Rockchip Developer FAQ Storage

文件标识: RK-PC-YF-133

发布版本：V2.1.0

日期：2024-07-26

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

本文主要指导读者了解启动流程，对存储进行配置和调试。

各芯片 **feature** 支持状态

芯片名称	内核版本
所有产品	--

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

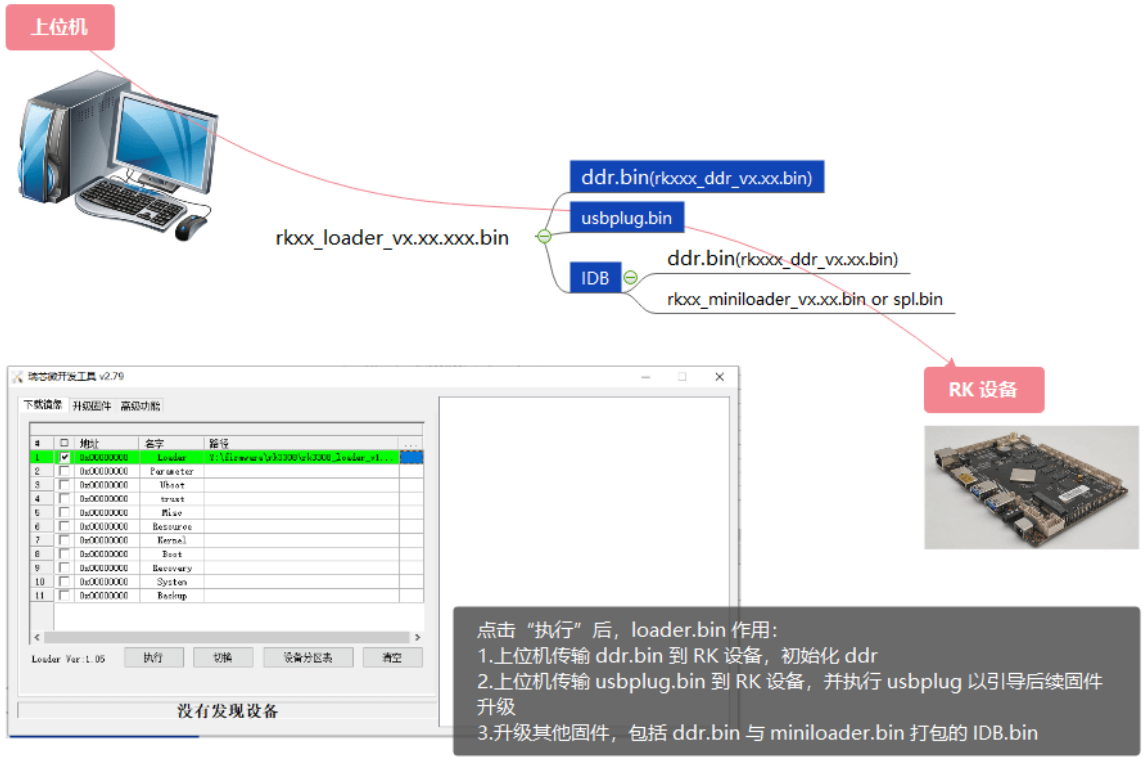
修订记录

版本号	作者	修改日期	修改说明
V1.0.0	赵仪峰、林鼎强	2021-05-08	初始版本
V1.1.0	林鼎强	2021-07-06	主要增加部分 UBIFS 异常案例
V1.2.0	林鼎强	2021-10-29	增加更多 Debug 实例
V2.0.0	林鼎强	2024-01-08	增加更多 Debug 实例，优化文档层级
V2.1.0	林鼎强	2024-07-26	增加更多 Debug 实例

