Rockchip RK356X Linux NVR SDK Quick Start

ID: RK-JC-YF-542

Release Version: V1.6.0

Release Date: 2023-02-03

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]mailto:fae@rock-chips.com)

Preface

Overview

Rockchip NVR SDK getting started and compilation guide.

Product Version

Chip Name	Kernel Version
RK356X	Linux 4.19

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	Author	Revision Date	Revision Description
V0.1.0	XYP	2021-4-1	Initial version
V1.0.0	XYP	2021-8-5	Release version, add FAQ
V1.1.0	XYP	2021-9-14	Update FAQ
V1.2.0	XYP	2021-10-29	Add FAQ and media FAQ document index
V1.3.0	XYP	2022-01-27	Modify the RKMPI directory and compile introduction
V1.3.1	XYP	2022-04-06	Fix uboot and display related issues
V1.4.0	XYP	2022-06-11	Fix PCIE3 signal compatibility issue and add debugfs Optimized display pass-through and sync modes
V1.5.0	XYP	2022-07-28	Add splicing sync pll, clk adjustment interface RKMPI version update to V1.7.2
V1.5.1	XYP	2022-11-11	update sdk download path
V1.6.0	XYP	2023-02-03	Update the flash driver Update the driver to fix the wrong source decoding problem, Support EDP hot plug function, update RKMPI V1.9.1

Content

1. Introduction

The NVR SDK is cropped based on RK356X complete linux SDK(rk356x_linux_release_v1.1.3_20210805.xml) , optimized for multi-video playback capabilities for NVR-type products.

The SDK download address:

```
repo init --repo-url https://gerrit.rock-chips.com:8443/repo-release/tools/repo \
-u https://gerrit.rock-chips.com:8443/linux/rockchip/platform/manifests \
-b linux -m rk356x_nvr_linux_lite.xml

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

In case of problems, please refer to `Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf`

2. 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 4.19.
  -- 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.
```


3. Software Update Records

The V1.6.0 version VS the V1.5.1 version, main updating the points as follows:

Number	Update Brief Description	A Detailed Description	Remark
1	upport Edp hot plug	[Problem description]Support Edp hot plug [Cause Analysis] Not involved [Solution] Not involved [Main modification file] kernel: drivers/gpu/drm/bridge/analogix/analogix_dp_core.c	Must Update
2	Update flash driver	[Problem description]Update flash driver,increase compatibility [Cause Analysis] Not involved [Solution] Not involved [Main modification file] kernel: drivers/rkflash/ u-boot: drivers/spi/、drivers/mtd/ rkbin: bin/rk35/	Must Update
3	Multimedia MPI version updated to V1.9.1	[Problem description] Multimedia MPI version updated to V1.9.1 [Cause Analysis] Not involved [Solution] Not involved [Main modification file] build/app/RKMPI_Release/	Must Update
4	Update mpp driver fix video dec error	[Problem description] Error source decoding probability anomaly [Cause Analysis] In case of error, decoder and iommu access hardware exception [Solution] Modify the process to handle abnormal access in a unified manner [Main modification file] kernel: drivers/video/rockchip/mpp	Must Update
5	Fix the abnormality after frequent down and up of eth	[Problem description] Frequent dump and up operations on eth lead to eth exceptions [Cause Analysis] Frequent operations lead to application without free completion [Solution] Modify the process to handle the free state [Main modification file] kernel： drivers/net/ethernet/stmicro/stmmac/stmmac_main.c	Must Update
6	Update crypto driver	[Problem description] Update crypto driver. [Cause Analysis] Not involved [Solution] Not involved [Main modification file] kernel：crypto/ kernel: drivers/crypto/rockchip/	Optional

Number	Update Brief Description	A Detailed Description	Remark
7	Fix vpll not correctly	[Problem description] When both vp0 and vp1 are bound to hpll, the frequency of vp1 is wrong [Cause Analysis] The pll fine-tuning function causes the frequency of vp1 to be incorrect [Solution] Modify clk file pll fine-tuning only affects hpll [Main modification file] kernel：drivers/clk/rockchip/clk-rk3568.c drivers/clk/rockchip/clk-pll.c	Must Update
8	Add node to enable or disable crtc, support vp0 vp1 and vp2	[Problem description] Support vp0 and vp1 and vp2 switch crtc [Cause Analysis] Not involved [Solution] Not involved [Main modification file] kernel： drivers/gpu/drm/rockchip/rockchip_drm_vop2.c	Must Update

4. 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`.

