Rockchip Developer Guide Dual Storage

文件标识：RK-KF-YF-461

发布版本：V1.4.0

日期：2023-10-20

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文介绍 Rockchip SDK 对于双存储方案的支持原理及相应配置。

产品版本

芯片名称	内核版本
具备多个 IO 独立的存储的芯片	kernel 4.19 及以上

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	林鼎强	2022-07-12	初始版本
V1.1.0	赵仪峰	2022-07-22	增加vendor stroage支持
V1.2.0	林鼎强	2023-06-25	增加 RK PCIe EP 双存储方案 支持，增加固件打包及 OTA 升级说明
V1.3.0	林鼎强	2023-08-08	增加 Nor Flash 固件分区布局裁剪建议
V1.4.0	林鼎强	2023-10-20	增加 pcie_idb.img 镜像制作说明

目录

1. 简介

多存储支持

RK SOC 通常支持多个 IO 独立的存储控制器，如 RK3568 同时支持 Nor flash 与 eMMC，RK3588 支持 Nor flash 与 PCIe SSD，对应存储控制器的 IO 相互独立。 而根据是否支持被掩膜代码探测到可将存储器件分为 Bootable 和 Non-bootable 类型，例如：

- bootable 器件：eMMC、SPI Flash、PP Nand
- Non-bootable 器件：NVMe SSD、SATA（仅 Non-bootable 存储方案掩膜代码无法正常引导启动）