目录

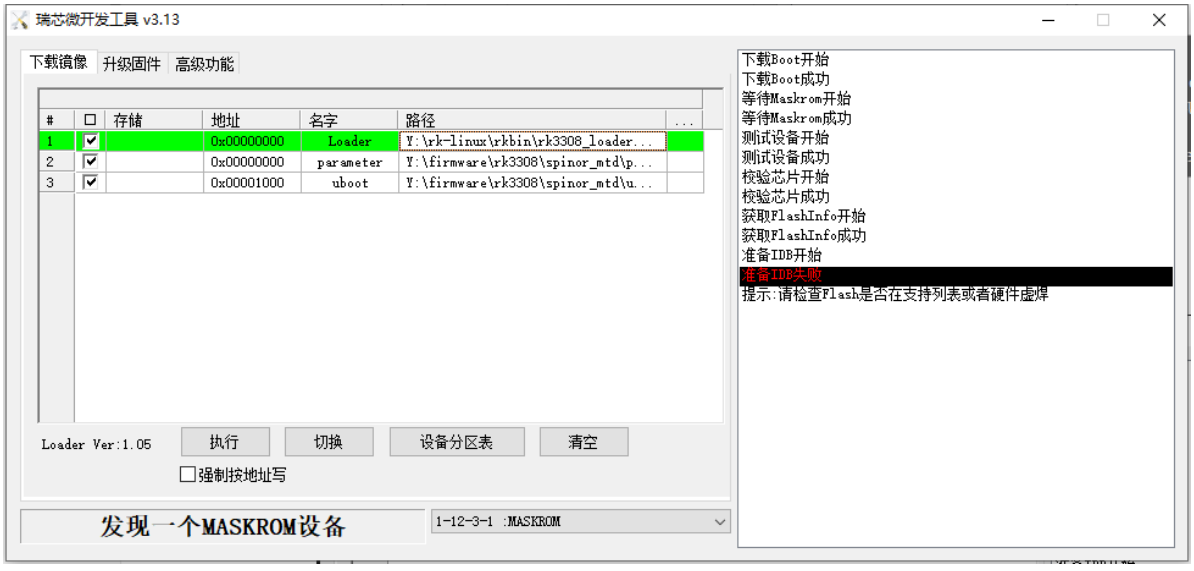
1. 固件烧录问题

1.1 烧录过程中 Loader 的作用



1.2 烧录 Flash 工具报错

工具界面：



工具 log：

```
18:05:40 125    Layer<1-1-2-3-1>:下载Boot成功
```

```
18:05:40 127    Layer<1-1-2-3-1>:等待Maskrom开始
18:05:40 552    Layer<1-1-2-3-1>:等待Maskrom成功
18:05:40 557    Layer<1-1-2-3-1>:测试设备开始
18:05:40 572    Layer<1-1-2-3-1>:测试设备成功
18:05:40 583    Layer<1-1-2-3-1>:校验芯片开始
18:05:40 592    Layer<1-1-2-3-1>:校验芯片成功
18:05:40 600    Layer<1-1-2-3-1>:获取FlashInfo开始
18:05:40 606    <LAYER 1-12-3-1> INFO:FlashInfo: 00 00 00 00 00 04 04 00 28 00 00
18:05:40 615    <LAYER 1-12-3-1> INFO:GetFlashInfo-->Emmc storage.
18:05:40 626    Layer<1-1-2-3-1>:获取FlashInfo成功
18:05:40 640    Layer<1-1-2-3-1>:准备IDB开始
18:05:40 645    <LAYER 1-12-3-1> ERROR:PrepareIDB-->No Found 1st Flash CS
18:05:40 663    Error:Layer<1-1-2-3-1>:准备IDB失败
18:05:40 686    Layer<1-12-3-1>: RunProc is ending, ret=0
```

设备 UART 串口打印：

```
ddr.bin log:     DDR Version V1.30 20191125
ddr.bin log:     REG2C: 0x00000031, 0x00000031
ddr.bin log:     In
ddr.bin log:     451MHz
ddr.bin log:     DDR2
ddr.bin log:     Col=10 Bank=2 Row=13 Size=64MB
ddr.bin log:     msch:0
ddr.bin log:     OUT
usbplug log:     Boot1 Release Time: Dec  7 2020 11:25:17, version: 1.25
usbplug log:     chip_id:000000,0
usbplug log:     DPLL = 1300 MHz
usbplug log:     ...nandc_flash_init enter...
usbplug log:     No.1 FLASH ID:ff ff ff ff ff ff                          /*
flash id 全 FF，未探测到 SLC Nand */
usbplug log:     DPLL = 1300 MHz
usbplug log:     sfc nor id: ff ff ff                                     /*
flash id 全 FF，未探测到 SPI Nor */
usbplug log:     DPLL = 1300 MHz
usbplug log:     sfc_nand id: ff ff ff                                    /*
flash id 全 FF，未探测到 SPI Nand */
usbplug log:     UsbBoot ...11943
usbplug log:     powerOn 664713
```

问题分析：

1. ddr.bin log 如有异常，请联系 DDR 工程师
2. 如果所用存储为并口 SLC Nand、SPI Nand 或 SPI Nor，如果升级过程 flash ID 为全 FF，则多为 flash vcc 供电异常、io 电平异常或焊接异常
3. 如果 flash ID 为正确值，打印 “The device not support yet!”，且所用颗粒在 《RK SpiNor and SLC Nand SupportList 》中显示相应平台已支持，则所用固件为旧版本，未添加该颗粒支持，先同步存储驱动：Redmine -> FAE 工程 -> 文档 -> Nand flash支持列表 -> 存储补丁连接
4. 如果同步后依旧无法正确下载，请联系 RK 存储小组工程师

1.3 烧录 SLC Nand 镜像无法启动

1.3.1 pin38 功能脚被使能

烧录过程常见 log：

```
prog read s error: = 7fc00 1fff0e0 ffffffff
prog read d error: = 7fc00 0 ffffffff
phyBlk = 0x1ff die = 0 block_in_die = 0x1ff 0x80000000
prog read s error: = 7f800 1fef0e0 ffffffff
prog read d error: = 7f800 0 ffffffff
phyBlk = 0x1fe die = 0 block_in_die = 0x1fe 0xc0000000
prog read s error: = 7f400 1fdf0e0 ffffffff
prog read d error: = 7f400 0 ffffffff
```

请确认 flash pin38 是否为功能脚，是否配置为除去使能对应电平：

- Lock#，通常为低有效，内部弱上拉，请悬空该脚或外接上拉电阻，以手册为准
- PT，通常为高有效，内部弱下拉，请悬空该脚或外接下拉电阻，以手册为准
- NC，无功能脚，请悬空