4.1 Check compile command

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

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

Default option is 'all'.
```

4.2 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
```

4.3 Compile and package each module

4.3.1 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=rk3568
#  The default can not be modified
export RK_UBOOT_FORMAT_TYPE=fit

spi nand configuration
# Select the configuration according to whether the hardware has pmic or
not:rk3568-spi-nand/rk3568-spi-nand-pmic
export RK_UBOOT_DEFCONFIG=rk3568-spi-nand
# The default can not be modified
export RK_UBOOT_FORMAT_TYPE=no-fit
# Single uboot size, final uboot.img size = single uboot * count
export RK_UBOOT_TOTAL_SIZE=1024
# The uboot number of uboot.img is count, there are 2 for backup.
export RK_UBOOT_BACKUP_COUNT=2
# Single trust size,final trust.img size = single trust * count
export RK_TRUST_TOTAL_SIZE=1024
# The trust number of trust.img is count, there are 2 for backup.
export RK_TRUST_BACKUP_COUNT=2
```

4.3.2 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=rk3568_nvr.config
```

```
# The dts of the kernel is modified according to the board type: rk3568-nvr-demo-v10-linux rk3568-evb1-ddr4-v10-linux rk3568-nvr-demo-v12-linux
export RK_KERNEL_DTS=rk3568-nvr-demo-v12-linux
# 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 nvr product configuration
export RK_KERNEL_DEFCONFIG_FRAGMENT=rk3568_nvr.config
# The dts of the kernel is modified according to the board type: rk3568-nvr-demo-v10-linux   rk3568-evb1-ddr4-v10-linux rk3568-nvr-demo-v12-linux
export RK_KERNEL_DTS=rk3568-nvr-demo-v12-linux-spi-nand
# The default can not be modified
export RK_BOOT_IMG=zboot.img
```

4.3.3 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
Rootfs compile command
# 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/rockdevrk356x-package-file-nvr-emmc.txt
export RK_PACKAGE_FILE=rk356x-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/rockdevrk356x-package-file-nvr-spi-nand.txt
export RK_PACKAGE_FILE=rk356x-package-file-nvr-spi-nand
```

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

4.3.4 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
```

4.4 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

```
Execute 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.

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`.

If make RKMPI fail, try `./build.sh cleanall` to clean env first.

5. 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/doc/`

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

6. Firmware Programming

For programming, please refer `docs/RK356x/Rockchip_RK356X_Linux_SDK_*_V*_*_CN.pdf`

7. SecureBoot Function

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

8. FAQ

Q: 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

Q: 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
```

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

A:

Check the NPU frequency: cat /sys/kernel/debug/clk/clk_scmi_npu/clk_rate

Check the GPU frequency: cat /sys/kernel/debug/clk/clk_scmi_gpu/clk_rate or
cat /sys/devices/platform/fde60000.gpu/devfreq/fde60000.gpu/cur_freq

Check the GPU load: cat /sys/devices/platform/fde60000.gpu/utilisation

Check the CPU frequency: cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq

Check the CPU available frequency voltmeters: cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequ

Enable the CPU performance mode, run the highest frequency: echo performance > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor

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

Check the VDEC frequency: cat /sys/kernel/debug/clk/clk_rkvdec_core/clk_rate

Q: 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"
```

Q: 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";
```

Q: How to modify rootfs to initramfs

A: Steps to modify:

1. Configure Initial RAM filesystem and RAM disk(initramfs/initrd) support in menuconfig.

   Config to the file or cpio path of the root file system. The path does not need to be filled in, just copy the compiled initramfs.cpio.gz to the usr directory under the kernel path.

```
make ARCH=arm64 menuconfig
General setup --->
[*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
```

2. Modify `bootargs = "earlycon=uart8250,mmio32,0xfe660000 swiotlb=1 console=ttyFIQ0 root=PARTUUID=614e0000-0000 rw rootwait"` in `kernel/arch/arm64/boot/dts/rockchip/rk3568-linux.dtsi` ;

Change `root=PARTUUID=614e0000-0000 rw rootwait` to `root=/dev/ram rw rdinit=/linuxrc`.

3. If using compression format, the default decompressed boot cannot exceed 56M. If it exceeds, the uboot decompression will fail. The reason is that the decompression address exceeds the limit and the compressed address is overwritten.

   Modify as follows, which can be expanded to 98M:

```
ubuntu@srv:/home1/rk3568_linux_sdk/u-boot$ git diff
diff --git a/include/configs/rk3568_common.h b/include/configs/rk3568_common.h
index c869b4e3ce..3c3e14ec09 100644
--- a/include/configs/rk3568_common.h
+++ b/include/configs/rk3568_common.h
@@ -77,8 +77,8 @@
        "fdt_addr_r=0x0a100000\0" \
        "kernel_addr_no_bl32_r=0x00280000\0" \
        "kernel_addr_r=0x00a80000\0" \
-       -"kernel_addr_c=0x04080000\0" \
-       -"ramdisk_addr_r=0x0a200000\0"
+       +"kernel_addr_c=0x0a280000\0" \
+       +"ramdisk_addr_r=0x04000000\0"

 #include <config_distro_bootcmd.h>
```

4. Note that initramfs will not mount devtmpfs, so there are not many device nodes under the root file system /dev.

   Need to do this manually: /bin/mount -t devtmpfs devtmpfs /dev`
   Or modify as follows:

```
diff --git a/board/rockchip/common/base/etc/inittab
b/board/rockchip/common/base/etc/inittab
index 491145cdf0..0c236a9986 100644
--- a/board/rockchip/common/base/etc/inittab
+++ b/board/rockchip/common/base/etc/inittab
@@ -15,6 +15,7 @@

 ::sysinit:/bin/mount -t proc proc /proc
+::sysinit:/bin/mount -t devtmpfs dev /dev
 ::sysinit:/bin/mount -o remount,rw /
 ::sysinit:/bin/mkdir -p /dev/pts
 ::sysinit:/bin/mkdir -p /dev/shm
```

Q: Instructions for vnvm partitons.

A: The vnvm partition is reserved for saving ETH MAC address, machine serial number and other information.

If there is no vnvm 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` .

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)
```

Q: How to grab a flame graph for analysis.

A: How to grab a flame graph:

1. 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
```

Q: RK3568 slave supports RAMBOOT

A: Follow these steps:

1. Modify the uboot configuration to rk3568-ramboot.config, and recompile uboot, then get u-boot/uboot.img, u-boot/rk356x_ramboot_loader_v1.09.108.bin and u-boot/trust.img.

```
if use bl32 in rkbin/RKTRUST/RK3568TRUST.ini,should open config
CONFIG_OPTEE_CLIENT=y in rk3568-ramboot.config
./make.sh rk3568-ramboot --sz-uboot 2048 1 --sz-trust 1024 1

or if not use bl32 in rkbin/RKTRUST/RK3568TRUST.ini, use default rk3568-
ramboot.config
./make.sh rk3568-ramboot --sz-uboot 2048 1 --sz-trust 512 1
also should change [BL32_OPTION] SEC=1 to SEC=0  in
rkbin/RKTRUST/RK3568TRUST.ini, otherwise will occur fails when building
```

