameter-nvr-emmc.txt
# Rootfs format, supports ext4 squashfs ubi by default
export RK_ROOTFS_TYPE=ext4
# The default can not be modified
export RK_ROOTFS_IMG=rootfs.${RK_ROOTFS_TYPE}
# The update package file, after adding or deleting partitions, you need to modify this problem in order to package the correct update.img.
tools/linux/Linux_Pack_Firmware/rockdev/rk3588-package-file-nvr-emmc.txt
export RK_PACKAGE_FILE=rk3588-package-file-nvr-emmc

spi nand configuration
# Parameter partition table, adding or deleting partitions can modify this file, build/parameter-nvr-spinand.txt
export RK_PARAMETER=parameter-nvr-spinand.txt
# Rootfs format, supports ext4 squashfs ubi by default, spi nand only supports squashfs ubi
export RK_ROOTFS_TYPE=ubi
# The default can not be modified
export RK_ROOTFS_IMG=rootfs.${RK_ROOTFS_TYPE}
# The update package file, after adding or deleting partitions, you need to modify this problem in order to package the correct update.img.
tools/linux/Linux_Pack_Firmware/rockdevrk3588-package-file-nvr-spi-nand.txt
export RK_PACKAGE_FILE=rk3588-package-file-nvr-spi-nand
```

Customers can add or delete the content of the root file system in `build/rootfs/`.

5.3.4 Firmware packaging

After compiling the various parts of Kernel/U-Boot/Rootfs above, enter the root directory of the project directory and run the following command to automatically package all firmware into the rockdev directory and generate compile time, XML, patches and firmware to the IMAGE directory:

```
### Firmware packaging command
./build_emmc.sh update
```

5.4 Application compiling

The current compilation method only supports CMake scripts. For other compilation systems, you can refer to the build/app/build/build.sh script to configure the compilation toolchain.

Take our released RKMPI as an example:

1. Enable SDK environment

```
Run in the root directory ./build_emmc.sh env
```

2. Compile

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

The compiled bin file will be in the `build/app/bin` directory.

Note: If make RKMPI fail, try ./build_emmc.sh cleanenv to clean env first.

The compiled bin file can be run on the board in the following two ways:

1. You can put it in the `build/rootfs/usr/bin` directory and run `./build_emmc.sh rootfs` to regenerate `rootfs.img` and then program it.
2. Mount the nfs device on the board `mount -t nfs -o nolock 169.254.210.33:/opt/rootfs /mnt/nfs`.

6. RKMPI Media Package

RKMPI is the interface of the Rockchip multimedia processing platform. There is a released RKMPI package in the `build/app/RKMPI_Release` directory.

For related documents, please refer to: `build/app/RKMPI_Release/README.md`

Media FAQ document: `build/app/RKMPI_Release/doc/en/Rockchip_FAQ_MPI_EN.pdf`

7. Firmware Programming

For programming, please refer to the sdk release document programming chapter.

8. SecureBoot Function

For the secure boot function, please refer to the document:
`docs/en/Linux/Security/Rockchip_Developer_Guide_Linux_Secure_Boot_EN.pdf`.

9. FAQ

9.1 Software and hardware version adaptation standards. After software adaptation, you need to confirm that the following nodes are normal

1. Check the frequency and voltage table to confirm that all nodes used are present and configured correctly. For example, CPU, GPU, NPU, RKVENC, etc. The voltage corresponding to different chips may be different. You can check whether it is normal by comparing it with dts.

```
[root@RK3588:/userdata]# cat /sys/kernel/debug/opp/opp_summary
 device                rate(Hz)     target(uV)     min(uV)     max(uV)
-----------------------------------------------------------------------
 platform-fdab0000.npu
                      300000000       675000      675000      850000
```

```
                                    675000      675000      850000
                     400000000      675000      675000      850000
                                    675000      675000      850000
                     500000000      675000      675000      850000
                                    675000      675000      850000
                     600000000      675000      675000      850000
                                    675000      675000      850000
                     700000000      687500      687500      850000
                                    687500      687500      850000
                     800000000      725000      725000      850000
                                    725000      725000      850000
                     900000000      762500      762500      850000
                                    762500      762500      850000
                    1000000000      812500      812500      850000
                                    812500      812500      850000
platform-fdbe0000.rkvenc-core
                     800000000      775000      775000      850000
                                    775000      775000      850000
platform-fdbd0000.rkvenc-core
                     800000000      775000      775000      850000
                                    775000      775000      850000
platform-fb000000.gpu
                     300000000      675000      675000      850000
                                    675000      675000      850000
                     400000000      675000      675000      850000
                                    675000      675000      850000
                     600000000      675000      675000      850000
                                    675000      675000      850000
                     700000000      675000      675000      850000
                                    675000      675000      850000
                     800000000      712500      712500      850000
                                    712500      712500      850000
                     900000000      762500      762500      850000
                                    762500      762500      850000
cpu4
                    1200000000      687500      687500      950000
                                    687500      687500      950000
                    1416000000      725000      725000      950000
                                    725000      725000      950000
                    1608000000      812500      812500      950000
                                    812500      812500      950000
                    1800000000      912500      912500      950000
                                    912500      912500      950000
cpu2
                    1200000000      675000      675000     1000000
                                    675000      675000     1000000
                    1416000000      675000      675000     1000000
                                    675000      675000     1000000
                    1608000000      725000      725000     1000000
                                    725000      725000     1000000
                    1800000000      800000      800000     1000000
                                    800000      800000     1000000
                    2016000000      875000      875000     1000000
                                    875000      875000     1000000
                    2208000000      962500      962500     1000000
                                    962500      962500     1000000
cpu0
                    1200000000      675000      675000     1000000
```

```
                                    675000      675000      1000000
                    1416000000      675000      675000      1000000
                                    675000      675000      1000000
                    1608000000      725000      725000      1000000
                                    725000      725000      1000000
                    1800000000      800000      800000      1000000
                                    800000      800000      1000000
                    2016000000      875000      875000      1000000
                                    875000      875000      1000000
                    2208000000      962500      962500      1000000
                                    962500      962500      1000000
```

2. Check the voltmeter to confirm that the main power supplies such as vdd_cpu_lit_s0, vdd_cpu_big0_s0, vdd_cpu_big1_s0, vdd_npu_s0, vdd_gpu_s0 are functioning normally.

```
[root@RK3588:/]# cat /sys/kernel/debug/regulator/regulator_summary  | grep cpu
       vdd_cpu_lit_s0             1    3      0  normal   812mV      0mA 550mV   950mV
          cpu4-mem                0                                    0mA 812mV   950mV
          cpu4-cpu                0                                    0mA 725mV   950mV
          vdd_cpu_lit_s0          0                                    0mA 0mV     0mV
       vdd_cpu_big0_s0            1    3      0  normal   675mV      0mA 550mV  1050mV
          cpu0-mem                0                                    0mA 675mV  1000mV
          cpu0-cpu                0                                    0mA 675mV  1000mV
          vdd_cpu_big0_s0         0                                    0mA 0mV     0mV
       vdd_cpu_big1_s0            1    3      0  normal   675mV      0mA 550mV  1050mV
          cpu2-mem                0                                    0mA 675mV  1000mV
          cpu2-cpu                0                                    0mA 675mV  1000mV
          vdd_cpu_big1_s0         0                                    0mA 0mV     0mV
[root@RK3588:/]# cat /sys/kernel/debug/regulator/regulator_summary  | grep npu
       vdd_npu_s0                 1    5      0  normal   812mV      0mA 550mV   950mV
          fdab0000.npu-mem        0                                    0mA 812mV   950mV
          fdab0000.npu-rknpu      0                                    0mA 812mV   950mV
          fdab0000.npu-mem        0                                    0mA 0mV     0mV
          fdab0000.npu-rknpu      0                                    0mA 0mV     0mV
          vdd_npu_s0              0                                    0mA 0mV     0mV
[root@RK3588:/]# cat /sys/kernel/debug/regulator/regulator_summary  | grep gpu
```

```
        vdd_gpu_s0                 1    5       0  normal   675mV      0mA
550mV   950mV
           fb000000.gpu-mem        0                                     0mA
675mV   950mV
           fb000000.gpu-mali       0                                     0mA
675mV   950mV
           fb000000.gpu-mem        0                                     0mA
0mV     0mV
           fb000000.gpu-mali       0                                     0mA
0mV     0mV
           vdd_gpu_s0              0                                     0mA
0mV     0mV
```

9.2 Put the resources in the build/rootfs/usr/bin directory and compile the rootfs and then program it, it points that the rootfs partition is too small

A: The parameter partition table needs to be modified, and the files are saved in the build directory.

emmc: parameter-nvr-emmc.txt

spi-nand: parameter-nvr-spinand.txt

Partition size algorithm: For example, if rootfs needs to be configured with 200M, then 200M * 2048 == 0x64000

0x00064000	**0x0000a800**	**rootfs**
Partition Size	Partition Start Address	Partition Name

9.3 When rootfs is set to ext4 format, the remaining space of the root directory is too small after programming

A: Because rootfs is packaged according to the size of the build/rootfs directory and there is not much space left, you can generate a new img size as follows, and the total size cannot exceed the partition size configured by parameter.

```
+++ 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
```

9.4 How to check NPU/GPU/CPU/DDR/VDEC frequency usage, etc

A: Check the NPU frequency: cat sys/kernel/debug/clk/clk_summary |grep scmi_clk_npu

Check the GPU frequency: cat /sys/devices/platform/fb000000.gpu/devfreq/fb000000.gpu/cur_freq

Check the GPU load: cat /sys/devices/platform/fb000000.gpu/devfreq/fb000000.gpu/load

Check the CPU frequency: small cores: cat /sys/devices/system/cpu/cpu4/cpufreq/scaling_cur_freq Big cores(1-2): cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq Big cores(3-4): cat /sys/devices/system/cpu/cpu2/cpufreq/scaling_cur_freq

Check available frequency voltmeters: cat /sys/kernel/debug/opp/opp_summary

Enable the CPU performance mode, run the highest frequency: Small cores: echo performance > /sys/devices/system/cpu/cpu4/cpufreq/scaling_governor Big cores(1-2): echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor Big cores(3-4): echo performance > /sys/devices/system/cpu/cpu2/cpufreq/scaling_governor

Enable the GPU performance mode, run the highest frequency: echo performance > /sys/devices/platform/fb000000.gpu/devfreq/fb000000.gpu/governor

Check the DDR frequency: cat /sys/kernel/debug/clk/scmi_clk_ddr/clk_rate

Check the VDEC frequency: cat sys/kernel/debug/clk/clk_summary |grep clk_rkvdec

9.5 UBOOT will load the boot partition by default, how to change the boot partition to another name in the parameter

A: Steps to modify:

1. Modify the boot partition to the required name in the parameter.
2. Under uboot, modify `u-boot/include/boot_rkimg.h` accordingly.

```
#define PART_BOOT                       "boot"
#define ANDROID_PARTITION_BOOT          "boot"
```

9.6 How does the kernel identify and load the root file system

A: Kernel will identify the root file system in two ways:

1. Assign the root file system in bootargs. Refer to the practice of ubi:

```
bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0 ubi.mtd=4
root=ubi0:rootfs rootfstype=ubifs";
```

2. The RK programming (window, linux) tool identifies the uuid:rootfs in the parameter, and then adds the uuid to the rootfs partition. The default configuration in the bootargs of kernel uses the uuid to find the rootfs.