1.4 烧录 SPI Nand 镜像启动卡在 SPL 阶段

如有以下 log，则说明用户所用 SDK 未添加对应 spinand 的软件支持，请获取对应的存储补丁，补丁在 ”Redmine 》 FAE 项目 》 文档 》 Nand flash支持列表“ 下。

```
DDR Version V1.04 20201030
DDR, 328MHz
BW=32 Col=10 Bk=8 CS0 Row=15 CS=1 Die BW=16 Size=1024MB
change to 328MHz
change to 528MHz
change to 784MHz
change to 924MHz(final freq)
out
U-Boot SPL board init
U-Boot SPL 2017.09.g2d25c32e07-201118 #huke (Feb 26 2021 - 13:51:03)
unknown raw ID phN    # 该 log 说明 spl 驱动未支持该颗粒，需重新编译 spl，且核对并更新
uboot, kernel 对应补丁
unrecognized JEDEC id bytes: ff, 98, ed
Tring to boot from MMC1
Card did not respond to voltage select!
mmc_init: -95, time 9
spl: mmc init failed with error: -95
SPL: failed to boot from all boot devices
### ERROR ### Please RESET the board ###
# Reset the board to bootrom #
```

1.5 如何进入 Maskrom 模式

1. 有 maskrom 按键设备：先长按 reset 按键，再按下 maskrom 按键，松 reset（会看到进入工具显示 maskrom），再松 maskrom （次序很重要，避免异常）；
2. 无 maskrom 按键设备：maskrom 按键用 “flash 数据线 io0 短接地代替”。

1.6 如何单独编译测试 spl 固件

更改编译方式

SDK 使用编译好的 spl 固件，固件地址为 rkbin/bin/xx/ 目录，即使用 ./build.sh u-boot 命令编译出来的 loader 打包所用的 spl 固件非最新编译 spl 固件，可以参考以下命令使用自编译 spl 固件：

```
cd uboot
./make.sh rv1126
./make.sh spl-s
```

uboot 路径下输出文件：rv1126_spl_loader_vxxx.xxx.bin　uboot.img，请使用这个 loader.bin 和 uboot.img

如何确认 **spl** 已更新成功

"U-Boot SPL 2017.09-gf899931b5f-201209 #ldq (Dec 10 2020 - 15:34:13)"

可通过 spl log 确认，spl 是否有更新为自行编译的版本。

如何用自编译 **spl** 替换 **SDK spl**

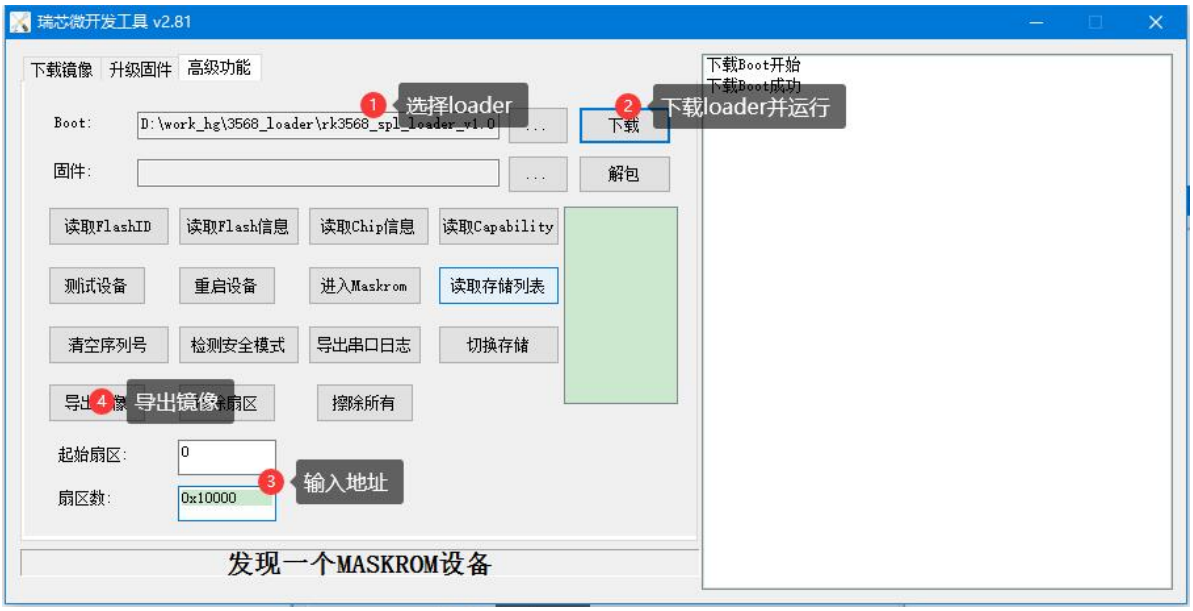
将编译后的 uboot/spl/u-boot-spl.bin 替换 ./rkbin/bin/xx/xxxx_spl_xxxx.bin