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:

```
./upgrade_tool/upgrade_tool_ramboot  rd 3     //If the device is already in
maskrom mode, you don't need to run this command.
./upgrade_tool/upgrade_tool_ramboot  ul  rk356x_ramboot_loader_v1.09.108.bin
./upgrade_tool/upgrade_tool_ramboot  wl 0x2000 uboot.img
./upgrade_tool/upgrade_tool_ramboot  wl 0x42000 trust.img
//./upgrade_tool/upgrade_tool_ramboot  wl 0x80000 boot.img  //If the boot.img is
not delivered from the host, you don't need to run this command.
./upgrade_tool/upgrade_tool_ramboot  run 0x2000 0x42000 0x80000 uboot.img
trust.img boot.img
```

Q: 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.

Q: 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.

Q: 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

Q: 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/RK3568MINIALL.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= 1333.
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/rk3568-linux.dtsi:18: rockchip,baudrate = <1500000>; /* Only 115200 and 1500000 */

Q: How to modify the correspondence between gmac and eth under linux

A: On RK3568, gmac0 corresponds to eth1 by default, and gmac1 corresponds to eth0. Because the SDK code needs to be upstream by default, the enumeration order in dtsi is required to be sorted according to the register address.

gmac1: ethernet@fe01000 to eth0

gmac0: ethernet@fe2a0000 to eth1

If you need to change the order, you need to modify the definition order of gmac0 and gmac1 in rk3568.dtsi, that is, put gmac0 in front of gmac1.

Q: 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 dev[:part]

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

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

Q: How to check HDMI output parameters

A: cat /sys/kernel/debug/dw-hdmi/status

Q: Why does a confirmation box pop up when compiling the kernel?

A: Because the current SDK needs to confirm whether the io power domain configured in the dts is consistent with the hardware board, a confirmation box will be displayed before the kernel is successfully compiled for the first time.

In addition, in order to prevent customers from directly including the reference dtsi file to the actual project, causing the actual project hardware voltage and software dts configuration voltage domain to not match, the default voltage domain of all reference dtsi in the SDK is set to 3.3V. After downloading the SDK, if you need to compile RK EVB firmware, VCCIO4 and VCCIO6 need to be configured as 1V8, and the rest should be configured as 3V3. Before changing, you need to confirm with the hardware engineer whether it is consistent with the actual voltage of the hardware.

For the customer's board, please confirm with the hardware engineer. It needs to be based on the actual hardware configuration. The voltage domain configuration is very important. If it does not match, it will have serious consequences.

RK NVR SDK board (V10&V12) need to modify as follows:

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi
index 7d72e714a61d..58d2e531ca6a 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi
@@ -332,9 +332,9 @@
        pmuio2-supply = <&vcc_3v3>;
        vccio1-supply = <&vcc_3v3>;
        vccio3-supply = <&vcc_3v3>;
-       vccio4-supply = <&vcc_3v3>;
+       vccio4-supply = <&vcc_1v8>;
        vccio5-supply = <&vcc_3v3>;
-       vccio6-supply = <&vcc_3v3>;
+       vccio6-supply = <&vcc_1v8>;
        vccio7-supply = <&vcc_3v3>;
         };
```

EVB SDK board need to modify as follows:

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
index 1bfaeb681069..21680d999834 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
@@ -1544,9 +1544,9 @@
        pmuio2-supply = <&vcc3v3_pmu>;
        vccio1-supply = <&vccio_acodec>;
        vccio3-supply = <&vccio_sd>;
-       vccio4-supply = <&vcc_3v3>;
+       vccio4-supply = <&vcc_1v8>;
        vccio5-supply = <&vcc_3v3>;
-       vccio6-supply = <&vcc_3v3>;
+       vccio6-supply = <&vcc_1v8>;
        vccio7-supply = <&vcc_3v3>;
```

```
};
```

Q: 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.

Q: VGA constant output

A: For the NVR SDK, HDMI and VGA are forced to output. At the same time, the application can still detect the HDMI hot-plug event and obtain the HDMI status.