应用方案

为了支持 Non-bootable 主存方案，所以需要搭配一个 bootable 存储器件，例如：

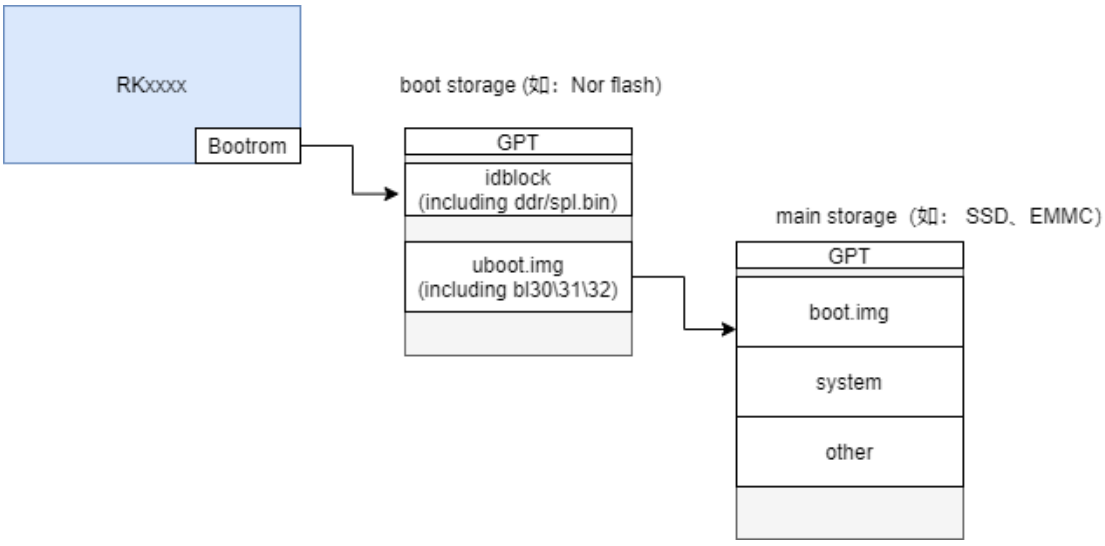
- Nor + NVMe SSD\SATA 实现大容量存储支持

除此之外，还有以下应用：

- Nor + eMMC 实现特殊固件保护机制，例如可恢复小系统存储在 Nor 中避免 eMMC 出现不可挽救异常。

2. 原理

2.1 RK 通用双存储方案启动



RK 双存储方案流程及固件信息：

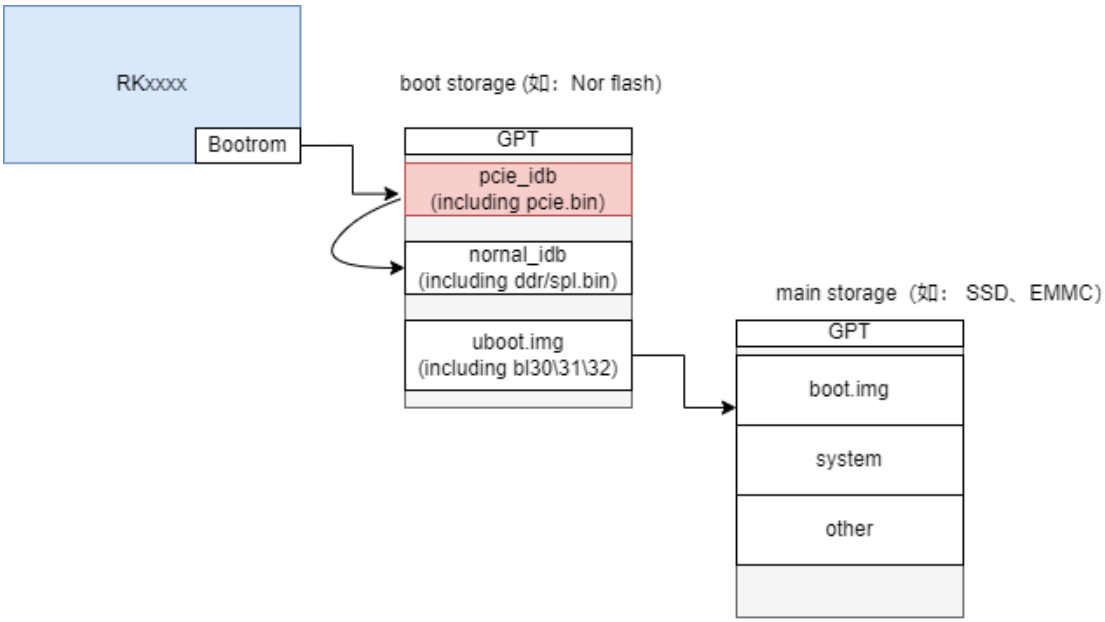
启动阶段	固件存储位置	该阶段所初始化的主存储	所需配置
Bootrom	芯片掩膜 ROM	bootable storage，例如 Nor	\
SPL	bootable storage，例如 Nor	bootable storage，例如 Nor	\
u-boot	bootable storage，例如 Nor	main storage，例如 NVMe	u-boot 特殊配置
kernel	main storage，例如 NVMe	main storage，例如 NVMe	内核标准配置

说明：

- u-boot 阶段默认初始化前级所用的存储类型，所以此时要转换为初始化主存储，如 NVMe，需要添加 u-boot 特殊配置，详细参考 “配置” 章节

2.2 RK PCIe EP 双存储方案启动

RK 部分 SOC 支持 PCIe EP 功能，支持设计为 PCIe EP 标准卡对接通用 PC，因此 PCIe 控制器有 early init 的需求，故增加 pcie_idb.img 镜像实现该功能，流程如下：



RK 双存储方案流程及固件信息：

启动阶段	固件存储位置	该阶段所初始化的主存储	所需配置
Bootrom	芯片掩膜 ROM	bootable storage，例如 Nor	\
PCIe.bin	bootable storage，例如 Nor	NULL	\
SPL	bootable storage，例如 Nor	bootable storage，例如 Nor	\
u-boot	bootable storage，例如 Nor	main storage，例如 NVMe	u-boot 特殊配置
kernel	main storage，例如 NVMe	main storage，例如 NVMe	内核标准配置

说明：

- u-boot 阶段默认初始化前级所用的存储类型，所以此时要转换为初始化主存储，如 NVMe，需要添加 u-boot 特殊配置，详细参考 “配置” 章节

3. 配置

如 “原理” 章节所述，RK 双存储方案 SDK 在 u-boot 阶段修改为初始化目标存储，因此 u-boot 配置需关注以下主存储驱动配置、主存储转换配置和使用 Embedded Kernel DTB 配置。

3.1 u-boot 配置

3.1.1 主存储驱动配置

参考 《Rockchip_Developer_Guide_UBoot_Nextdev_CN.pdf》 文档 “CH05-驱动模块” 章节：

- emmc 参考 "Storage" 章节，通常默认兼容
- PCIe NVMe 参考 “PCIe” 章节

3.1.2 主存储转换配置

defconfig 添加配置;

```
CONFIG_ROCKCHIP_BOOTDEV="nvme 0"    # 可择 "nvme"-NVMe "mmc"-eMMC "scsi"-SATA
CONFIG_ROCKCHIP_EMMC_IOMUX=y        # Nor + eMMC 方案 spl.bin 中完成 eMMC iomux
```