1.7 如何获取 Flash 镜像或 debug 信息

1.7.1 Export Image

Android tool

Maskrom mode 下导出固件：



Loader mode 下导出固件：

无需下载 loader，工具直接输入地址导出镜像。

Linux upgrade tool & Android_Console_Tool

Maskrom mode 下导出固件：

```
db MiniloadAll.bin
rl 0x0 0x10000 Exportimage.bin
```

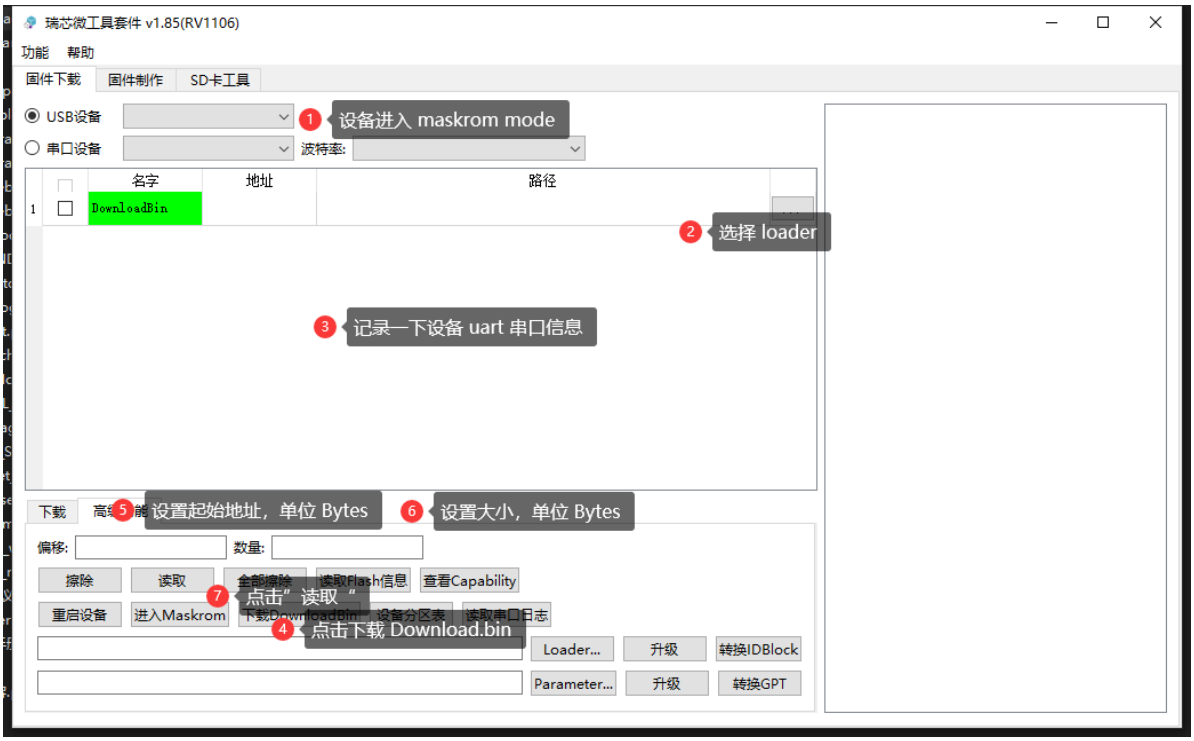
Loader mode 下导出固件：

```
rl 0x0 0x10000 Exportimage.bin
```