```
bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0
root=PARTUUID=614e0000-0000 rw rootwait";
```

9.7 How to modify rootfs to initramfs

A: Steps to modify:

1. Prepare rootfs.cpio.gz

   You can decompress the prepared rootfs.tar.gz and package the initramfs package. The following takes the rootfs.tar.gz in the build directory of nvr sdk as an example:

```
cd build/
mkdir rootfs
tar -zxvf rootfs.tar.gz -C rootfs/
cd rootfs
```

   Need to add mount: mount -t devtmpfs devtmpfs /dev, the default ramdisk is not automatically mounted. In addition, dev/console needs to be deleted, otherwise the serial port cannot output normally.

```
+++ b/rootfs/etc/inittab
@@ -15,6 +15,7 @@
 # Startup the system
 +::sysinit:/bin/mount -t devtmpfs devtmpfs /dev
 ::sysinit:/bin/mount -t proc proc /proc
 ::sysinit:/bin/mount -o remount,rw /
 ::sysinit:/bin/mkdir -p /dev/pts
 ::sysinit:/bin/mkdir -p /dev/shm
 # Delete the default console file
 rm rootfs/dev/console -rf
```

   Package rootfs.cpio.gz

```
cd rootfs
find . | cpio -o -H newc -O ../rootfs.cpio      #Be sure to enter rootfs to generate
cd -
ls -l rootfs.cpio
-rw-rw-r-- 1 huangjc huangjc 97453056 Dec 13 17:19 rootfs.cpio
gzip -9 -f rootfs.cpio
ls -l rootfs.cpio.gz
-rw-rw-r-- 1 huangjc huangjc 38610705 Dec 13 17:19 rootfs.cpio.gz
```

2. Modify the bootargs parameter in the dts corresponding to the machine kernel, and the mount point is /dev/ram, such as the default nvr demo (emmc) board dts modification:

```
huangjc@S1-GITSER-144:~/rk3588_nvr_sdk/kernel$ git diff .
diff --git a/arch/arm64/boot/dts/rockchip/rk3588-linux.dtsi
b/arch/arm64/boot/dts/rockchip/rk3588-linux.dtsi
index 8f75679..fc6445f 100644
--- a/arch/arm64/boot/dts/rockchip/rk3588-linux.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-linux.dtsi
@@ -6,7 +6,7 @@
/ {
        chosen: chosen {
-               bootargs = "earlycon=uart8250,mmio32,0xfeb50000
console=ttyFIQ0 clk_gate.always_on=1 pm_domains.always_on=1
root=PARTUUID=614e0000-0000 rw rootwait";
+               bootargs = "earlycon=uart8250,mmio32,0xfeb50000
console=ttyFIQ0 clk_gate.always_on=1 pm_domains.always_on=1
root=/dev/ram rdinit=/sbin/init rw rootwait";
        };

        fiq_debugger: fiq-debugger {
```

3. Compile and generate boot.img:

```
#Compile the kernel first
./build_spi_nand.sh kernel

#Package the boot image with ramdisk
cd kernel
cp arch/arm64/boot/Image .
./scripts/mkbootimg --kernel Image  --ramdisk ../build/rootfs.cpio.gz --
second resource.img -o boot.img
```

4. Adjust the partition table: At this time, the boot image will be much larger. The boot partition should be setting larger in the partition table. The rootfs partition can be removed according to the situation. For example, the partition table configuration of spi nand can be changed to:

```
huangjc@S1-GITSER-144:~/rk3588_nvr_sdk/build$ git diff
diff --git a/parameter-nvr-spinand.txt b/parameter-nvr-spinand.txt
index 9bd949a..9cb37af 100755
--- a/parameter-nvr-spinand.txt
+++ b/parameter-nvr-spinand.txt
@@ -8,5 +8,5 @@ MACHINE: 0xffffffff
 CHECK_MASK: 0x80
 PWR_HLD: 0,0,A,0,1
 TYPE: GPT
 -CMDLINE:
mtdparts=rk29xxnand:0x00000800@0x00000800(vnvm),0x00002000@0x00001000(ub
oot),0x00006000@0x00003000(boot),0x00024000@0x00009000(rootfs),-
@0x0002D000(userdata:grow)
 +CMDLINE:
mtdparts=rk29xxnand:0x00000800@0x00000800(vnvm),0x00002000@0x00001000(ub
oot),0x0002A000@0x00003000(boot),-@0x0002D000(userdata:grow)
 uuid:rootfs=614e0000-0000-4b53-8000-1d28000054a9
```

Just program the normal loader uboot and the compiled boot.img.

5. Note: If it is packaged in the zImage compression format, the default decompressed boot cannot exceed 48M. If it exceeds, the uboot decompression will fail. The reason is that the decompression address exceeds the limit and the compressed address is overwritten. If you use the initram method to package the kernel, you need to confirm that if the kernel is lzma, you need to change it to lz4 compression. Modify as follows, assign the address of ramdisk_addr_r to kernel_addr_c, which can be expanded to more than 100M:

```
huangjc@S1-GITSER-144:~/rk3588_nvr_sdk/u-boot$ git diff
diff --git a/include/configs/rk3588_common.h
b/include/configs/rk3588_common.h
index 4966dc4826..3c062df185 100644
--- a/include/configs/rk3588_common.h
+++ b/include/configs/rk3588_common.h
@@ -56,7 +56,7 @@
        "pxefile_addr_r=0x00600000\0" \
        "fdt_addr_r=0x0a100000\0" \
        "kernel_addr_r=0x00400000\0" \
-       "kernel_addr_c=0x03080000\0" \
+       "kernel_addr_c=0x0a200000\0" \
        "ramdisk_addr_r=0x0a200000\0"

 #include <config_distro_bootcmd.h>
```

9.8 Instructions for vendor partitions

A: The vendor partition is reserved for saving ETH MAC address, machine serial number and other information. The MTD device needs to add a vnvm partition in the partition table (parameter); the vendor partition of other devices such as EMMC is included in the first 4M space.

9.9 How to remove vendor partition

A: EMMC: If the vendor partition is no required, the `CONFIG_ROCKCHIP_VENDOR_PARTITION` configuration needs to be turned off in uboot.

SPI NAND: If the vendor partition is no required, the `CONFIG_ROCKCHIP_VENDOR_PARTITION` configuration needs to be turned off in uboot, and remove the vnvm partition in partition table.

If there is no vendor partition, the information such as MAC address that needs to be saved by the user, for example it can be saved in the ENV: `setenv -f ethmac 00:11:22:33:44:55` .

Note: The starting address of the uboot partition under the EMMC device is suggested to start from 4M, if it is less than 4M (minimum 3M), the vendor partition must be closed according to the above method and the `CONFIG_ENV_IS_IN_MMC` configuration cannot be enabled, otherwise the uboot firmware will be lost.

9.10 How to operate vendor partition

A: It can be read and written by vendor_storage this tool , or by the PC tool tools\windows\RKDevInfoWriteTool_1.2.6.

```
There are 16 types
                "VENDOR_SN_ID"
```

```
                "VENDOR_WIFI_MAC_ID"
                "VENDOR_LAN_MAC_ID"
                "VENDOR_BT_MAC_ID"
                "VENDOR_HDCP_14_HDMI_ID"
                "VENDOR_HDCP_14_DP_ID"
                "VENDOR_HDCP_2x_ID"
                "VENDOR_DRM_KEY_ID"
                "VENDOR_PLAYREADY_Cert_ID"
                "VENDOR_ATTENTION_KEY_ID"
                "VENDOR_PLAYREADY_ROOT_KEY_0_ID"
                "VENDOR_PLAYREADY_ROOT_KEY_1_ID"
                "VENDOR_SENSOR_CALIBRATION_ID"
                "VENODR_RESERVE_ID_14"
                "VENDOR_IMEI_ID"
                "VENDOR_CUSTOM_ID"
                And custom can define other id like
                VENDOR_CUSTOM_ID_1A (define ID = 26)
```

9.11 How to grab a flame graph for analysis

A: How to grab a flame graph:

1. Push the perf tool, grab perf data, and run on the machine:

```
perf record -a -g -e cpu-cycles -p 643 -o data/perf.data
```

2. Convert it to a flame graph and run on the machine:

```
perf script --symfs=/ -i perf.data > perf.unfold
```

3. Export perf.unfold to the FlameGraph directory on the PC, and run on the PC:

```
./stackcollapse-perf.pl perf.unfold &> perf.folded
./flamegraph.pl perf.folded > perf.svg
```

9.12 The machine has not programmed boot.img, and uboot cannot recognize the network card normally

A: uboot needs to rely on the dtb of boot.img. If there is no boot.img in the machine, then you need to package the dtb in uboot.

1. Rename the dtb of the kernel to kern.dtb, put it in the uboot/dts directory, and then recompile uboot.

   After compilation, dtb will be packaged in uboot.img, and it is no longer necessary to rely on dtb in boot.

9.13 The machine has no reserved programming button, how to enter the programming mode

A: There are two programming modes on the RK platform: the Maskrom mode and the Loader mode (U-Boot).

1. How to enter the Loader programming mode:

   Method 1: When turning on the machine, press and hold the Recovery button.

   Method 2: When turning on the machine, press and hold the ctrl+d key combination in the pc serial port.

   Method 3: Enter in the U-Boot command line: download or rockusb 0 *devtype*devnum.

2. How to enter the Maskrom programming mode:

   Method 1: When turning on the machine, press and hold the ctrl+b key combination in the pc serial port.

   Method 2: Enter in the U-Boot command line: rbrom.

9.14 How to save the ENV to flash in uboot

A: Enable the ENV partition in uboot, the ENV is saved in memory (CONFIG_ENV_IS_NOWHERE) by default.

1. Open CONFIG_ENV_IS_IN_BLK_DEV configuration in uboot's config.
2. Add the partition to save the ENV in parameter, for example, 0x00000800@0x00001800(env).
3. Modify CONFIG_ENV_OFFSET and CONFIG_ENV_SIZE in the uboot configuration file according to the size of the env partition. For example (from 3M, size 1M): CONFIG_ENV_OFFSET=0x300000, CONFIG_ENV_SIZE=0x100000.
4. Call env_save() in uboot to save, or write with setenv -f xxx xxx.
5. To read the data of the env partition in the kernel, you can use the tool fw_printenv under u-boot/tools/env.

   Compile fw_printenv ./make.sh env.

   Note: The parameters such as MTD device name, Device offset and Env. size in fw_env.config need to be configured according to the location and size of the actual env partition.

   u-boot/tools/env/fw_printenv // the env read and write tool

   u-boot/tools/env/fw_env.config // the env configuration file

   u-boot/tools/env/README // documents for the env read-write tool

9.15 How to modify the serial port baud rate

A: The default baud rate of the SDK is 1.5M. If you need to modify it, follow the steps below to modify the tool in the rkbin/tools directory:

1. Confirm the packaged bin: the packaged script is specified in uboot config, the default is rkbin/RKBOOT/RK3588MINIALL.ini. Please check which ddr bin is packaged.
2. Copy the packaged bin: Copy the corresponding packaged ddr bin to the rkbin/tools directory.
3. Modify parameters: Only modify the parameters that need to be modified, and do not modify other parameters.

   If you want to modify the serial port baud rate, please modify uart baudrate=115200 in ddrbin_param.txt.

If you want to modify the frequency of ddr, please modify the frequency of the corresponding ddr type, for example, ddr lp4: lp4_freq= 2112.

4. Run the command to modify the bin: Run ./ddrbin_tool ddrbin_param.txt DDR_BIN_NAME.bin (DDR_BIN_NAME is the bin copied in the third step).
5. Copy and overwrite the original bin: Copy the modified ddr bin to the rkbin/bin/rk35 directory.
6. Recompile: Recompile uboot to generate a new loader.
7. Under the kernel, modify the required baud rate in dts.

   Refer to: kernel/arch/arm64/boot/dts/rockchip/rk3588-linux.dtsi:18: rockchip,baudrate = <1500000>; /* Only 115200 and 1500000 */

9.16 How to start boot under uboot

A: Under uboot, the system can be loaded by the following methods:

1. After identifying the U disk/emmc, etc., load the firmware in it.

   Use the fatload command to load the kernel in the U disk/emmc, and then use the bootm command to start it.

   `fatload <interface> <dev[:part]> <addr> <filename> <bytes>`

   interface: the interface used, such as: MMC, USB.

   dev [:part]: The device where the file is saved, eg: ide 0:1.

   addr: starting address saved in memory.

   filename: the name of the loaded file.

   bytes: the number of bytes to copy.

   ```
   For example:
   usb  start          Initialize the USB device. If it is not executed, the content of the usb device cannot be operated.
   fatls usb 0         List all files in U disk.
   fatinfo usb 0       U disk properties and other information.
   fatload usb 0:1 0x20000000 boot.img
   bootm 0x20000000    Boot kernel from memory 0x20000000.
   ```

2. Download boot from ftp, please refer to `docs/en/Common/UBOOT/Rockchip_Developer_Guide_UBoot_Nextdev_EN.pdf`:

   ```
   dhcp 0x20000000 172.16.21.161:boot.img
   bootm 0x20000000
   ```

9.17 Why after changing the emmc of the SDK board to spi nand, it keeps going into maskrom mode after programming the firmware

A: Need to do the following checks:

1. You need to confirm that the partition defined in the parameter cannot exceed the capacity of the spi nand.

For example, if a spi nand with a total capacity of 128MB and a block size of 128K needs to reserve 5 blocks at the tail for bad block processing, etc., the maximum partition capacity cannot exceed 128M-128K*5 (about 127M).

If the defined partition exceeds the actual capacity, it will enter maskrom.

2. If the last partition with grow is in ubifs format, you need to use the RKDevTool after 2.89. Previous versions has problems for this case.

9.18 Ubifs file system space optimization, and production methods

A: There are optimizations for ubifs such as bad block management in the NVR SDK, can be at ease use.

1. Spi nand this kind of bare storage medium, for partitions of 3M and above, ubifs is suggested to use.

   In build_spi_nand.sh, by default, rootfs and data can be configured as ubifs, which needs to be configured as follows:

   ```
   export RK_ROOTFS_TYPE=ubi
   export RK_USERDATA_TYPE=ubi
   ```

   For other partitions to configure as ubifs, please refer to mk_ubi_image() in build/tools/mk-image.sh or refer to the document: docs/en/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_Flash_Open_Source_Solution_EN.pdf
   `

2. Ubi block supports squashfs

   Refer to the document: `docs/en/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_Flash_Open_Source_Solution_EN.pdf`

3. Ubifs space optimization

   Refer to the document: `docs/en/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_Flash_Open_Source_Solution_EN.pdf`

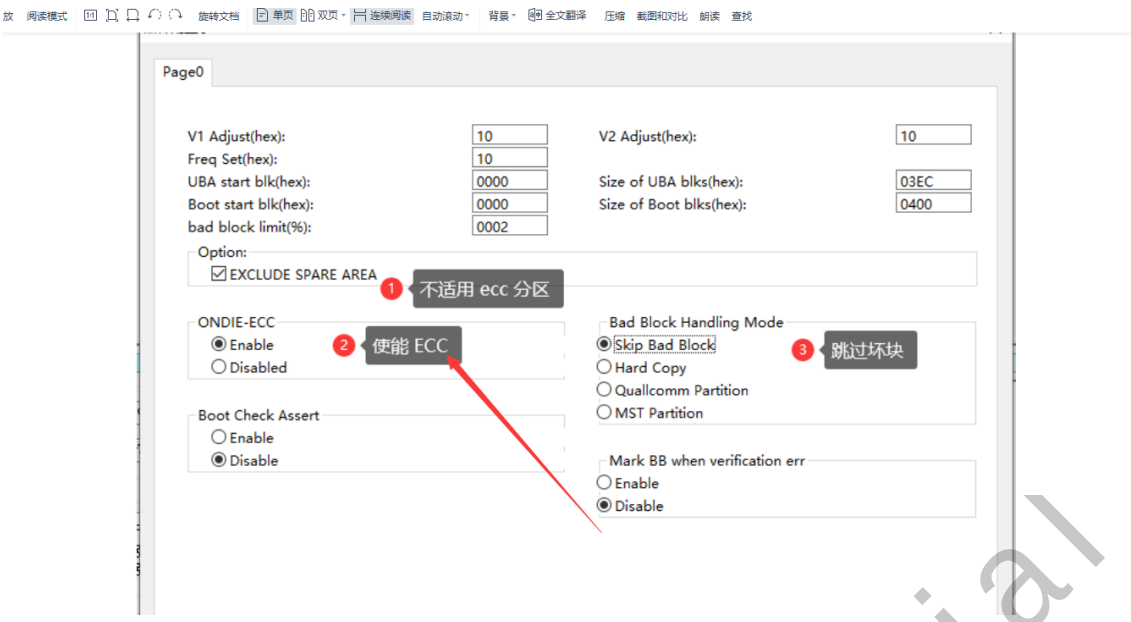
9.19 Whether spi nand supports the programmer to program firmware

A: Spi nand supports programmer to program firmware.

1. Refer to the programmer programming chapter in the document `docs/en/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_Flash_Open_Source_Solution_EN.pdf`, and use the `./tools/linux/programmer_image_tool/programmer_image_tool` tool to convert the image for programmer burning.

2. Use the tool provided by the programmer to package the separate images into a program image. For example: nsp7500 series.

   Note: There is no OOB in the image by default, and there is no ECC in the image. When the programmer is configured, the ONDIE-ECC option is turned on.



9.20 How to upgrade boot and other partitions in uboot

A: Uboot provides the tftpflash tool by default to upgrade all partitions except loader and GPT.

For specific usage, please refer to chapter 5.23.4 in `docs/cn/Common/UBOOT/Rockchip_Developer_Guide_UBoot_Nextdev_CN.pdf`.

You can also use the tftp tool to download to the ddr, and then use the mtd write interface to write.

9.21 How to upgrade the loader and GPT partition table in uboot

A: Need to follow the steps below to upgrade:

1. Convert the loader and parameter into a format that can be operated by mtd, assuming that the block size is 128K.

   ```
   ./tools/linux/programmer_image_tool/programmer_image_tool -i update.img -b 128 -p 2 -t spinand -o out
   ```

   Convert parameter to gpt.img, Miniloader to idblock.img. The size of gpt.img is 128K,and the size of idblock.img is 128K.

   Note: Programmer_image_tool cannot be converted to parameters alone, it needs to be packaged into update.img and then converted at one time. The loader can be converted separately.

2. Idblock does double backup, `cat out/idblock.img >> idblock_mutli_copies.img`.

3. When writing data to the flash, the size of the block is determined according to the model of the flash, which is normally 128k or 256k.

   - Write gpt.img to the 0th block, and call blk_dwrite(xx,0, sizeof(gtp.img)/512, gpt data). Note: The units of the parameters here are all 512B.
   - Write idblock_mutli_copies.img to the position of block 1st-6th.
   - Assuming that the block size is 128K, call blk_dwrite(xx, 128k/512, sizeof(idblock_mutli_copies.img)/512, xx data).

   Note: The written loader cannot exceed the position of the 7th block.

4. Note: When writing a backup idblock, the address of the 7th block cannot be exceeded. The idblock is only saved in the 1st-6th block, and the end address should be block 7. If the flash block is 256K, it should be noted that the size of the loader partition allocated in the parameter must be 2M.

9.22 How to upgrade the loader and GPT partition table in user mode

A: We do not recommend upgrading loader and GPT in user mode, as the risk is high.

Starting from address 0, create a loader partition, overwriting the previous reserved partition. The upper layer updates the loader and GPT functions by writing to this mtd0 partition.

Implementation methods and steps:

1. The loader partition is added to the parameter, and the subsequent partitions can be modified according to the needs. The capacity of the spi flash is relatively small, and the loader partition should be controlled within 1M (the size of 1M is assumed to be a block of 128K, if the block is 256K, it is controlled at 2M). The loader partition is allocated in the parameter as follows (1M is allocated, and the complete file content can be found in parameter.txt under the folder).

   CMDLINE: mtdparts=rk29xxnand:0x00000800@0x00000000(loader),0x00000800@0x00000800(vnvm)

2. Convert the loader and parameter into a format that can be operated by mtd, assuming that the block size is 128K.

   `./tools/linux/programmer_image_tool/programmer_image_tool -i update.img -b 128 -p 2 -t spinand -o out` Convert parameter to gpt.img, Miniloader to idblock.img. The size of gpt.img is 128K, and the size of idblock.img is 128K.

   Note: Programmer_image_tool cannot be converted to parameters alone, it needs to be packaged into update.img and then converted at one time. The loader can be converted separately.

3. Push the generated gpt.img and idblock.img to the board through adb, use the mtd_debug tool to erase and then write to complete the corresponding content update; The following commands all assume that the flash block is 128K, gpt.img and idblock.img are located in the loader partition and belong to mtd0, gpt is located in the 0th block, and idblock is located in the 1st-6th block, There are a total of 6 block sizes, so the offsets of the two are 0x0 and 0x20000 respectively (the offset value is the offset relative to mtd0). If the block size is 256K, the corresponding offset and size can be modified;

   ```
   mtd_debug erase dev/mtd0 0x0 0x20000
   mtd_debug write dev/mtd0 0x0 0x20000 userdata/gpt.img
   mtd_debug erase dev/mtd0 0x20000 0x60000
   mtd_debug write dev/mtd0 0x20000 0x60000 userdata/idblock.img
   ```

4. The idblock partition needs to be backed up at least twice to prevent the loss of the loader due to power failure during writing. Therefore, after the first idblock is successfully written, another copy is written to the subsequent address.

   Please note: When writing a backup idblock, the address of the 7th block cannot be exceeded, the idblock is only saved in blocks 1st-6th, and the end address is required to be block 7. If the block is 256K, it should be noted that the size of the loader partition allocated in the parameter must be 2M.

   The command also assumes the block 128K case, and the block 256K please modify the corresponding offset and size by yourself:

   ```
   mtd_debug erase dev/mtd0 0x80000 0x60000
   mtd_debug write dev/mtd0 0x80000 0x60000 userdata/idblock.img
   ```

9.23 How to reduce the size of uboot/boot

A: The NVR SDK optimizes the size of uboot and boot by default.

1. Uboot:

   It is suggested to crop out unwanted CMDs. If you do not need to support logo display under uboot, you can remove the configuration such as CONFIG_DRM_ROCKCHIP=y, and you can also enable gzip compression, which can significantly reduce the size of the uboot image. How to crop uboot into 1M, refer to the method as follow:

   1.1 Open the compression under uboot and modifies the rkbin directory:

```
huangjc@S1-GITSER-144:~/rk3588_nvr_sdk/rkbin$ git diff
diff --git a/RKTRUST/RK3588TRUST.ini b/RKTRUST/RK3588TRUST.ini
index 91ef1f1..11aa4e9 100644
--- a/RKTRUST/RK3588TRUST.ini
+++ b/RKTRUST/RK3588TRUST.ini
@@ -13,3 +13,5 @@ SEC=0
 SEC=0
 [OUTPUT]
 PATH=trust.img
+[COMPRESSION]
+COMPRESSION=gzip
```

   1.2 Modify the size of the uboot package and the numbers of backups(the numbers of backups can be configured by the user, and double backups are recommended):

```
diff --git a/configs/rk3588_defconfig b/configs/rk3588_defconfig
index 13a0de6..dbc7852 100644
--- a/configs/rk3588_defconfig
+++ b/configs/rk3588_defconfig
@@ -220,3 +220,5 @@ CONFIG_RK_AVB_LIBAVB_USER=y
CONFIG_OPTEE_CLIENT=y
CONFIG_OPTEE_V2=y
CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION=y
+CONFIG_SPL_FIT_IMAGE_KB=1024
+CONFIG_SPL_FIT_IMAGE_MULTIPLE=2
```

   1.3 Regenerate loader and uboot:

   Enter the u-boot directory, run ./make.sh rk3588 --spl-new , rk3588_spl_loader_v1.05.109.bin and uboot.img will be generated in the u-boot directory.

   Note: Loader must be replaced, because the loader of the default SDK traverses uboot according to the size of 2M.

2. Boot:

   Crop out unwanted peripheral drivers, such as: wifi/bt, display, touch screen.

   1. Some debugging options under kernel hacking can be remove, which can save more space. Note: After removing these options, all ko must be recompiled, otherwise there will be problems such as crashes caused by mismatching boot ko.

   These listed below can be used as a reference to remove, please combine with the actual project to select.

```
Kernel hacking  --->
    [*] Collect scheduler debugging info
    [*] Collect scheduler statistics
Lock Debugging (spinlocks, mutexes, etc...)  --->
    [*] Spinlock and rw-lock debugging: basic checks
[*] Verbose BUG() reporting (adds 70K)
[*] Debug credential management
[*] Tracers  --->
[*] Runtime Testing  --->
```

2. The file system will take up a lot of space. You can consider removing some unused file systems under the file system.

```
File systems  --->
```

3. Other peripherals that are not used need to be cut according to the actual project, the SDK has submitted the cutting reference configuration of the NVR Demo board by default, you can view the kernel configuration below:

```
arch/arm64/configs/rk3588_nvr.config
```

4. The kernel uses the lzma compressed image instead, which has a higher compression rate and is about 6M less than the lz4 compressed image by default. The reference modification is as follows:

The kernel compilation script is modified to lzma:

```
diff --git a/arch/arm64/Makefile b/arch/arm64/Makefile
index 7e2f0c3..30c2492 100644
--- a/arch/arm64/Makefile
+++ b/arch/arm64/Makefile
@@ -211,9 +211,9 @@ MAKE_MODULES ?= y
 %.img:
 ifeq ("$(CONFIG_MODULES)$(MAKE_MODULES)$(srctree)","yy$(objtree)")
-       $(Q)$(MAKE) rockchip/$*.dtb Image.lz4 modules
+       $(Q)$(MAKE) rockchip/$*.dtb Image.lzma modules
 else
-       $(Q)$(MAKE) rockchip/$*.dtb Image.lz4
+       $(Q)$(MAKE) rockchip/$*.dtb Image.lzma
 endif
        $(Q)$(srctree)/scripts/mkimg --dtb $*.dtb
diff --git a/scripts/mkimg b/scripts/mkimg
index d3e8d0c..643ea3d 100755
--- a/scripts/mkimg
+++ b/scripts/mkimg
@@ -58,7 +58,7 @@ if [ "${ARCH}" == "arm" ]; then
        ZIMAGE=zImage
 else
        DTB_PATH=${objtree}/arch/arm64/boot/dts/rockchip/${DTB}
-       ZIMAGE=Image.lz4
+       ZIMAGE=Image.lzma
 fi
 KERNEL_ZIMAGE_PATH=${objtree}/arch/${ARCH}/boot/${ZIMAGE}
 KERNEL_ZIMAGE_ARG="--kernel ${KERNEL_ZIMAGE_PATH}"
@@ -218,7 +218,7 @@ make_fit_boot_img()
        cp -a resource.img ${OUT}/resource
```

```
30
31          if [ "${ARCH}" == "arm64" ]; then
32  -               sed -i -e 's/arch = ""/arch = "arm64"/g' -e
    's/compression = ""/compression = "lz4"/' ${ITS}
33  +               sed -i -e 's/arch = ""/arch = "arm64"/g' -e
    's/compression = ""/compression = "lzma"/' ${ITS}
34          else
35                  sed -i -e 's/arch = ""/arch = "arm"/g' -e 's/compression
    = ""/compression = "none"/' ${ITS}
36          fi
```

Modify the kernel image path in the compilation script:

```
1   huangjc@S1-GITSER-144:~/rk3588_nvr_sdk/build$ git diff
2   diff --git a/tools/build_spi_nand.sh b/tools/build_spi_nand.sh
3   index bd11bd4..ad97b14 100755
4   --- a/tools/build_spi_nand.sh
5   +++ b/tools/build_spi_nand.sh
6   @@ -24,7 +24,7 @@ export RK_KERNEL_DTS=rk3588-nvr-demo-v10-spi-nand
7    # boot image type
8    export RK_BOOT_IMG=zboot.img
9    # kernel image path
10  -export RK_KERNEL_IMG=kernel/arch/arm64/boot/Image.lz4
11  +export RK_KERNEL_IMG=kernel/arch/arm64/boot/Image.lzma
12   # kernel image format type: fit(flattened image tree)
13   export RK_KERNEL_FIT_ITS=zboot.its
14   # parameter for GPT table
15  diff --git a/tools/build_emmc.sh b/tools/build_emmc.sh
16  index 0a00936..378fb63 100755
17  --- a/tools/build_emmc.sh
18  +++ b/tools/build_emmc.sh
19  @@ -24,7 +24,7 @@ export RK_KERNEL_DTS=rk3588-nvr-demo-v10
20   # boot image type
21   export RK_BOOT_IMG=zboot.img
22   # kernel image path
23  -export RK_KERNEL_IMG=kernel/arch/arm64/boot/Image.lz4
24  +export RK_KERNEL_IMG=kernel/arch/arm64/boot/Image.lzma
25   # kernel image format type: fit(flattened image tree)
26   export RK_KERNEL_FIT_ITS=zboot.its
27  diff --git a/tools/zboot.its b/tools/zboot.its
28  index c873ba2..a3099a1 100644
29  --- a/tools/zboot.its
30  +++ b/tools/zboot.its
31  @@ -22,11 +22,11 @@
32           };
33
34           kernel {
35  -            data = /incbin/("kernel/arch/arm64/boot/Image.lz4");
36  +            data = /incbin/("kernel/arch/arm64/boot/Image.lzma");
37               type = "kernel";
38               arch = "arm64";
39               os = "linux";
40  -            compression = "lz4";
41  +            compression = "lzma";
42               entry = <0xffffff01>;
43               load = <0xffffff01>;
```

9.24 How to check the pin multiplexing configuration

A: `cat sys/kernel/debug/pinctrl/pinctrl-rockchip-pinctrl/pinmux-pins`

Q: How to switch device and host for USB3.0 OTG port.

A: The default USB3.0 OTG port is otg mode.

Switch to host: `echo host > /sys/devices/platform/fd5d0000.syscon/fd5d0000.syscon:usb2-phy@0/otg_mode`

Switch to device: `echo peripheral > /sys/devices/platform/fd5d0000.syscon/fd5d0000.syscon:usb2-phy@0/otg_mode`

9.25 How to mount NFS

A: Depend on: sbin/mount.nfs, sbin/mount.nfs4, sbin/umount.nfs, sbin/umount.nfs4, ./usr/lib/libtirpc.so.3

These libraries can be found in the rootfs provided by the NVR SDK; the kernel config provided by the NVR SDK supports the NFS function by default.

Mount commands:

```
Linux :
mount -t nfs -o nolock 10.12.201.5:/nfs  /mnt/nfs  或者 mount -t nfs -o nolock,nfsvers=3,vers=3 10.12.201.5:/nfs  /mnt/nfs

Window:
mount \\10.12.201.15\nfs   I:\
```

9.26 How to do GDB debugging

A: The GDB debugging notes:

1. The rootfs released by the NVR SDK supports GDB by default: build/rootfs/usr/bin/gdb.
2. Start gdb debugging by gdb xxx or gdb attach pid. For specific commands, please refer to the network information.

   Tips for grabbing all thread stacks: thread apply all bt full
3. GDB ignores signal handling:

   handle SIGPIPE nostop noprint

   handle SIGUSR2 nostop noprint

   handle SIG32 nostop noprint

   handle SIG34 nostop noprint

   set print pretty on

9.27 Rootfs is read only, how to link new library files

A: The dynamic library search path specified by setting the environment variable LD_LIBRARY_PATH; (a temporary environment variable can be added with the export LD_LIBRARY_PATH="NEWDIRS" command).

For example: export LD_LIBRARY_PATH='/usr/local/lib:/nfs/tcpdump'

Note: The path of the library configured by LD_LIBRARY_PATH has a higher search priority than /usr/lib under the default rootfs when linking, so it can be used to temporarily verify some libraries.

9.28 I2C interface i2c_master_send failed to send large data (eg 24KB)

A: The timeout time in 2c-rk3x.c is changed from 1s to 3s.

9.29 Enable HDMI IN

A: You can open the HDMI IN configuration test as follows:

1. The kernel opens the configuration and confirms the CMA memory.