3.1.3 NVMe/SATA 使用 Embedded Kernel DTB 配置

RK u-boot 分为两个阶段：

- u-boot 初始化早期，使用 u-boot 标准 dtb，dtb 信息量少，相关配置参考 “存储驱动配置” 章节
- u-boot 后期阶段，可选：
  - using kernel dtb：使用后级镜像中的 kernel dtb 文件，启动过程中解析并初始化对应设备驱动
  - using embedded dtb：打包 u-boot 特定目录下的 dtb 文件到 u-boot 镜像内，启动过程中解析并初始化对应设备驱动 —— 使用该配置

using embedded dtb

defconfig 配置：

```
CONFIG_EMBED_KERNEL_DTB_ALWAYS=y
CONFIG_EMBED_KERNEL_DTB_PATH="dts/rk3588-evb1-lp4-v10.dtb"     # dtb 存放目录
CONFIG_SPL_FIT_IMAGE_KB=2560                                   # 通常使用 embedded
dtb 后 u-boot 固件会变大
```

添加 dtb 文件：

- 编译内核所需的 kernel dtb，存放在 u-boot/dts/ 目录下，建议沿用 kernel dtb 命名以作区分
- 后续如有更新需要，更新替换 dtb 即可

3.2 Kernel 配置

使用 Kernel 标准配置。

3.3 Android 配置

Android 中修改存储介质为对应的 pcie，用于分区挂载。

```
@sys2_206:~/4_Android12_29_sdk/device/rockchip/rk3588$ git diff
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -25,7 +25,7 @@ PRODUCT_KERNEL_CONFIG ?= rockchip_defconfig pcie_wifi.config

 #BOARD_AVB_ENABLE := true
 # used for fstab_generator, sdmmc controller address
-PRODUCT_BOOT_DEVICE := fe2e0000.mmc
+PRODUCT_BOOT_DEVICE := fe180000.pcie
 PRODUCT_SDMMC_DEVICE := fe2c0000.mmc

 SF_PRIMARY_DISPLAY_ORIENTATION := 0
```

4. 固件分区布局及打包

4.1 RK 通用双存储方案固件

默认 SDK 仅支持单存储固件打包，客户可参考以下脚本修改 SDK 打包脚本，或直接使用新打包方案实现双存储固件打包。

工具目录

```
➜  dual_storage tree
.
├── afptool
├── gen-package-file.sh
├── Image                               # main storage 固件
│   ├── boot.img
│   ├── MiniLoaderAll.bin
│   ├── parameter.txt
│   └── userdata.img
├── Image-nor                           # boot storage 固件
│   ├── MiniLoaderAll.bin
│   ├── parameter.txt
│   └── uboot.img
├── mkupdate.sh
└── rkImageMaker
```

新增 Image-nor 目录

编译生成的 Nor flash 固件置于 Image-nor 目录中，其中 parameter.txt 布局可参考如下配置：

```
CMDLINE:mtdparts=rk29xxnand:0x00000080@0x00000000(gpt),0x00000800@0x00000080(idb)
,0x00000800@0x00001000(vnvm),0x00002000@0x00001800(uboot)
```