注意：

- 扇区地址单位为 512B per sector

toolskit

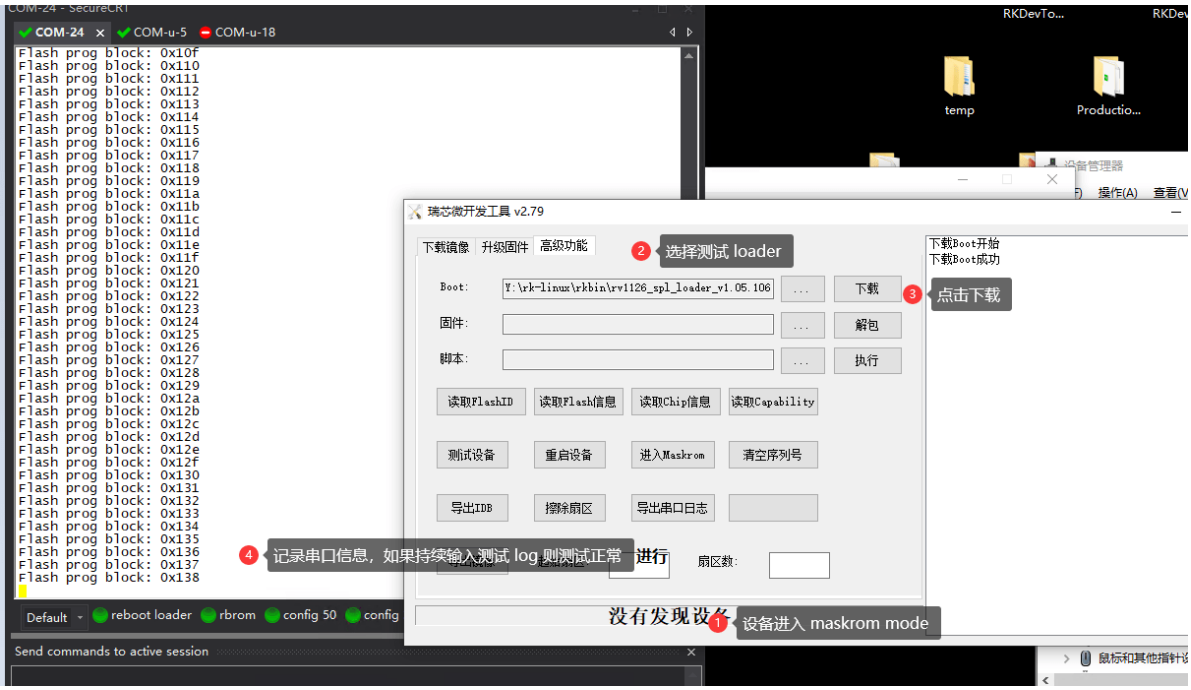


1.7.2 Dump All

特殊 loader 尾缀为 dump_all.bin，导出 RK ftl 特定的 flash 信息。

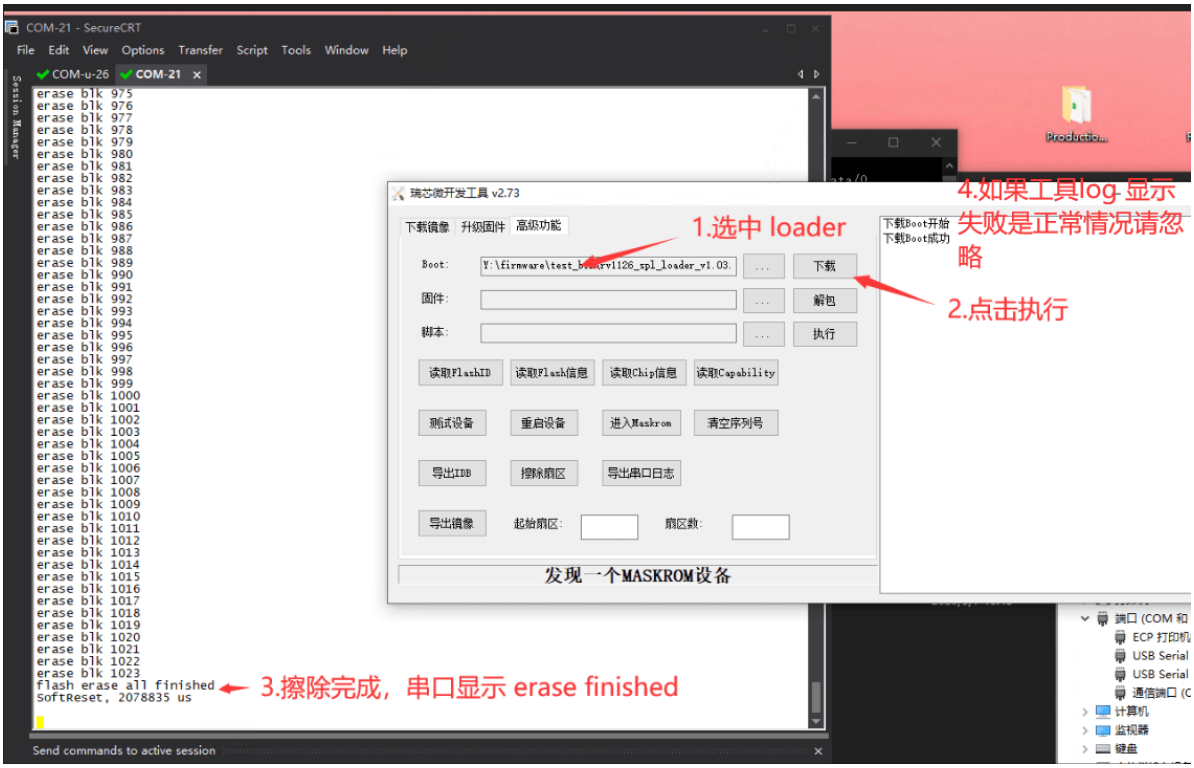
1.7.3 Flash Stress Test

特殊 loader 尾缀为 stress_test.bin，flash 简单读写擦除压测。



1.7.4 Flash Erase All

特殊 loader 尾缀为 erase_all.bin，完全擦除 flash 数据。



2. 文件系统问题

2.1 UBIFS 分区升级前未擦除整个分区

升级使用 UBIFS 的镜像时要求先将其分区整个擦除或者使用 ubiupdatevol 工具升级（推荐），通常出错信息如下：

```
...
[    4.452318] ubi0: attaching mtd5
[    4.685258] ubi0 error: ubi_attach: bad image sequence number 1251817256 in
PEB 543, expected 1051610977
[    4.685317] Erase counter header dump:
[    4.685334]  magic          0x55424923
[    4.685348]  version        1
[    4.685361]  ec             1
[    4.685374]  vid_hdr_offset 2048
[    4.685386]  data_offset    4096
[    4.685400]  image_seq      1251817256
[    4.685413]  hdr_crc        0x88225c8a
[    4.685426] erase counter header hexdump:
[    4.685753] ubi0 error: ubi_attach_mtd_dev: failed to attach mtd5, error -22
[    4.685814] UBI error: cannot attach mtd5
...
[    4.695772] ALSA device list:
[    4.695796]   #7: Loopback 1
[    4.696063] VFS: Cannot open root device "ubi0:rootfs" or unknown-block(0,0):
error -19
[    4.696091] Please append a correct "root=" boot option; here are the
available partitions:
...
```

建议工具版本：

- windows rkdevtool v2.79 及以上
- Linux_Upgrade_Tool_v1.59 及以上

2.2 UBIFS 镜像制作不匹配对应 flash 规格

注意制作镜像时使用的 mkfs 命令所带的 -e 和 -m 等与 flash 相关信息是否一致，如不一致，系统启动后将无法找到 UBI 相关信息，通常报错如下：

case 1

```
...
[    1.319729] ubi0: attaching mtd3
[    1.549286] ubi0: scanning is finished
[    1.549326] ubi0 error: ubi_read_volume_table: the layout volume was not found
[    1.549464] ubi0 error: ubi_attach_mtd_dev: failed to attach mtd3, error -22
[    1.549499] UBI error: cannot attach mtd3
...
[    1.563848]   #7: Loopback 1
[    1.564173] UBIFS error (pid: 1): cannot open "ubi0:rootfs", error -19VFS:
Cannot open root device "ubi0:rootfs" or unknown-block(0,0): error -19
...
```