```
--- a/arch/arm64/configs/rk3588_nvr.config
+++ b/arch/arm64/configs/rk3588_nvr.config
@@ -32,3 +32,5 @@ CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768
CONFIG_MD_RAID456=y
CONFIG_MD_MULTIPATH=y
+
+CONFIG_VIDEO_ROCKCHIP_HDMIRX=y

#256M CMA memory is reserved in dts
#The EVB1 board is configured by default, and the NVR board needs to be manually configured, refer kernel/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
/* If hdmirx node is disabled, delete the reserved-memory node here. */
        reserved-memory {
                #address-cells = <2>;
                #size-cells = <2>;
                ranges;

                /* Reserve 256MB memory for hdmirx-controller@fdee0000 */
                cma {
                        compatible = "shared-dma-pool";
                        reusable;
                        reg = <0x0 (256 * 0x100000) 0x0 (256 * 0x100000)>;
                        linux,cma-default;
                };
        };
```

2. Enable hdmirx in dts and reserve 256M CMA memory

The EVB1 board is configured by default, no need to modify, the NVR board dts needs to be manually configured:

```
--- a/arch/arm64/boot/dts/rockchip/rk3588-nvr-demo.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-nvr-demo.dtsi
@@ -9,6 +9,21 @@
#include "rk3588-rk806-single.dtsi"

/ {
+       /* If hdmirx node is disabled, delete the reserved-memory node here. */
+       reserved-memory {
+               #address-cells = <2>;
+               #size-cells = <2>;
+               ranges;
+
+               /* Reserve 256MB memory for hdmirx-controller@fdee0000 */
+               cma {
+                       compatible = "shared-dma-pool";
+                       reusable;
+                       reg = <0x0 (256 * 0x100000) 0x0 (256 * 0x100000)>;
+                       linux,cma-default;
+               };
+       };
+
    i2s0_sound: i2s0-sound {
        status = "okay";
        compatible = "simple-audio-card";

--- a/arch/arm64/boot/dts/rockchip/rk3588-nvr-demo.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-nvr-demo.dtsi
@@ -279,6 +279,15 @@
        status = "okay";
     };

+/* Should work with at least 128MB cma reserved above. */
+&hdmirx_ctrler {
+       status = "okay";
+
+       /* Effective level used to trigger HPD: 0-low, 1-high */
+       hpd-trigger-level = <1>;
+       hdmirx-det-gpios = <&gpio2 13 GPIO_ACTIVE_LOW>;
+};
+
&i2c0 {
        status = "okay";
        pinctrl-names = "default";
```

The CMA memory reserved for HDMIIN can also directly modify the size of default reserved shared-dma-pool in arch/arm64/boot/dts/rockchip/rk3588-linux.dtsi.

3. Confirm whether HDMI IN is enabled and grab a image:

```
#Note: The NVR board may be video0, the EVB1 board may be video17
v4l2-ctl --verbose -d /dev/video17 -D
```

```
You can see the rk_hdmirx information as follows:
VIDIOC_QUERYCAP: ok
Driver Info:
        Driver name      : rk_hdmirx
        Card type        : rk_hdmirx
        Bus info         : fdee0000.hdmirx-controller
        Driver version   : 5.10.66
        Capabilities     : 0x84201000
                Video Capture Multiplanar
                Streaming
                Extended Pix Format
                Device Capabilities
        Device Caps      : 0x04201000
                Video Capture Multiplanar
                Streaming
                Extended Pix Format

#Snapshot reference command (pixel format, resolution, output path, etc. need to be configured according to the actual situation):
#Note: first use v4l2-ctl -d /dev/video17 --get-fmt-video to query the resolution and format of hdmiin input, and then configure it into the capture command parameters. The resolution and format of the snapshot must be consistent with the format input by hdmiin.
v4l2-ctl -d /dev/video17 --set-fmt-video=width=3840,height=2160,pixelformat='RGB3' --stream-mmap=4 --stream-skip=0 --stream-to=/data/4kp60_rgb24_1.yuv --stream-count=5 --stream-poll

#Supported formats:
v4l2-ctl -d dev/video0 --list-formats

#Status detection:
v4l2-ctl -d /dev/video17 --poll-for-event=ctrl=power_present

#EDID query:
#Direct output screen:
v4l2-ctl -d /dev/video17 --get-edid=pad=0  --fix-edid-checksums
#Save to file:
v4l2-ctl -d /dev/video17 --get-edid=pad=0,file=/data/edid  --fix-edid-checksums

#EDID setting:
v4l2-ctl -d /dev/video17 --set-edid=pad=0,file=/data/edid --fix-edid-checksums

#Query input signal format:
v4l2-ctl -d /dev/video17 --get-fmt-video
```

9.30 Check the SDK version

A: In addition to checking the SDK version by XML in the release document, you can read the SDK version information by visiting the following nodes in firmware:

```
cat /proc/sdk_service/version
```

9.31 RK3588 slave supports RAMBOOT

A: Follow these steps:

1. Modify the uboot configuration to rk3588-ramboot.config, and recompile uboot:

```
./make.sh rk3568-ramboot --sz-uboot 2048 1 --sz-trust 1024 1
```

Then get u-boot/uboot.img, u-boot/rk3588_ramboot_loader_v1.06.106.bin, u-boot/trust.img.

2. The host is connected to usbhost, the slave is connected to usb otg, the slave enters the loader mode, put the firmware of the slave into the host and then run the following commands:

```
# If the device is already in maskrom mode, you don't need to run this
command
./upgrade_tool/upgrade_tool_ramboot  rd 3
# Program loader, uboot, trust to the slave
./upgrade_tool/upgrade_tool_ramboot  db  rk3588_ramboot_loader_v1.06.106.bin
./upgrade_tool/upgrade_tool_ramboot  wl 0x2000 uboot.img
./upgrade_tool/upgrade_tool_ramboot  wl 0x42000 trust.img
# If the boot.img is not delivered from the host, you don't need to run this
command
./upgrade_tool/upgrade_tool_ramboot  wl 0x80000 boot.img
# Start slave firmware
./upgrade_tool/upgrade_tool_ramboot  run 0x2000 0x42000 0x80000 uboot.img
trust.img boot.img
```

9.32 Enable the hardware security module

A: Add configuration enable in dts:

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3588-linux.dtsi
b/arch/arm64/boot/dts/rockchip/rk3588-linux.dtsi
index 3860b95..657e163 100644
--- a/arch/arm64/boot/dts/rockchip/rk3588-linux.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-linux.dtsi
@@ -46,6 +46,14 @@
                status = "okay";
        };

+       firmware {
+               optee: optee {
+                       compatible = "linaro,optee-tz";
+                       method = "smc";
+               };
+       };
+
        ramoops: ramoops@110000 {
                compatible = "ramoops";
                reg = <0x0 0x110000 0x0 0xf0000>;
```

Note: OPTEE only supports CMA continuous physical memory, you need to adjust the size of the reserved CMA according to the actual situation (directly modify the size of the default reserved shared-dma-pool in rk3588-linux.dtsi). Please refer to the document for details:

```
<sdk>/docs/en/Linux/Security/Rockchip_Developer_Guide_TEE_SDK_EN.pdf
<sdk>/docs/en/Linux/Security/Rockchip_Developer_Guide_Crypto_HWRNG_EN.pdf
```

9.33 CPU accessing the DDR address space above 4G in UBOOT

A: Uboot only maps the DDR address space of 4G by default (but it is passed to the kernel according to the actual space size), and the address PCIe other than 4G can still be accessed.

If the CPU wants to map 4-8g addresses, it needs to be modified as follows:

```
diff --git a/arch/arm/mach-rockchip/rk3588/rk3588.c b/arch/arm/mach-rockchip/rk3588/rk3588.c
index 32c0493915..be44e4f99d 100644
--- a/arch/arm/mach-rockchip/rk3588/rk3588.c
+++ b/arch/arm/mach-rockchip/rk3588/rk3588.c
@@ -86,6 +86,12 @@ static struct mm_region rk3588_mem_map[] = {
      PTE_BLOCK_NON_SHARE |
      PTE_BLOCK_PXN | PTE_BLOCK_UXN
   },  {
+   .virt = 0x100000000UL,
+   .phys = 0x100000000UL,
+   .size = 0x100000000UL,
+   .attrs = PTE_BLOCK_MEMTYPE(MT_NORMAL) |
+     PTE_BLOCK_INNER_SHARE
+ }, {
    .virt = 0x900000000,
    .phys = 0x900000000,
    .size = 0x150000000,
--
```

9.34 Enable MessageQueue

A: MessageQueue is not enabled by default in the kernel. If necessary, open the following configuration in the kernel:

```
CONFIG_POSIX_MQUEUE=y
```

Otherwise, MQ_ Open() will report an error and return to enosys.

9.35 Check the connection status of HDMI, DP, etc

```
#Check the connection status of hdmi1,hdmi2 is card0-HDMI-A-2
cat /sys/class/drm/card0-HDMI-A-1/status
#Check the connection status of dp1,hdmi2 iscard0-DP-2
cat /sys/class/drm/card0-DP-1/status
```

9.36 Modify SATA3 from 6G to 3G

A: Refer to the following modifications:

```
diff --git a/drivers/phy/rockchip/phy-rockchip-naneng-combphy.c
b/drivers/phy/rockchip/phy-rockchip-naneng-combphy.c
index aa86fcc766fd..4aa7b6e258c4 100644
--- a/drivers/phy/rockchip/phy-rockchip-naneng-combphy.c
+++ b/drivers/phy/rockchip/phy-rockchip-naneng-combphy.c
@@ -806,8 +806,8 @@ static const struct rockchip_combphy_grfcfg
rk3588_combphy_grfcfgs = {
        .con2_for_sata          = { 0x0008, 15, 0, 0x00, 0x80c1 },
        .con3_for_sata          = { 0x000c, 15, 0, 0x00, 0x0407 },
        /* pipe-grf */
-       .pipe_con0_for_sata     = { 0x0000, 11, 5, 0x00, 0x22 },
-       .pipe_con1_for_sata     = { 0x0000, 2, 0, 0x00, 0x2 },
+       .pipe_con0_for_sata     = { 0x0000, 11, 5, 0x00, 0x11 },
+       .pipe_con1_for_sata     = { 0x0000, 2, 0, 0x00, 0x1 },
};
```

9.37 Enable UVC

A: You can open the UVC configuration test as follows:

1. Confirm that the kernel UVC configuration is open and the SDK has included the following submission:

```
commit 04a5209bf6a228d6ab616eb566e08cd4276f44e8
Author: Mark Huang <huangjc@rock-chips.com>
Date:   Fri May 13 09:04:31 2022 +0800

    usb: support uvc function

    1.update rk3588_nvr.config for uvc
    2.sync usb driver from develop-5.10 branch:
    commit c7bcfa4a886662a4238be9d40cfc566e860b72f4
    Author: William Wu <william.wu@rock-chips.com>
    Date:   Wed Apr 27 18:35:15 2022 +0800

        usb: dwc3: gadget: properly handle miss isoc event

        If miss isoc event happens, the current code just set
        the req status to -EXDEV and giveback the req to the usb
        gadget driver, and then stop the active transfer with the
        cmd DWC3_DEPCMD_ENDTRANSFER and wait for a XferNotReady
        event to restart a transfer again. However, for isoc
        ep in transfer, it cause to lost the isoc data of the
        req.

        This patch moves the miss isoc req to pending_list in
        order to restart transfer immediately instead of give
        back the req to the usb gadget driver.

        Signed-off-by: William Wu <william.wu@rock-chips.com>
        Change-Id: Idf38d9fd4d483854473c18f792d1996fb5fcab4b
```

```
29
30      Signed-off-by: Mark Huang <huangjc@rock-chips.com>
```