修改 mkupdate.sh 脚本以支持双存储打包

```
#!/bin/bash
# Author: kenjc@rock-chips.com
# 2021-08-13
# Use: ./mkupdate.sh PLATFORM IMAGE_PATH to pack update.img

declare -A vendor_id_map
vendor_id_map["rk3588"]="-RK3588"
vendor_id_map["rk356x"]="-RK3568"
vendor_id_map["rk3326"]="-RK3326"
vendor_id_map["px30"]="-RKPX30"
vendor_id_map["rk3368"]="-RK330A"
vendor_id_map["rk322x"]="-RK322A"
vendor_id_map["rk3399pro"]="-RK330C"
vendor_id_map["rk3328"]="-RK322H"
vendor_id_map["rk3288"]="-RK32"
vendor_id_map["rk3126c"]="-RK312A"
vendor_id_map["rk3399"]="-RK330C"

readonly PLATFORM=$1
readonly IMAGE_PATH=$2
readonly IMAGE_PATH_NOR=$3
readonly PACKAGE_FILE=package-file-tmp

echo "packing update.img with $IMAGE_PATH ${vendor_id_map[$PLATFORM]}"

pause() {
  echo "Press any key to quit:"
  read -n1 -s key
  exit 1
}


if [ ! -f "$IMAGE_PATH/parameter.txt" ]; then
    echo "Error:No found parameter!"
#   pause
fi

echo "regenernate $PACKAGE_FILE..."
if [ -f "$PACKAGE_FILE" ]; then
    rm -rf $PACKAGE_FILE
fi
./gen-package-file.sh $IMAGE_PATH > $PACKAGE_FILE

echo "start to make emmc update.img..."
./afptool -pack ./ $IMAGE_PATH/update.img $PACKAGE_FILE || pause
./rkImageMaker ${vendor_id_map[$PLATFORM]} $IMAGE_PATH/MiniLoaderAll.bin
$IMAGE_PATH/update.img emmc_update.img -os_type:androidos -storage:emmc || pause
echo "Making emmc_update.img OK."

echo "regenernate $PACKAGE_FILE..."
if [ -f "$PACKAGE_FILE" ]; then
```

```
    rm -rf $PACKAGE_FILE
fi

./gen-package-file.sh $IMAGE_PATH_NOR > $PACKAGE_FILE

echo "start to make spinor update.img..."

./afptool -pack ./ $IMAGE_PATH_NOR/update.img $PACKAGE_FILE || pause
./rkImageMaker ${vendor_id_map[$PLATFORM]} $IMAGE_PATH_NOR/MiniLoaderAll.bin
$IMAGE_PATH_NOR/update.img spinor_update.img -os_type:androidos -storage:spinor
|| pause
echo "Making spi_update.img OK."

./rkImageMaker -merge ./update.img ./emmc_update.img ./spinor_update.img
rm ./spinor_update.img ./emmc_update.img

exit 0
```

执行脚本

RK3568:

```
./mkupdate.sh rk356x Image/ Image-nor/
```

RK3588:

```
./mkupdate.sh rk3588 Image/ Image-nor/
```

4.2 RK PCIe EP 双存储方案固件打包

打包方案原理同 RK 通用双存储方案。

工具目录

```
➜  dual_storage tree
.
├── afptool
├── gen-package-file.sh
├── Image                          # main storage 固件
│   ├── boot.img
│   ├── MiniLoaderAll.bin
│   ├── parameter.txt
│   └── userdata.img
├── Image-nor                      # boot storage 固件
│   ├── MiniLoaderAll.bin
│   ├── normal_idb.img             # idblock.img, 包含 ddr.bin 和 spl.bin
│   ├── parameter.txt
│   ├── pcie_idb.img               # PCIe_idblock.img, 源码工程 bootloader-ep 默认输出
双备份镜像, 包含 pcie.bin
│   └── uboot.img
├── mkupdate.sh
└── rkImageMaker
```

说明:

- pcie_idb.img 请参考以下命令生成，进入 rkbin 目录：

```
#生成 loader
./tools/boot_merger RKBOOT/RK3588MINIALL_PCIE_EP.ini
➜  rkbin git:(master) ✗ ls rk3588_pcie_loader_v1.00.bin
rk3588_pcie_loader_v1.00.bin

#制作 idb，要求使用 spinand type 做对齐
./tools/programmer_image_tool -i rk3588_pcie_loader_v1.00.bin -b 128 -p 2 -t
spinand -o ./

#制作双备份的 pcie_idb.img
cat idblock.img > pcie_idb.img
cat idblock.img >> pcie_idb.img
rm idblock.img
```

- nornal_idb.img 请参考以下命令生成，进入 rkbin 目录：

```
#生成 loader
./tools/boot_merger RKBOOT/RK3588MINIALL.ini
➜  rkbin git:(master) ✗ ls rk3588_spl_loader_v1.14.113.bin
rk3588_spl_loader_v1.14.113.bin

#制作 idb，要求使用 spinand type 做对齐
./tools/programmer_image_tool -i rk3588_spl_loader_v1.14.113.bin -b 128 -p 2 -t
spinand -o ./

#制作双备份的 normal_idb.img
cat idblock.img > normal_idb.img
cat idblock.img >> normal_idb.img
rm idblock.img
```

新增 Image-nor 目录

```
CMDLINE:mtdparts=rk29xxnand:0x00000080@0x00000000(gpt),0x00000400@0x00000080(pcie
_idb),0x00000b80@0x00000480(normal_idb),0x00000800@0x00001000(vnvm),0x00002000@0x
00001800(uboot)
```