case 2： 实际 page size 4KB，镜像 page size 2KB

```
[    0.517603] ubi0 error: validate_ec_hdr: bad VID header offset 2048, expected
4096
[    0.517618] ubi0 error: validate_ec_hdr: bad EC header
[    0.517626] Erase counter header dump:
[    0.517634]  magic          0x55424923
[    0.517639]  version        1
[    0.517645]  ec             0
[    0.517650]  vid_hdr_offset 2048
[    0.517657]  data_offset    4096
[    0.517664]  image_seq      2053699779
[    0.517672]  hdr_crc        0x2416c7f2
[    0.517679] erase counter header hexdump:
[    0.517701] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 4.19.111 #2
```

case3： 实际 page size 2KB，镜像 page size 4KB

```
[    1.678775] ubi0: attaching mtd6
[    1.679040] ubi0 error: validate_ec_hdr: bad VID header offset 4096, expected
2048
[    1.679054] ubi0 error: validate_ec_hdr: bad EC header
[    1.679068] Erase counter header dump:
[    1.679079]  magic          0x55424923
[    1.679090]  version        1
[    1.679101]  ec             0
[    1.679110]  vid_hdr_offset 4096
[    1.679120]  data_offset    8192
[    1.679131]  image_seq      362848614
[    1.679141]  hdr_crc        0x9438b345
```

```
[    1.679157] erase counter header hexdump:
[    1.679195] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 4.19.172 #8
[    1.679207] Hardware name: Rockchip RK3566 For 360 X200 (DT)
[    1.679220] Call trace:
[    1.679238]  dump_backtrace+0x0/0x178
```

2.3 UBIFS 镜像比分区大

减小镜像或增大分区，通常报错如下：

```
[    2.121208] ubi0 error: vtbl_check: too large reserved_pebs 824, good PEBs 767
[    2.121295] ubi0 error: vtbl_check: volume table check failed: record 0, error
9
[    2.121338] Volume table record 0 dump:
[    2.121375]  reserved_pebs   824
[    2.121412]  alignment       1
[    2.121449]  data_pad        0
[    2.121529]  vol_type        1
[    2.121568]  upd_marker      0
[    2.121605]  name_len        6
[    2.121642]  name            rootfs
[    2.121680]  crc             0x9ab0c6a3
[    2.122162] ubi0 error: ubi_attach_mtd_dev: failed to attach mtd3, error -22
[    2.122278] UBI error: cannot attach mtd3
```

2.4 UBIFS 固件尾部被破坏

该问题需要进一步分析问题，通常报错如下：

```
[    1.867789] ALSA device list:
[    1.867830]   #7: Loopback 1
[    1.916907] UBIFS (ubi0:0): UBIFS: mounted UBI device 0, volume 0, name
"rootfs", R/O mode
[    1.917010] UBIFS (ubi0:0): LEB size: 126976 bytes (124 KiB), min./max. I/O
unit sizes: 2048 bytes/2048 bytes
[    1.917061] UBIFS (ubi0:0): FS size: 103231488 bytes (98 MiB, 813 LEBs),
journal size 9023488 bytes (8 MiB, 72 LEBs)
[    1.917106] UBIFS (ubi0:0): reserved for root: 0 bytes (0 KiB)
[    1.917155] UBIFS (ubi0:0): media format: w4/r0 (latest is w5/r0), UUID
1C1DDBB4-77E6-4AA9-855F-ED8F10EB5916, small LPT model
```

2.5 UBIFS 日记保留过大

UBIFS 为日志文件系统，如果日志保留过大而分区又小，可能无法正常挂载。

可以考虑调整参数：

```
-j, --jrn-size=SIZE       journal size
```

通常报错如下：

```
[root@RV1126_RV1109:/]# mount -t ubifs /dev/ubi2_0 /record/
[   65.788339] UBIFS error (ubi2:0 pid 890): ubifs_read_superblock: mount:
mounting /dev/ubi2_0 on /record/ failed: Invalid argument
too large journal size (8388608 bytes), only 3555328 bytes available in the main
area
[   65.78[root@RV1126_RV1109:/]# 8394] UBIFS error (ubi2:0 pid 890):
ubifs_read_superblock: bad superblock, error 1
[   65.788413]  magic          0x6101831
[   65.788434]  crc            0x663d9574
[   65.788455]  node_type      6 (superblock node)
[   65.788475]  group_type     0 (no node group)
[   65.788493]  sqnum          1
[   65.788512]  len            4096
[   65.788531]  key_hash       0 (R5)
[   65.788620]  key_fmt        0 (simple)
[   65.788639]  flags          0x4
[   65.788660]  big_lpt        0
[   65.788684]  space_fixup    1
[   65.788706]  min_io_size    2048
[   65.788728]  leb_size       126976
[   65.788755]  leb_cnt        39
[   65.788773]  max_leb_cnt    2048
[   65.788786]  max_bud_bytes  8388608
[   65.788799]  log_lebs       5
[   65.788812]  lpt_lebs       2
[   65.788826]  orph_lebs      1
[   65.788838]  jhead_cnt      1
[   65.788851]  fanout         8
[   65.788864]  lsave_cnt      256
[   65.788877]  default_compr  1
[   65.788890]  rp_size        0
[   65.788905]  rp_uid         0
[   65.788918]  rp_gid         0
[   65.788931]  fmt_version    4
[   65.788945]  time_gran      1000000000
[   65.788961]  UUID           DBC08BA0-15FE-49BD-BC47-8A9200F1D625
```