1. In the uboot/kernel stage, HDMI & VGA are forced to output by default, and the default output resolution is 1024x768.

   The detailed configuration is in the &route_hdmi and &route_edp nodes of arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi.

2. The application can obtain the status of the HDMI interface through the drm interface. The application layer can be able to register callbacks through the interface RK_S32 RK_MPI_VO_RegCallbackFunc(RK_U32 enIntfType, RK_U32 u32Id, RK_VO_CALLBACK_FUNC_S *pstCallbackFunc). When the interface state changes, the callback function is notified.

3. After application is running, the corresponding device will always output after enabling VoDev of HDMI&VGA. It is recommended to use RK356X_VO_DEV_HD0 for HDMI and RK356X_VO_DEV_HD1 for VGA.

Q: Does RK3568 support the same content displayed on dual-screen and the different content displayed on dual-screen?

A: RK3568 supports the above 2 display modes.

1. The same content display mode: HDMI & VGA only support one output resolution at the same time.
2. The different content display mode: suggest that HDMI supports up to 4KP30 and VGA supports up to 1080P30.
3. Same content display/different content display switching is achieved by operating VoDev at the application layer

   The same content display mode: enable edp_in_vp0 in dts; the application layer enables RK356X_VO_DEV_HD0, VoPubAttr.enIntfType = VO_INTF_HDMI | VO_INTF_EDP;

   The different content display mode: the application layer enables 2 VoDev.

Q: 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/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_Nand_Flash_Open_Source_Solution_CN.pdf`

2. Ubi block supports squashfs

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

3. Ubifs space optimization

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

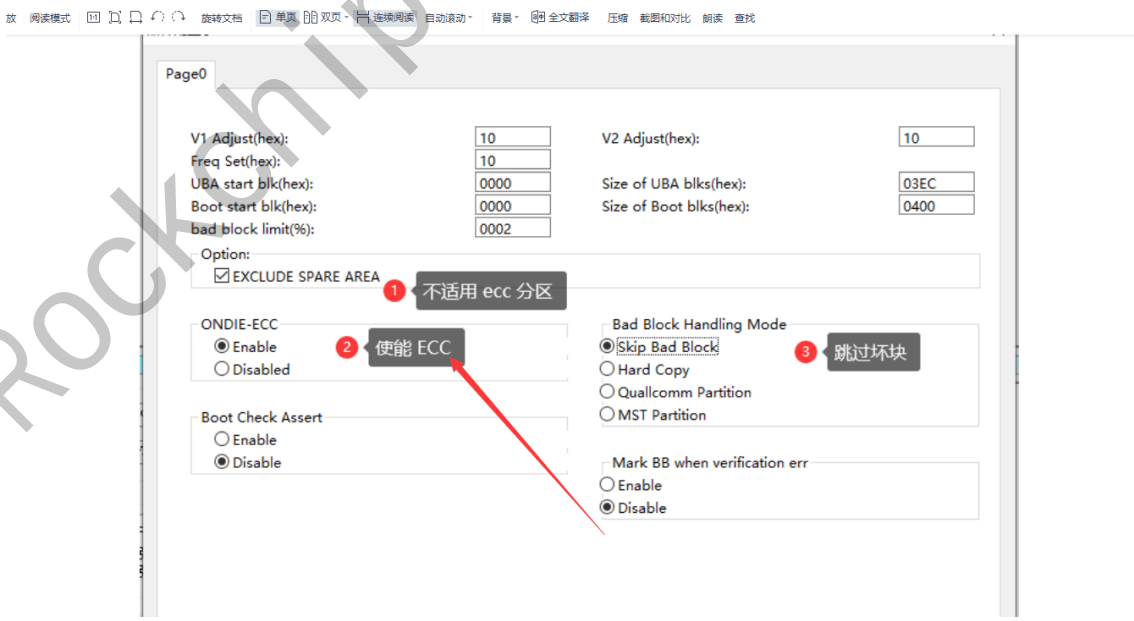
Q: Whether spi nand supports the programmer to program firmware

A: Spi nand supports programmer to program firmware.