修改 mkupdate.sh 脚本以支持双存储打包

同 “RK 通用双存储方案” 相应章节。

执行脚本

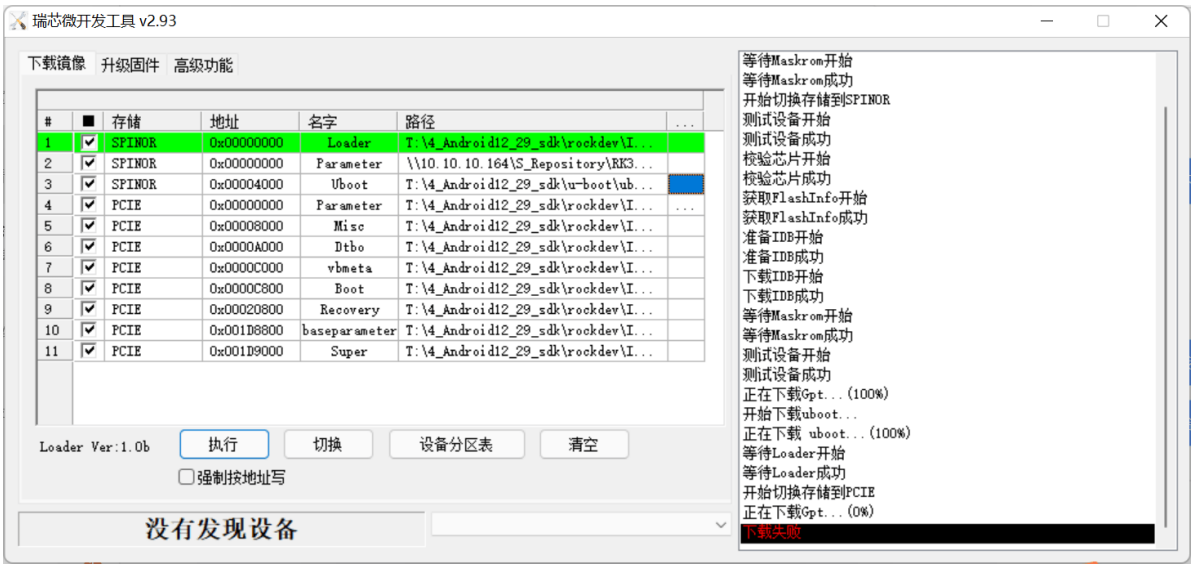
同 “RK 通用双存储方案” 相应章节。

5. 固件烧录

RK 烧录工具支持多存储烧录的方式。

windows：RKDevTool_Release_v2.93 及以上版本工具，详细参考阅读工具包下的《开发工具用户手册》中的 “1.12 多设备选择 ” 章节。

一次性烧录



仅烧录 SPI Nor

设备进入 maskrom mode，仅勾选存储为 SPINOR 固件

仅烧录 NVMe

设备进入 loader mode，仅勾选存储为 PCIe 固件

6. OTA 升级

6.1 RK 通用双存储方案 OTA 升级

Nor+ eMMC 为例

- Nor配置：

  Nor 部分包含 MiniLoader 、parameter.txt 、 uboot.img 这三个分区都可以支持OTA升级，这里以 uboot 分区举例。**不推荐对 MiniLoader 和paramter.txt 分区进行升级，风险较高**。

  uboot 分区是在 Nor 的存储设备中，boot 和 rootfs 是在eMMC存储设备中，所以需要在内核中添加 uboot 的分区信息以及 Nor 的设备配置。

  添加分区信息：

  基于 “RK 通用双存储方案固件打包” 章节中的 parameter.txt 布局定义recovery dts mtdparts 分区表：

  ```
  mtdparts=sfc_nor:0x00100000@0x00010000(idb),0x00100000@0x00200000(vnvm),0x004
  00000@0x00300000(uboot)
  ```

  添加Nor的设备配置：

```
&sfc {
        status = "okay";
        pinctrl-names = "default";
        pinctrl-0 = <&fspim1_pins>;                    // 选择实际 io
        flash@0 {
                compatible = "jedec,spi-nor";
                reg = <0>;
                spi-max-frequency = <75000000>;
                spi-rx-bus-width = <4>;
                spi-tx-bus-width = <1>;
        };
};
```

添加成功后即可在根文件系统查看是否存在 /dev/mtdblock 设备。若设备不存在请检查配置信息以及查看开机 log，是否有 spi-nor 相关报错，可能是 iomux 配置错误或者 Nor 的型号不支持等问题。