2.6 UBI block 建立 SquashFS 缺少 ZLIB 支持

开启宏 CONFIG_SQUASHFS_ZLIB，通常报错如下：

```
[    4.564088] rockchip-mipi-dphy-rx: No link between dphy and sensor
[    4.564102] rkisp0: update sensor failed
[    4.564181] ALSA device list:
[    4.564191]   #0: rockchip,rk809-codec
[    4.564200]   #7: Loopback 1
[    4.566057] squashfs: SQUASHFS error: Filesystem uses "zlib" compression. This
is not supported
[    4.566158] List of all partitions:
[    4.566179] fd00           31792 ubiblock0_0
[    4.566181]  (driver?)
[    4.566196] No filesystem could mount root, tried:
[    4.566197]  squashfs
[    4.566206]
```

```
[    4.566225] Kernel panic - not syncing: VFS: Unable to mount root fs on
unknown-block(253,0)
[    4.570008] CPU: 3 PID: 1 Comm: swapper/0 Not tainted 4.19.149 #3
[    4.570547] Hardware name: Generic DT based system
[    4.570993] [<b010f47c>] (unwind_backtrace) from [<b010b9a8>]
(show_stack+0x10/0x14)
[    4.571677] [<b010b9a8>] (show_stack) from [<b0827b68>] (dump_stack+0x90/0xa4)
[    4.572317] [<b0827b68>] (dump_stack) from [<b01264fc>] (panic+0x114/0x2a0)
[    4.572962] [<b01264fc>] (panic) from [<b0c013a8>]
(mount_block_root+0x2a4/0x2f4)
[    4.573629] [<b0c013a8>] (mount_block_root) from [<b0c015d0>]
(prepare_namespace+0x150/0x1
```

2.7 UBIFS 空分区容量过小制作镜像后无法挂载

UBIFS 最小分区:

```
Minimum block num = 4（固定预留） + B + 17  /* B - 为坏块替换预留的 flash blocks,与
ubiattach - b 参数相关 */
```

可通过 ubiattach 时打印 log 来判断，例如：

```
ubi4: available PEBs: 7, total reserved PEBs: 24, PEBs reserved for bad PEB
handling: 20   /* B = 20 */
```

如果分区 available PEBs + total reserved PEBs < Minimum block num，则挂载时会报错：

```
mount: mounting /dev/ubi4_0 on userdata failed: Invalid argument
```

2.8 UBIFS 选用 LZO 还是 ZLIB

RK SDK 默认选用 LZO 压缩，特定 cace 103M 压缩 到 80M，而 ZLIB 压缩率相近，如有更高压缩率需求，可使用 SquashFS。

2.9 UBIFS 镜像展开大于分区大小

通常制作 UBIFS 镜像时使用 autoresize 特性，如果在以下命令对应的配置文件中指定了镜像大小，很可能造成镜像展开超过分区大小的可能：

```
ubinize -o ubi.img -m 2048 -p 128kiB -O 2048 ubinize.cfg

//ubinize.cfg
[ubifs]
  mode=ubi
  image=ubifs.img
  vol_id=0
  vol_size=192MB                  // 无需指定大小，使用 autoresize 特性
  vol_type=dynamic
  vol_name=rootfs
  vol_flags=autoresize
```

通常报错信息如下：

```
[    2.345243] ubi0: attaching mtd2
[    2.415631] ubi0: scanning is finished
[    2.419742] ubi0 error: vtbl_check: too large reserved_pebs 1586, good PEBs
699
[    2.419785] ubi0 error: vtbl_check: volume table check failed: record 0, error
9
[    2.419792] Volume table record 0 dump:
[    2.419798]  reserved_pebs   1586
[    2.419803]  alignment       1
[    2.419808]  data_pad        0
[    2.419813]  vol_type        1
[    2.419818]  upd_marker      0
[    2.419823]  name_len        6
[    2.419829]  name            rootfs
[    2.419835]  crc             0x23e79a61
[    2.420000] ubi0 error: ubi_attach_mtd_dev: failed to attach mtd2, error -22
[    2.420024] UBI error: cannot attach mtd2
```

UBIFS 在 loader mode 下升级改动后的 gpt 无法正常挂载

RK SDK mtd 启动方案，如在 loader mode 下升级 ”gpt 改动的固件“，则会导致所要升级的目标固件升级在 u-boot 启动时所生成的旧坏块映射关系上，因此 loader mode 下仅支持 gpt 没有改动的固件升级，否则，请转 maskrom mode 下升级，如果误升级，通常有以下 log：

case 1：

```
[    0.659432] ubi0: attaching mtd3
[    1.264467] ubi0: scanning is finished
[    1.271151] ubi0 warning: ubi_read_volume_table: static volume 0 misses 1 LEBs
- corrupted
[    1.302342] ubi0: volume 0 ("rootfs") re-sized from 613 to 636 LEBs
[    1.303492] ubi0: attached mtd3 (name "rootfs", size 85 MiB)
[    1.303526] ubi0: PEB size: 131072 bytes (128 KiB), LEB size: 126976 bytes
[    1.303533] ubi0: min./max. I/O unit sizes: 2048/2048, sub-page size 2048
[    1.303540] ubi0: VID header offset: 2048 (aligned 2048), data offset: 4096
[    1.303547] ubi0: good PEBs: 677, bad PEBs: 3, corrupted PEBs: 0
[    1.303559] ubi0: user volume: 1, internal volumes: 1, max. volumes count: 128
[    1.303568] ubi0: max/mean erase counter: 1/0, WL threshold: 4096, image
sequence number: 980761742
[    1.303578] ubi0: available PEBs: 0, total reserved PEBs: 677, PEBs reserved
for bad PEB handling: 37
[    1.303599] ubi0: background thread "ubi_bgt0d" started, PID 80
```