2. Make rk_mpi_uvc demo program， demo in build\app\rkmpi_release\example\rk_mpi_uvc, bin will be generated when MPI is compiled.
3. Copy rkuvc ini、usb_ config.sh and uvc_mpi_cfg.conf to /data in machine, and the boot script is started in the reference mode:

```
1  #RkLunch.sh
2  #!/bin/sh
3  post_chk()
4  {
5      export rt_log_level=3
6      export rk_mpi_uvc_log_level=5
7      export uac_app_log_level=2
8
9      touch /tmp/uvc_no_timeout
10     #default uvc+adb
11     /data/usb_config.sh #uac1
12     ifconfig lo 127.0.0.1
13
14     /data/rk_mpi_uvc &
15 }
16 start_app()
17 {
18 #The purpose here is to remove the ADB configured in the default rootfs
   and reconfigure it as a composite device of ADB and UVC through USB
   script
19   if [ -f "/etc/init.d/S50usbdevice" ];then
20      mv /etc/init.d/S50usbdevice /etc/S50usbdevice -f
21      killall -9 adbd
22      rm -rf /sys/kernel/config/usb_gadget/rockchip/configs/b.1/f*
23      echo none > /sys/kernel/config/usb_gadget/rockchip/UDC
24      rmdir /sys/kernel/config/usb_gadget/rockchip/functions/ffs.adb
25      UDC=`ls /sys/class/udc/| awk '{print $1}'`
26      echo $UDC  > /sys/bus/platform/drivers/dwc3/unbind
27      echo $UDC  > /sys/bus/platform/drivers/dwc3/bind
28   fi
29   echo "##### start uvc app ..."
30   post_chk &
31 }
32 ulimit -c unlimited
33 echo "/data/core-%p-%e" > /proc/sys/kernel/core_pattern
34
35 start_app
36
```

4. The host side uses amcap or portplayer programs, selects the preview UVC device, selects the resolution in YUV format, and opens the preview to see the following color output interface:

Note: the default demo uses the YUV data produced to send UVC preview. If you need to collect ISP images to send UVC preview, you can refer to the VI method of MPI and the UVC data interface in demo to send preview.

5. If you need the composite UAC device function, you can enable the configuration support as follows, and you need to confirm that the SDK has been updated to V1.4.0 or above:

    1. Open the uac application compilation configuration:

```
huangjc@S1-GITSER-144:~/rk3588_nvr_sdk/build/app/RKMPI_Release/example/rk_mpi_uvc$ git diff .
diff --git a/app/RKMPI_Release/example/rk_mpi_uvc/CMakeLists.txt b/app/RKMPI_Release/example/rk_mpi_uvc/CMakeLists.txt
index bb0a75a..e6ab0c9 100755
--- a/app/RKMPI_Release/example/rk_mpi_uvc/CMakeLists.txt
+++ b/app/RKMPI_Release/example/rk_mpi_uvc/CMakeLists.txt
@@ -25,7 +25,7 @@include_directories(${PROJECT_SOURCE_DIR}/include
${CMAKE_SYSROOT}/usr/include/libdrm)

-option(COMPILE_FOR_UVC_UAC "compile for uac" OFF)
+option(COMPILE_FOR_UVC_UAC "compile for uac" ON)
 if(COMPILE_FOR_UVC_UAC)
   include(uac/uac.cmake)
 endif()
```

    2. Modify the usb configuration script and integrate it into the firmware test according to the 2nd and 3rd steps of UVC above:

```
huangjc@S1-GITSER-144:~/rk3588_nvr_sdk/build/app/RKMPI_Release/example/rk_mpi_uvc$ git diff RkLunch.sh
diff --git a/app/RKMPI_Release/example/rk_mpi_uvc/RkLunch.sh b/app/RKMPI_Release/example/rk_mpi_uvc/RkLunch.sh
index d929444..922be06 100755
--- a/app/RKMPI_Release/example/rk_mpi_uvc/RkLunch.sh
+++ b/app/RKMPI_Release/example/rk_mpi_uvc/RkLunch.sh
@@ -17,7 +17,7 @@ check_usb_state()
        UDC=`ls /sys/class/udc/| awk '{print $1}'`
        echo $UDC  > /sys/bus/platform/drivers/dwc3/unbind
        echo $UDC  > /sys/bus/platform/drivers/dwc3/bind
-       /data/usb_config.sh #uac1
+       /data/usb_config.sh uac1
           /data/rk_mpi_uvc &
@@ -32,7 +32,7 @@ post_chk()
     touch /tmp/uvc_no_timeout
        #default uvc+adb
-    /data/usb_config.sh #uac1
+    /data/usb_config.sh uac1
     ifconfig lo 127.0.0.1
diff --git a/app/RKMPI_Release/example/rk_mpi_uvc/rkuvc.ini b/app/RKMPI_Release/example/rk_mpi_uvc/rkuvc.ini
index 3b5f1ea..8802a40 100755
--- a/app/RKMPI_Release/example/rk_mpi_uvc/rkuvc.ini
+++ b/app/RKMPI_Release/example/rk_mpi_uvc/rkuvc.ini
@@ -2,7 +2,7 @@
 enable_aiq = 0
 enable_vo = 0
 enable_npu = 0
-enable_uac = 0
+enable_uac = 1
 [isp.0.adjustment]
 contrast    = 50
```

Note: The default compound uac1, if you need to support uac2, just change the heel parameter in the above script to uac2 directly, the recording and playback configuration is controlled in the uac configuration in the usb_config.sh script, you can modify the descriptor configuration as needed.

9.38 Enable AIQ

You can get camera image as follows, for example imx415

1. Enable the follow configs in kernel

```
CONFIG_VIDEO_ROCKCHIP_CIF=y
CONFIG_VIDEO_ROCKCHIP_ISP=y
CONFIG_VIDEO_ROCKCHIP_ISPP=y
```

2. Enable the camera config, at first make sure the kernel have the camera driver

```
CONFIG_VIDEO_IMX415=y
```

3. Config dts

```
arch/arm64/boot/dts/rockchip/rk3588-evb1-imx415.dtsi
```

4. Put the camera iq file in the follow folder

```
3588-nvr/build/app/rkaiq_3A_server/iqfiles$ ls
imx415_CMK-OT2022-PX1_IR0147-50IRC-8M-F20.json
```

5. Enable the 3A SERVER config in build_emmc.sh

```
3588-nvr$ vi build_emmc.sh
# pack 3A_server in rootfs
export RK_3A_SERVER_IN_ROOTFS=true
```

6. Use the ps -ef command to make sure the rkaiq_3A_server is running after boot the system

```
[root@RK3588:/]# ps -ef
root       656     1  0 12:00 ?        00:00:00 /bin/sh -c
/usr/bin/rkaiq_3A_ser
```

7. How to get camera image

   Use media-ctl -p -d /dev/media0 to get sensor resolution, if no camera info, check 1 2 3 chapter

```
[root@RK3588:/]# media-ctl -p -d /dev/media0                    //
- entity 48: m00_b_imx415 5-001a (1 pad, 1 link)
      type V4L2 subdev subtype Sensor flags 0
      device node name /dev/v4l-subdev2
  pad0: Source
          [fmt:SGBRG10_1X10/3864x2192@10000/300000 field:none
           crop.bounds:(12,16)/3840x2160]
          -> "rockchip-csi2-dphy0":0 [ENABLED]
```

   Use media-ctl -p -d /dev/media1 to find rkisp_mainpath node, and the video node with it

```
[root@RK3588:/]# media-ctl -p -d /dev/media1                    //
- entity 6: rkisp_mainpath (1 pad, 1 link)
      type Node subtype V4L flags 0
      device node name /dev/video8
  pad0: Sink
          <- "rkisp-isp-subdev":2 [ENABLED]
```

   Use v4l2-ctl Command to get sensor image, replace the video node and resolution

```
v4l2-ctl -d /dev/video8  --set-fmt-
video=width=3840,height=2160,pixelformat=NV12  --stream-mmap=3  --stream-
skip=10  --stream-to=/tmp/2160p.yuv  --stream-count=1  --stream-poll
```

9.39 Multi-VP sync, used for sync of multiple output ports of the same CPU

1. RK3588 NVR SDK should update to V1.3.0 or higher.
2. Multi-VP sync interface.

   1）Directly operate the kernel node: `echo 1 2 > sys/kernel/debug/dri/0/video_port0/vp_sync` //vp1 and vp2 sync to vp0

   2）MPI interface :

```
#define BIT(x) (1 << x)
RK_U32 timeout = 10;
RK_U32 Devs = BIT(0) | BIT(2);//vp0 vp2 sync
while (timeout--) {
      RK_U32 Ret = RK_MPI_VO_SyncDevs(Devs);
      if (Ret){
            RK_LOGE("RK_MPI_VO_SyncDevs fail retry, timeout=%d",timeout);
            usleep(100000);
      }
      else{
            RK_LOGE("RK_MPI_VO_SyncDevs succeed");
            break;
      }
 }
```

Notes

1. The sync mode needs to be set after all voDevs that need to be synchronized are enabled. Only supports native HDMI, DP/eDP, BT656/BT1120 interface output sync for now.
2. If RK_MPI_VO_SetVcntTiming is set, need to set the sync mode after setting vcnt.

   If vcnt is set, the single-screen video cannot exceed 2 channels to ensure sync.

   If single-screen multi-channel video needs to be synchronized, need to configure vcnt to 0.
3. If RK_MPI_VO_SetHdmiParam() is set, need to usleep(1000*1000llu) after setting attribute; and then call the sync mode interface after vp is enabled.
4. The above method only ensures the sync of vp, the application layer needs to make sure the data sent to each vo is synchronized. To ensure sync, here are some suggestions:
   - Configure vdec to preview mode RK_MPI_VDEC_SetDisplayMode(u32Ch, VIDEO_DISPLAY_MODE_PREVIEW).
   - It is recommended to trigger the application to send frames to vo through vsync interrupt, and confirm that the time to send data to vo is in the first half of vsync (that is, the interval between vsync interrupt and sending data to vo cannot exceed vsync/2). It can use RK_MPI_VO_RegVsyncCallbackFunc to register vsync interrupt, or use the wait vbalk of drm to get the vsync interrupt callback. Note: In the vsync callback, the frame sending action cannot be performed directly, and time-consuming operations cannot be performed.
   - If the single-screen only outputs one full-screen image, it suggests to sent the layer pass-through mode stLayerAttr.bBypassFrame=RK_TRUE to reduce errors caused by intermediate links.

     Use RK_MPI_VO_SetChnRecvThreshold () to configure the VO CHN buffer number to 1 before enable chn.

- If single-screen multi-channel video splicing needs to be synchronized, need to configure vcnt to 0. That is, do not call RK_MPI_VO_SetVcntTiming() to configure vcnt, the default vcnt is 0. Use RK_MPI_VO_SetChnRecvThreshold() to configure VO CHN buffer number to 3 before enable chn. **Note:** Do not apply for the release of the buffer sent to vo frequently. It is recommended to use the bufferpool to recycle the buffer.

9.40 Synchronization between different RK3588 CPUs

1. RK3588 NVR SDK should update to V1.3.0 or higher.
2. Different CPUs need to enable vo at the same time to ensure sync during initialization.

   If different CPUs are on the same board, you can send the interrupt signal to every CPU through CPLD(or other hardware sync signal), and enable at the same time.