1. Refer to the programmer programming chapter in the document `docs/Linux/ApplicationNote/Rockchip_Developer_Guide_Linux_Nand_Flash_Open_Source_Solution_CN.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.



Q: 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/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.

Q: 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.

Q: 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
```

Q: How to reduce the size of uboot/boot

A: The NVR SDK optimizes the size of uboot and boot by default. The size of uboot is 1M, the size of boot is about 5M.

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.

2. Boot:

   Crop out unwanted peripheral drivers, such as: wifi, 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.

Q: 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/fe8a0000.usb2-phy/otg_mode`

Switch to device: `echo peripheral > /sys/devices/platform/fe8a0000.usb2-phy/otg_mode`

Q: 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  or mount -t nfs -o
nolock,nfsvers=3,vers=3 10.12.201.5:/nfs  /mnt/nfs

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

Q: DDR maintains a fixed frequency and turns off DDR frequency conversion.

A: The NVR SDK defaults to configure DDR to maintain a fixed frequency.

1. Set the frequency in command:

```
# Check available frequencies
cat /sys/class/devfreq/dmc/available_frequencies
# Set governors to userspace
echo userspace > /sys/class/devfreq/dmc/governor
# Set frequency
echo 1560000000 > /sys/class/devfreq/dmc/userspace/set_freq
```

2. Disable frequency conversion: disable dmc in the dts, refer the follows:

```
--- a/arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-evb.dtsi
@@ -402,7 +402,7 @@
 &dmc {
        center-supply = <&vdd_logic>;
-       status = "okay";
+       status = "disabled";
         };
```

Q: 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

Q: 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.

Q: I2c_master_send failed to send large data (eg 24KB)

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

Q: Configure SATA speed

A: Modify kernel driver:

```
  pipe_con0_for_sata     = { 0x0000, 15, 0, 0x00, 0x0000 }, 1.5G
  pipe_con0_for_sata     = { 0x0000, 15, 0, 0x00, 0x1110 }, 3G
  pipe_con0_for_sata     = { 0x0000, 15, 0, 0x00, 0x2220 }, 6G
```

```
diff --git a/drivers/phy/rockchip/phy-rockchip-naneng-combphy.c
b/drivers/phy/rockchip/phy-rockchip-naneng-combphy.c
  index 08445c1890eb..3df0e0e05ab4
  --- a/drivers/phy/rockchip/phy-rockchip-naneng-combphy.c
  +++ b/drivers/phy/rockchip/phy-rockchip-naneng-combphy.c
  @@ -604,7 +604,7 @@ static const struct rockchip_combphy_grfcfg
rk3568_combphy_grfcfgs = {
          .con2_for_sata          = { 0x0008, 15, 0, 0x00, 0x80c3 },
          .con3_for_sata          = { 0x000c, 15, 0, 0x00, 0x4407 },
          /* pipe-grf */
  -       .pipe_con0_for_sata     = { 0x0000, 15, 0, 0x00, 0x2220 },
  +       .pipe_con0_for_sata     = { 0x0000, 15, 0, 0x00, 0x0000 },
          .pipe_sgmii_mac_sel     = { 0x0040, 1, 1, 0x00, 0x01 },
          .pipe_xpcs_phy_ready    = { 0x0040, 2, 2, 0x00, 0x01 },
          .u3otg0_port_en         = { 0x0104, 15, 0, 0x0181, 0x1100 },
```

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

A:

1. RK3568 NVR SDK should update to V1.5 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.

- In pass-through mode, when sending frame is 30fps and display is 60fps(that is, two vsync sending one frame of data), there is frequent out-of-sync, you can add 3ms before RK_MPI_VO_SendFrame() and then do the sync test.
- 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 4 before enable chn. Use RK_MPI_VO_SetLayerDispBufLen() to configure layer buffer number to 5.

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

  When vo sending frame has been synchronized, there is frequent out-of-sync, you can add 3ms before RK_MPI_VO_SendFrame() and then do a sync test.

  Note: Do not apply for the release of the buffer sent to vo frequently. It is recommended to directly use the buffer decoded by vdec or use the bufferpool to recycle the buffer.

Q: Synchronization between different RK3568 CPUs

A: RK3568 NVR SDK should update to V1.5 or higher.

1. 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 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;
```