- eMMC 配置：

  eMMC 的分区信息都在 eMMC 设备的 parameter.txt 中，只要在分区表添加对应分区，根文件系统就会生成对应的 mmcblock 设备。

Nor 和 eMMC 设备对应的分区节点都生成后，使用 RK OTA 方案可以通过如下链接下载 recovery 补丁，参考补丁说明进行 OTA 升级。自定义升级方案可自由选择方式对分区节点进行升级：

```
链接：https://pan.baidu.com/s/1LiutCDJZ8BXiEN6X8U6teQ?pwd=nwm3
提取码：nwm3
```

6.2 RK PCIe EP 双存储方案 OTA 升级

原理同RK 通用双存储方案 OTA 升级，仅修改添加分区信息。

基于 “RK PCIe EP 双存储方案固件打包” 章节中的 parameter.txt 布局定义recovery dts mtdparts 分区表：

```
mtdparts=sfc_nor:0x00080000@0x00010000(pcie_idb),0x00170000@0x00090000(normal_idb
),0x00100000@0x00200000(vnvm),0x00400000@0x00300000(uboot)
```

7. SPI NOR VENDOR STORAGE支持

使用SPI NOR + SSD（SATA或NVME）双存储时，SN、MAC等数据是需要写到SPI NOR中。

SPI NOR需要定义一个"vnvm"分区来存放VENDOR STORAGE的数据，分区起始位置和大小都需要是64KB的整数倍。

7.1 分区表修改

在SPI NOR的分区表文件“parameter.txt”中增加一个"vnvm"分区，单位是sector（512 bytes）。
参考：

```
CMDLINE:
mtdparts=rk29xxnand:0x00000200@0x00000c00(vnvm),0x00003000@0x00000c00(uboot)
```

7.2 Kernel增加SPI NOR分区表

Kernel需要在dts的CMDLINE里面增加SPI NOR的分区表，和“parameter.txt”里面定义的各个分区的起始位置和大小都相同，单位是byte。
参考：

```
CMDLINE:
mtdparts=sfc_nor:0x00040000@0x00180000(vnvm),0x00600000@0x00200000(uboot)
```