   If different CPUs are not on the same board, a mechanism for synchronizing multiple CPUs is required, such as using the network IEEE1588V2 (linuxPTP) precise clock synchronization protocol, etc. RK3568/RK3588 supports hardware timestamps, according to our self-test, the error can be controlled within 10us after using hardware timestamp synchronization.

   Control method: Implemented with vop2_crtc_enable() in rockchip_drm_vop2.c, customers can encapsulate the sync interface in kernel mode by themselves:

   ```
   disable crtc: echo 0 > /sys/kernel/debug/dri/0/video_portN/enable //This is blocking and will not return until standby takes effect;
   enable crtc: echo 1 > /sys/kernel/debug/dri/0/video_portN/enable //This is non-blocking, and the scan of the first line starts immediately after the standby is canceled;
   ```

3. Because the clock sources of different RK3588 CPUs is different, there may be a phase difference in VSYNC after running for a while, and vsync needs to be fine-tuned.
   - The sync detection about vsync:

     Suggestion: Each CPU monitors its own vsync callback time, calculates the vsync time per unit time. The slave makes corresponding adjustments according to the difference between the master vsync time and the slave vsync time.

     Interface for obtaining vsync timestamp in user mode:

     Register vsync callback through Sample_VO_RegVsyncCallback to obtain vsync time.

     Obtaining vsync timestamp in kernel mode:

     The vsync interrupt processing is in void vop2_wb_handler(struct vop2_video_port *vp), the specific location is in the irqreturn_t vop2_isr(int irq, void *data) interrupt processing of rockchip_drm_vop2.c, and the accurate vsync timestamp can be obtained by calling ktime_get_real_ts64(struct timespec64 *ts) before calling vop2_wb_handler(vp).

     Since obtaining the vsync timestamp in user mode requires multiple callbacks, it is easy to bring errors due to system load and other influences. It is recommended to obtain vsync time in kernel mode.

   - The two native HDMIs use the HDMI PHY internal PLL, trimming as follows:

     ```
     echo 148500000 > /sys/kernel/debug/hdptxphy0/rate
     ```

     The adjusted result will not be reflected in clk_summary. Compare by checking the HDMI registers:

```
HDMI0:  cat /sys/kernel/debug/regmap/fed60000.hdmiphy-hdptx-combphy/registers
HDMI1:  cat /sys/kernel/debug/regmap/fed70000.hdmiphy-hdptx-combphy/registers
```

Follow the steps below to confirm whether hdmi uses phy clk or system clk

1. Confirm which vop is used by hdmi

hdmi 0 hangs on vop 2, `cat /sys/kernel/debug/dri/0/summary`

```
[root@RK3588:/]# cat /sys/kernel/debug/dri/0/summary
Video Port0: DISABLED
Video Port1: DISABLED
Video Port2: ACTIVE
Connector: HDMI-A-1
   bus_format[100a]: RGB888_1X24
   overlay_mode[0] output_mode[f] color_space[0], eotf:0
Display mode: 1920x1080p60
   clk[148500] real_clk[148500] type[40] flag[a]
   H: 1920 2008 2052 2200
   V: 1080 1084 1089 1125
Esmart2-win0: ACTIVE
   win_id: 9
   format: XR24 little-endian (0x34325258) SDR[0] color_space[0] glb_alpha[0xff]
   rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
   csc: y2r[0] r2y[0] csc mode[0]
   zpos: 2
   src: pos[0, 0] rect[1920 x 1080]
   dst: pos[0, 0] rect[1920 x 1080]
   buf[0]: addr: 0x0000000000000000 pitch: 7680 offset: 0
Video Port3: DISABLED
```

2. Check whether the dclk_vop is hung under the corresponding clk_hdmiphy_pixel.

dclk_vop2 hangs on clk_hdmiphy_pixel0, `cat /sys/kernel/debug/clk/clk_summary`

```
clk_hdmiphy_pixel0             2        3        0   148500000
   0
   dclk_vop2                     2        5        0   148500000
   0
```

dclk_vop hangs on clk_hdmiphy_pixel,the phy clk used by hdmi.

If it is hung under other clk, please apply the following modification patch in dts:

```
&display_subsystem {
   clocks = <&hdptxphy_hdmi_clk0>, <&hdptxphy_hdmi_clk1>;
   clock-names = "hdmi0_phy_pll", "hdmi1_phy_pll";
};

&hdptxphy_hdmi_clk0 {
   status = "okay";
};

&hdptxphy_hdmi_clk1 {
   status = "okay";
};
```

Notice:During the debugging process, it was found that not all frequencies could be adjusted correctly,Recommended frequency:

Standard frequency	Increase frequency	Reduce frequency
148.5M	148501200	148498900
297M	297002500	296998900 296997000
594M	594005000	593995000

- The other two vp interface use the system pll (pll_hpll) for adjustment, and uses the rockchip_pll_clk_compensation interface to adjust dclk.

  Use cat sys/kernel/debug/clk/clk_summary command to check the clock tree, check which PLL the dclk to be adjusted is under, and find the corresponding pll. After the adjustment, the frequency of clk_get_rate will be changed, or you can check the clock tree after setting:

```
cat sys/kernel/debug/clk/clk_summary
```

The reference code is as follows:

```
struct clk *clk = NULL, *clk1 = NULL;
int ret = 0, i = 0;

clk = __clk_lookup("aupll");
clk1 = __clk_lookup("pll_aupll");
if (clk == NULL)
    printk("---get aupll clk fail---\n");
if (clk1 == NULL)
    printk("---get pll_aupll clk fail---\n");
rockchip_pll_clk_compensation_reset(clk);
for (i = 1; i < 20; i++) {
    printk("clk name=%s,clk=%ld, clk1 name=%s,clk=%ld###\n",
           __clk_get_name(clk), clk_get_rate(clk),
__clk_get_name(clk1),
           clk_get_rate(clk1));
    ret = rockchip_pll_clk_compensation(clk, i * 50);
    printk("clk name=%s,clk=%ld, clk1 name=%s,clk=%ld###,ret=%d\n",
           __clk_get_name(clk), clk_get_rate(clk),
__clk_get_name(clk1),
           clk_get_rate(clk1), ret);
```

```
}

for (i = 1; i < 20; i++) {
    ret = rockchip_pll_clk_compensation(clk, -50 * i);
    printk("clk name=%s,clk=%ld, clk1 name=%s,clk=%ld###,ret=%d\n",
           __clk_get_name(clk), clk_get_rate(clk),
__clk_get_name(clk1),
           clk_get_rate(clk1), ret);
}

clk = __clk_lookup("v0pll");
clk1 = __clk_lookup("pll_v0pll");
if (clk == NULL)
    printk("---get v0pll clk fail---\n");
if (clk1 == NULL)
    printk("---get pll_v0pll clk fail---\n");
rockchip_pll_clk_compensation_reset(clk);
for (i = 1; i < 20; i++) {
    ret = rockchip_pll_clk_compensation(clk, i * 50);
    printk("clk name=%s,clk=%ld, clk1 name=%s,clk=%ld###,ret=%d\n",
           __clk_get_name(clk), clk_get_rate(clk),
__clk_get_name(clk1),
           clk_get_rate(clk1), ret);
}

for (i = 1; i < 20; i++) {
    ret = rockchip_pll_clk_compensation(clk, -50 * i);
    printk("clk name=%s,clk=%ld, clk1 name=%s,clk=%ld###,ret=%d\n",
           __clk_get_name(clk), clk_get_rate(clk),
__clk_get_name(clk1),
           clk_get_rate(clk1), ret);
}
```

Notes:

- The adjustment of rockchip_pll_clk_compensation() can only be adjusted based on the obtained reference frequency of clk, and the adjustment results cannot be accumulated.

  For example, calling rockchip_pll_clk_compensation(clk, 200) twice in a row results in 200ppm instead of 400ppm.

- The max number of pll which RK3588 can adjust is three. After each pll reference frequency change (for example, after switching the screen resolution), you need to call rockchip_pll_clk_compensation_reset(clk); and then call rockchip_pll_clk_compensation() to adjust.

  RK3568 vp1 mounts to vpll by default, vpll does not support fractional frequency division and cannot be adjusted. For synchronous splicing (vp0/vp1 output the same resolution), it is recommended to mount vp1 to hpll.

```
--- a/arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi
   @@ -519,8 +519,8 @@
   &vop {
     status = "okay";
-    assigned-clocks = <&cru DCLK_VOP1>;
-    assigned-clock-parents = <&cru PLL_VPLL>;
+/*   assigned-clocks = <&cru DCLK_VOP1>;
+    assigned-clock-parents = <&cru PLL_VPLL>;*/
     skip-ref-fb;
   };
```

- The adjustment interface rockchip_pll_clk_compensation() cannot be called in interrupt processing, because it may cause system scheduling, it is recommended to call it in the workqueue.
- t is recommended to shorten the adjustment period (for example: once a second), and make each adjustment smaller (for example: -2~2ppm each time). Each adjustment cycle needs to determine whether the pll parameters need to be adjusted (the adjustment parameters of the previous cycle are used as the benchmark to make appropriate modifications, and if no modification is required, the last pll adjustment parameters are maintained).
- The current method of adjusting dclk only verifies the native HDMI, DP/eDP, and BT656/BT1120 interfaces.

4. The above method only ensures the sync of vp, the application layer needs to make sure the data sent to each vo is synchronized. To ensure sync, here are some suggestions:
    - Configure vdec to preview mode RK_MPI_VDEC_SetDisplayMode(u32Ch, VIDEO_DISPLAY_MODE_PREVIEW).
    - It is recommended to trigger the application to send frames to vo through vsync interrupt, and confirm that the time to send data to vo is in the first half of vsync (that is, the interval between vsync interrupt and sending data to vo cannot exceed vsync/2).

      It can use RK_MPI_VO_RegVsyncCallbackFunc to register vsync interrupt, or use the wait vbalk of drm to get the vsync interrupt callback.

      Note: In the vsync callback, the frame sending action cannot be performed directly, and time-consuming operations cannot be performed.
    - If the single-screen only outputs one full-screen image, it suggests to sent the layer pass-through mode stLayerAttr.bBypassFrame=RK_TRUE to reduce errors caused by intermediate links.

      Use RK_MPI_VO_SetChnRecvThreshold () to configure the VO CHN buffer number to 1 before enable chn.

      In the pass-through mode, if the frame sent to VO is 30fps but the display output is 60fps (that is, two vsyncs send one frame of data), frequent out-of-sync phenomenon occurs, you can try adding a 3ms delay before RK_MPI_VO_SendFrame() Retest synchronicity.
    - If you want to ensure synchronization in the case of single-screen multi-channel video splicing, you need to configure vcnt to 0 to ensure synchronization. That is, do not call the RK_MPI_VO_SetVcntTiming() interface to configure VCNT, the default is 0. And before enabling chn, call the RK_MPI_VO_SetChnRecvThreshold() interface to configure the number of VO CHN caches to 4.

      Before enabling the layer, call the RK_MPI_VO_SetLayerDispBufLen() interface to set the buffer of the layer to 5. If the out-of-sync phenomenon occurs frequently when the frame sent by vo is already synchronized, you can try to add a 3ms delay before RK_MPI_VO_SendFrame() and then test the synchronization.