2. Because the clock sources of different RK3568 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.

   - RK3568 uses 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;
     int ret = 0, i = 0;

     clk = __clk_lookup("hpll");
     clk1 = __clk_lookup("pll_hpll");
     if (clk == NULL)
         printk("---get hpll clk fail---\n");
     if(clk1 == NULL)
         printk("---get pll_hpll clk fail---\n");

     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);
     ```

```
}
```

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.

  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.
- It 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.

3. 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 pass-through mode, when sending frame is 30fps and display is 60fps(that is, two vsync sending one frame of data), there is frequent out-of-sync, you can add 3ms before RK_MPI_VO_SendFrame() and then do the sync test.
- 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 4 before enable chn.

  Use RK_MPI_VO_SetLayerDispBufLen() to configure layer buffer number to 5.

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

  When vo sending frame has been synchronized, there is frequent out-of-sync, you can add 3ms before RK_MPI_VO_SendFrame() and then do a sync test.

  Note: Do not apply for the release of the buffer sent to vo frequently. It is recommended to directly use the buffer decoded by vdec or use the bufferpool to recycle the buffer.

Q: VI-VENC-VDEC-VO Path Delay Debug

A: RK3568 NVR SDK should update to V1.5 or higher.

1. 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.

Q: Enter the maskrom mode

A: 1) If there is a maskrom button, first press and hold the reset button, then press and hold the maskrom button, release the reset (you will see the maskrom displayed in the entry tool), and then release the maskrom (the order is very important to avoid abnormalities);

2).If there is not a maskrom button, use flash data cable io0 to short ground instead of maskrom button. If there is not uboot or uboot runs away, cannot enter the loader mode, can only enter the maskrom mode.

Q: The kernel uses the LZMA compression format by default. If the kernel is packaged in the initram method, uboot will fail at bootm.

A: Need to modify to LZ4:

```
diff --git a/arch/arm64/Makefile b/arch/arm64/Makefile
index a3a8e47..5b39088 100644
--- a/arch/arm64/Makefile
+++ b/arch/arm64/Makefile
@@ -189,7 +189,7 @@ define archhelp
   echo  '                  install to $$(INSTALL_PATH) and run lilo'
 endef

-kernel.img: Image.lzma
+kernel.img: Image.lz4
        $(Q)scripts/mkkrnlimg $(objtree)/arch/arm64/boot/Image
$(objtree)/kernel.img >/dev/null
        @echo '  Image:  kernel.img is ready'
 ifdef CONFIG_MODULES
diff --git a/scripts/mkimg b/scripts/mkimg
index eac3447..2944a6d 100755
--- a/scripts/mkimg
+++ b/scripts/mkimg
@@ -60,8 +60,8 @@ if [ "${ARCH}" == "arm" ]; then
        ZIMAGE=zImage
 else
        DTB_PATH=${objtree}/arch/arm64/boot/dts/rockchip/${DTB}
-       KERNEL_ZIMAGE_ARG="--kernel ${objtree}/arch/arm64/boot/Image.lzma"
-       ZIMAGE=Image.lzma
+       KERNEL_ZIMAGE_ARG="--kernel ${objtree}/arch/arm64/boot/Image.lz4"
+       ZIMAGE=Image.lz4
         fi
         if [ ! -f ${DTB_PATH} ]; then
        echo "No dtb" >&2
```