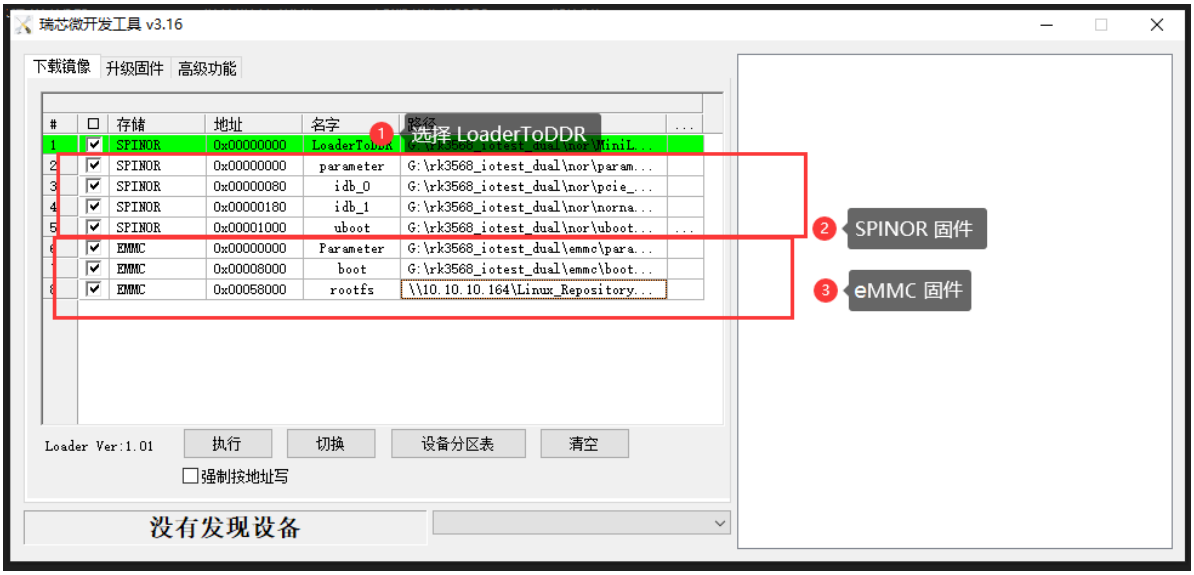
8. 常见问题

8.1 Nor Flash 固件分区布局裁剪建议

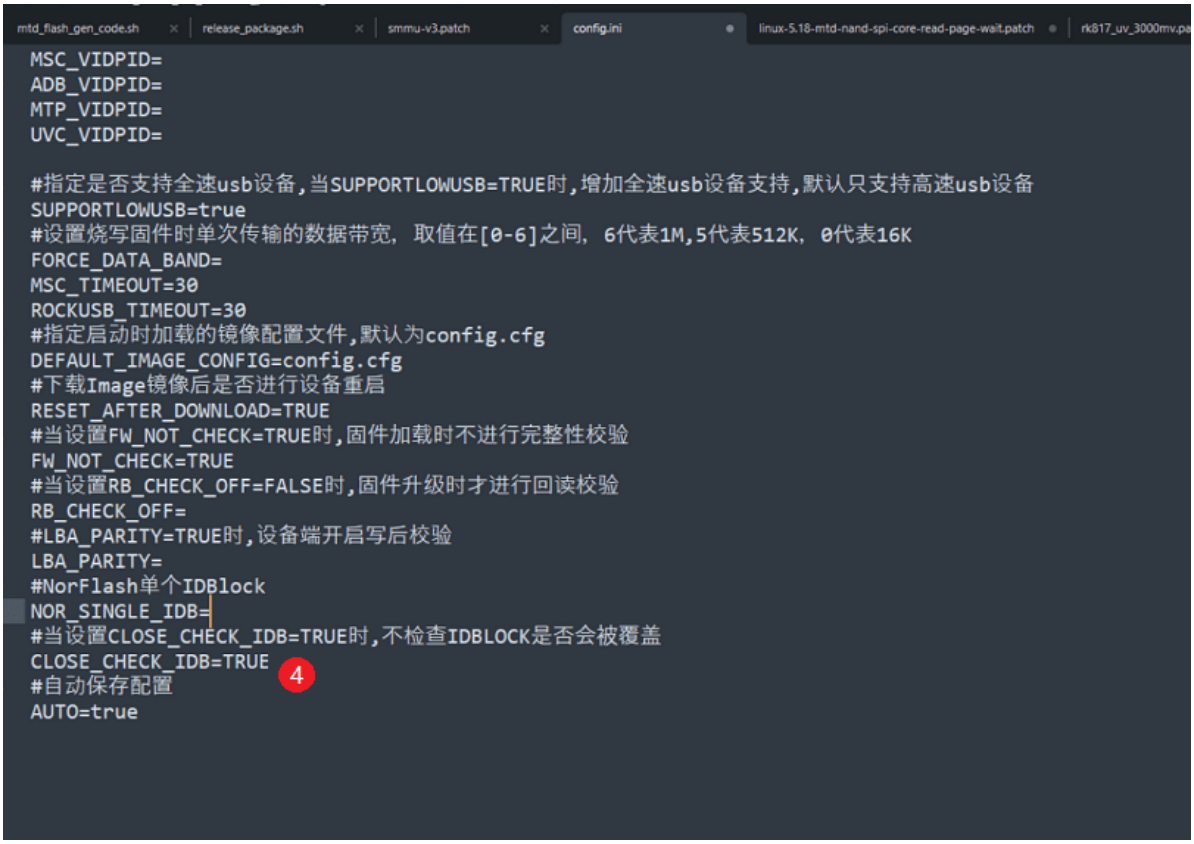
默认方案双存储中 Nor flash 升级 Loader 固件时会自动布局 idb 镜像，包括升级起始地址和多备份，建议保持默认方案。

如需调整 Nor flash 固件分区布局，以下以 RK3568 Nor + eMMC 方案提供建议。

AndroidTool 工具固件分区布局：



AndroidTool 工具配置文件修改：



说明:

1. 改 Loader 选项为 LoaderToDDR：取消 Loader 选项带来的 idb 镜像自动布局，该 Loader 固件仅做引导升级工具
2. SPI Nor 固件：其中 parameter 既能以 parameter 项 - parameter.txt 文件的方式由工具自动分区表升级，又能以 gpt 项 - gpt.img 镜像方式升级
3. eMMC 固件：整体同单存储 eMMC 方案，一样要有 parameter.txt 文件生成对应分区表
4. 由于取消 idb 镜像自动布局，升级工具需关闭 idb 镜像检测