Note: The buffer sent to vo should not be released frequently. It is recommended to use the buffer decoded by vdec directly or use the buffer pool to recycle the buffer.

The frame rate of the layer and the decoding frame rate should be configured to be the same. For example, to decode 30fps, the layer frame rate should also be configured as 30fps.

9.41 VI-VENC-VDEC-VO path delay debug

1. RK3588 NVR SDK should update to V1.3.0 or higher.
2. Suggestions for optimizing display path delay:
   - Configure vdec to preview mode RK_MPI_VDEC_SetDisplayMode(u32Ch, VIDEO_DISPLAY_MODE_PREVIEW), and use RK_MPI_VO_SetChnRecvThreshold () to configure the VO CHN buffer number to 1 before enable chn.

     Note: At this time, the application needs to ensure the uniformity of the frame sent.
   - The frame rate of the layer should be configured to 60fps in order to decrease VSYNC interval, and be aware of GPU usage.
   - If the input source is VI, you can consider the VI source to increase the mipi transmission frequency in order to shorten the transmission time.
   - If there is encoding, VENC is changed to 60FPS, the purpose is to reduce the delay fluctuation.
   - If CPU/GPU is in running performance mode, please note power consumption and heat dissipation.
   - Enable pass-through mode, configure stLayerAttr.bBypassFrame = RK_TRUE before enable_layer. When the layer has only one channel, it can be displayed directly, reducing the intermediate splicing process.

     Note: If the ui and the video are on the same layer, the pass-through mode cannot be entered. You need to disable the ui channel before entering the pass-through mode. If entering pass-through mode, the GPU load should be very low, 0 in most cases.
   - Call RK_MPI_VO_SetVcntTiming(VoDev, 900) before RK_MPI_VO_Enable;//If the resolution is 1080P, the second parameter is recommended to 900. In other resolutions, the second parameter is estimated by the displayed height * 0.8, and adjusted to an optimal value through the test.

     Note: If the resolution is changed, the corresponding VCNT also needs to be modified, otherwise there will be not displayed.
   - VDEC decoding is configured as decoding order:

     stVdecParam.stVdecVideoParam.enOutputOrder = VIDEO_OUTPUT_ORDER_DEC.
   - The encoding and decoding uses the multi-slice mechanism, which is not yet supported.

9.42 Prompt egl initialization failed

Check the following two points:

1. Check whether the rootfs/lib/firmware/mali_csffw.bin file exists, if not, copy it from the SDK build/rootfs/lib/firmware/mali_csffw.bin;
2. Check whether there is libmali.so.1 library file in rootfs, if not, copy it to machine /usr/lib/libmali.so.1 in SDK build/app/RKMPI_Release/sdk/deps/lib/libmali.so.1.

9.43 CMA memory is set to application-specific

The CMA configured by default is shared with the system. If you want to configure the CMA as an application-specific one, you need to open the following configuration:

```
diff --git a/arch/arm64/configs/rk3588_nvr.config
b/arch/arm64/configs/rk3588_nvr.config
index 3ca9dc7..a59faa8 100644
--- a/arch/arm64/configs/rk3588_nvr.config
+++ b/arch/arm64/configs/rk3588_nvr.config
@@ -75,3 +75,7 @@ CONFIG_MD_RAID456=y
 CONFIG_RAID6_PQ=y
 CONFIG_RAID6_PQ_BENCHMARK=y
 CONFIG_XOR_BLOCKS=y
+
+CONFIG_CMA_INACTIVE=y
```

After the patch is applied, the CMA memory will not be counted in the total in meminfo. The use of CMA needs to be viewed through the CMA debugging node, that is, `CONFIG_CMA_DEBUGFS=y, CONFIG_CMA_DEBUG=y` is enabled in the config. For details, please refer to "docs/cn/Common/MEMORY/Rockchip_Developer_Guide_Linux_CMA_CN.pdf".

9.44 Configuration download 575 firmware instructions

Refer to the following patches, and configure PM firmware or PMX firmware download according to the actual situation.

```
diff --git a/arch/arm64/configs/rk3588_nvr.config
b/arch/arm64/configs/rk3588_nvr.config
index 3ca9dc7..a59faa8 100644
--- a/arch/arm64/configs/rk3588_nvr.config
+++ b/arch/arm64/configs/rk3588_nvr.config
@@ -75,3 +75,7 @@ CONFIG_MD_RAID456=y
 CONFIG_RAID6_PQ=y
 CONFIG_RAID6_PQ_BENCHMARK=y
 CONFIG_XOR_BLOCKS=y
+
+CONFIG_SATA_PMP_JMB575_FW_DOWNLOAD=y
+CONFIG_SATA_PMP_JMB575_PMX_FW=y
```

Note: The macro SATA_PMP_JMB575_PMX_FW does not need to be defined for the following 5 ports.

9.45 Enter fiq under the serial port to trigger the debugging mode, and how to modify the trigger value

drivers/soc/rockchip/rk_fiq_debugger.c, debug_getc returns FIQ_DEBUGGER_BREAK will trigger fiq debug.

```
static int debug_getc(struct platform_device *pdev)
{
    unsigned int lsr;
```

```
    struct rk_fiq_debugger *t;
    unsigned int temp;
    static unsigned int n;
    static char buf[32];

    t = container_of(dev_get_platdata(&pdev->dev), typeof(*t), pdata);
    /*
     * Clear uart interrupt status
     */
    rk_fiq_read(t, UART_USR);
    lsr = rk_fiq_read_lsr(t);

    if (lsr & UART_LSR_DR) {
        temp = rk_fiq_read(t, UART_RX);
        buf[n & 0x1f] = temp;
        n++;
        if (temp == 'q' && n > 2) {
            if ((buf[(n - 2) & 0x1f] == 'i') &&
                (buf[(n - 3) & 0x1f] == 'f'))
                return FIQ_DEBUGGER_BREAK;
            else
                return temp;
        } else {
            return temp;
        }
    }

    return FIQ_DEBUGGER_NO_CHAR;
}
```

9.46 Turn off the scheduler and enable eas

Canceling energy-saving scheduling does not limit CPU usage, can improve CPU performance but will increase power consumption.

```
echo 0 > /proc/sys/kernel/sched_energy_aware
```

9.47 Dynamically modify the node configuration in dts under uboot, unified firmware for different versions

Add the following function in u-boot/arch/arm/mach-rockchip/rk3588/rk3588.c, refer to the following code to change the node to be modified to the required configuration.

```
int rk_board_fdt_fixup(void *blob)
{
    int node;
    char *prop;

    node = fdt_path_offset(blob, "/pcie@fe150000");
    if (node >= 0) {
        prop = (char *)fdt_getprop(blob, node, "status", NULL);
            if (!prop)
```

```
                return 0;
        printf("pcie3 status is %s\n", prop);
        if (!strcmp(prop, "disabled")) {
            printf("pcie3 status is disabled, now fixed to okay\n");
            fdt_setprop_string(blob, node, "status", "okay");
            prop = (char *)fdt_getprop(blob, node, "status", NULL);
            printf("after set, now pcie3 status is %s\n", prop);
        }
    }
    return 0;
}
```

9.48 Support standby wake-up, need to modify as follows

The SDK version is required to be greater than or equal to V1.4.0.

rkbin:

```
diff --git a/RKBOOT/RK3588MINIALL.ini b/RKBOOT/RK3588MINIALL.ini
index 81e3da6..69ed2ee 100644
--- a/RKBOOT/RK3588MINIALL.ini
+++ b/RKBOOT/RK3588MINIALL.ini
@@ -26,7 +26,7 @@ RC4_OFF=true
 [BOOT1_PARAM]
 WORD_0=0x0
 WORD_1=0x0
-WORD_2=0x4
+WORD_2=0x0
 WORD_3=0x0
 WORD_4=0x0
 WORD_5=0x0
```

kernel:

```
--- a/arch/arm64/boot/dts/rockchip/rk3588-evb.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-evb.dtsi
@@ -11,7 +11,7 @@
 #include <dt-bindings/display/drm_mipi_dsi.h>
 #include <dt-bindings/display/rockchip_vop.h>
 #include <dt-bindings/sensor-dev.h>
-#include "rk3588-cpu-swap.dtsi"
+//#include "rk3588-cpu-swap.dtsi"

 / {
        adc_keys: adc-keys {
diff --git a/arch/arm64/boot/dts/rockchip/rk3588-nvr.dtsi
b/arch/arm64/boot/dts/rockchip/rk3588-nvr.dtsi
index f9d8277..0381a1a 100644
--- a/arch/arm64/boot/dts/rockchip/rk3588-nvr.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-nvr.dtsi
@@ -11,7 +11,7 @@
 #include <dt-bindings/display/drm_mipi_dsi.h>
 #include <dt-bindings/display/rockchip_vop.h>
 #include <dt-bindings/sensor-dev.h>
-#include "rk3588-cpu-swap.dtsi"
```

```
+//#include "rk3588-cpu-swap.dtsi"

 / {
        adc_keys: adc-keys {
```

9.49 Uboot user mode writes ext4 image, only ext4 in uncompressed format can be written

1. The image needs to decompress simg2img and then mmc write or dd to write to the partition. Note that when packaging the ext 4 image, do not specify the size of the partition when specifying the size, a certain margin of the file size is required. You can refer to the packaging script build/tools/build.sh.
2. When mounting ext 4, resize it again. Reference system mount script build/rootfs/etc/init.d/S21mountall.sh. Resize before mounting, otherwise you need to support online resize.

9.50 Supports packaged recovery and misc partitions

The default is to turn off recovery partition packaging. You need to modify build_emmc.sh or build_spi_nand.sh to enable the function.

```
+++ b/tools/build_emmc.sh
@@ -31,7 +31,7 @@ export RK_KERNEL_IMG=kernel/arch/arm64/boot/Image.lz4
 # kernel image format type: fit(flattened image tree)
 export RK_KERNEL_FIT_ITS=zboot.its
 # recovery build
-export RK_RECOVERY=false
+export RK_RECOVERY=true
 # recovery image
 export RK_RECOVERY_IMG=recovery.img
 # Recovery image format type: fit(flattened image tree)
```

After turning on the function, recovery.img will be compiled in the build directory. After turning off recovery packaging, you need to delete the soft link file in the rockdev directory.

If you need to package it into update.img, you need to make the following modifications in the tools directory:

```
+++ b/linux/Linux_Pack_Firmware/rockdev/rk3588-package-file-nvr-emmc
@@ -6,11 +6,11 @@ bootloader    Image/MiniLoaderAll.bin
 parameter      Image/parameter.txt
 #trust         Image/trust.img
 uboot          Image/uboot.img
-#misc          Image/misc.img
+misc           Image/misc.img
 #resource      Image/resource.img
 #kernel                Image/kernel.img
 boot           Image/boot.img
-#recovery      Image/recovery.img
+recovery       Image/recovery.img
 rootfs         Image/rootfs.img
 #oem           Image/oem.img
 userdata       Image/userdata.img
```

9.51 Support VI-VENC--NET--VDEC-VO low-latency API and Demo

Related descriptions and documents can be viewed at: build/app/low_delay_net_display/

Compilation method:

```
./build_emmc.sh env
cd build/app/build
./build.sh ../low_delay_net_display
```

The compiled bin is in the build/app/bin directory