9. 附录

9.1 RK3588s EVB1 Nor + NVMe 参考补丁

```
From 89825ff12bc92ba9dd9e063e9835ecb03b732b92 Mon Sep 17 00:00:00 2001
From: Jon Lin <jon.lin@rock-chips.com>
Date: Fri, 19 Nov 2021 20:57:52 +0800
Subject: [PATCH] TEST: uboot: rk3588s-tablet&evb1: nvme

Change-Id: I332fcceb8984712a4cd3eec053f813590f4e9bbe
Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
---
 arch/arm/dts/rk3588-u-boot.dtsi | 34 ++++++++++++++++++++++++++++++++++
 configs/rk3588_defconfig        | 12 ++++++++++++
 2 files changed, 46 insertions(+)

diff --git a/arch/arm/dts/rk3588-u-boot.dtsi b/arch/arm/dts/rk3588-u-boot.dtsi
index 3fe8054aac..25ebf26873 100644
--- a/arch/arm/dts/rk3588-u-boot.dtsi
```

```
+++ b/arch/arm/dts/rk3588-u-boot.dtsi
@@ -22,6 +22,33 @@
         compatible = "rockchip,rk3588-secure-otp";
         reg = <0x0 0xfe3a0000 0x0 0x4000>;
     };
+
+   vcc12v_dcin: vcc12v-dcin {
+       u-boot,dm-pre-reloc;
+       compatible = "regulator-fixed";
+       regulator-name = "vcc12v_dcin";
+       regulator-always-on;
+       regulator-boot-on;
+       regulator-min-microvolt = <12000000>;
+       regulator-max-microvolt = <12000000>;
+   };
+
+   vcc3v3_pcie20: vcc3v3-pcie20 {
+       u-boot,dm-pre-reloc;
+       compatible = "regulator-fixed";
+       regulator-name = "vcc3v3_pcie20";
+       regulator-min-microvolt = <3300000>;
+       regulator-max-microvolt = <3300000>;
+       enable-active-high;
+       gpio = <&gpio4 RK_PB1 GPIO_ACTIVE_HIGH>;
+       startup-delay-us = <5000>;
+       vin-supply = <&vcc12v_dcin>;
+   };
+};
+
+&combphy2_psu {
+   u-boot,dm-pre-reloc;
+   status = "okay";
 };

 &firmware {
@@ -201,6 +228,13 @@
     u-boot,dm-spl;
 };

+&pcie2x1l1 {
+   u-boot,dm-pre-reloc;
+   reset-gpios = <&gpio4 RK_PA2 GPIO_ACTIVE_HIGH>;
+   vpcie3v3-supply = <&vcc3v3_pcie20>;
+   status = "okay";
+};
+
 &pinctrl {
     u-boot,dm-spl;
     /delete-node/ sdmmc;
diff --git a/configs/rk3588_defconfig b/configs/rk3588_defconfig
index f65da00444..f9fbaa145b 100644
--- a/configs/rk3588_defconfig
+++ b/configs/rk3588_defconfig
@@ -226,3 +226,15 @@ CONFIG_RK_AVB_LIBAVB_USER=y
 CONFIG_OPTEE_CLIENT=y
 CONFIG_OPTEE_V2=y
 CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION=y
+CONFIG_CMD_PCI=y
```

```
+CONFIG_NVME=y
+CONFIG_PCI=y
+CONFIG_DM_PCI=y
+CONFIG_DM_PCI_COMPAT=y
+CONFIG_PCIE_DW_ROCKCHIP=y
+CONFIG_PHY_ROCKCHIP_NANENG_COMBOPHY=y
+CONFIG_PHY_ROCKCHIP_SNPS_PCIE3=y
+CONFIG_DM_REGULATOR_FIXED=y
+CONFIG_ROCKCHIP_BOOTDEV="nvme 0"
+CONFIG_EMBED_KERNEL_DTB_ALWAYS=y
+CONFIG_SPL_FIT_IMAGE_KB=2560
--
2.17.1
```

9.2 RK3568 EVB1 Nor + eMMC 参考补丁

```
diff --git a/configs/rk3568_defconfig b/configs/rk3568_defconfig
index d157307403..82801f3dcb 100644
--- a/configs/rk3568_defconfig
+++ b/configs/rk3568_defconfig
@@ -222,3 +222,5 @@ CONFIG_RK_AVB_LIBAVB_USER=y
 CONFIG_OPTEE_CLIENT=y
 CONFIG_OPTEE_V2=y
 CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION=y
+CONFIG_ROCKCHIP_EMMC_IOMUX=y
+CONFIG_ROCKCHIP_BOOTDEV="mmc 0"
```

说明:

- 无需启用 Embedded Kernel DTB 配置