case 2:

```
[11:34:08:931][    0.661124] ubi0: attaching mtd6
[11:34:08:931][    0.662524] ubi0 error: ubi_attach: bad image sequence number
67638577 in PEB 1, expected 980761742
[11:34:08:931][    0.662571] Erase counter header dump:
[11:34:08:931][    0.662578]  magic          0x55424923
[11:34:08:931][    0.662584]  version        1
[11:34:08:931][    0.662590]  ec             0
[11:34:08:931][    0.662595]  vid_hdr_offset 2048
[11:34:08:931][    0.662601]  data_offset    4096
[11:34:08:931][    0.662606]  image_seq      67638577
[11:34:08:947][    0.662611]  hdr_crc        0x3a381cee
[11:34:08:947][    0.662616] erase counter header hexdump:
[11:34:08:949][    0.662645] ubi0 error: ubi_attach_mtd_dev: failed to attach
mtd6, error -22
[11:34:08:949][    0.662675] UBI error: cannot attach mtd6
[11:34:08:949][    0.662691] UBI: block: can't open volume on ubi0_-1, err=-19
```

2.10 UBIFS SPI Nor 支持

强调：

SPI Nor 建议使用 JFFS2 作为可读写文件系统选择。

内核配置：

```
CONFIG_MTD_SPI_NOR_USE_4K_SECTORS = n
```

制作镜像：

```
mkfs.ubifs -r data/ -m 1 -e 65408 -c 62 userdata.ubifs
ubinize -o ubi.img -m 1 -p 64KiB -O 64 ubinize_userdata.cfg

➜  ubifs cat ubinize_userdata.cfg
[userdata-volume]
mode=ubi
image=userdata.ubifs
vol_id=0
vol_size=2MiB
vol_type=dynamic
vol_name=userdata
vol_flags=autoresize
```

说明：

- 4KB 擦除无法支持原因，c->leb_size < UBIFS_MIN_LEB_SZ，其中 UBIFS_MIN_LEB_SZ 15KB，而 leb_size 随 mtd 属性

2.11 UBIFS userdata 异常时低格修复方案建议

userdata 低格修复方案建议：

- 将频繁写的文件存放在同一个分区
- 避免将不可修复的文件放在该分区
- 当分区出现挂载异常，按照以下方式进行修复：
  - 格式化分区
  - 重新生成关键数据以支持设备继续运行

低格命令参考：

```
ubidetach -m 7
ubiformat -y /dev/mtd7
ubiattach /dev/ubi_ctrl -m 7 -d 7
ubimkvol /dev/ubi7 -N userdata -m
mount -t ubifs /dev/ubi7_0 /userdata
```

2.12 UBIFS vol_name 未准确命名导致异常

异常 log：

```
[    6.278922] VFS: Cannot open root device "ubi0:rootfs" or unknown-block(0,0):
error -19
[    6.278943] Please append a correct "root=" boot option; here are the
available partitions:
```

说明：

- 该 log 为 rootfs 分区 vol_name 未定义为 rootfs 而导致

2.13 UBIFS squashfs 开机耗时长

对于希望获取文件系统更高可靠性，且对开机时间不敏感的 SPI Nand 产品，建议 squashfs 文件系统对应的 UBI 镜像制作时使用 static volume 属性，上电过程能对文件系统的完整性做校验：

- RK 当前 SDK 默认带有该属性
- 如需修改该支持，请用户自行维护数据完整性校验

问题

客户及本地测试发现 spinand 支持的 UBI 镜像上的 squashfs 文件系统，上电加载耗时较长，达到秒/10MB 级别，例如 50MB UBI 镜像分区，加载达到 5秒。

分析

制作 UBI 镜像时可选为 static 或 dynamic volume，其中：

- static 支持为只读，且上电挂载流程会对整个镜像做 crc 完整性校验，适合 kernel/iniramdisk 等没有文件系统的镜像做进一步镜像校验
- dynamic 支持可读写

UBIFS 官方文档建议

[UBIFS 官方文档说明]

> Why a dynamic volume is faster to access than a static volume of the same size?
> Static UBI volumes were originally designed to store blobs of data like configuration files. Static volumes provide CRC-32 protection for the stored data, and static volumes know how much data they store. E.g., if you have a static volume /dev/ubi0_1 of 254KiB in size consisting of 2 127KiB LEBs, and you store a 200KiB file there, and you read whole /dev/ubi0_1 , you'll get exactly 200KiB. If /dev/ubi0_1 was a dynamic volume, you'd read 254KiB. So static volumes make it easier to store blobs of data.
>
> Thus, static volumes are slower to access (comparing to dynamic volumes) because UBI has to check the data CRC-32 checksum. And note, it is not a good idea to use static volumes with R/O UBIFS because UBIFS also protects all the information it stores in the UBI volume with CRC-32 checksums, and using static volumes with UBIFS would only slow down UBIFS mount speed.

其中建议：

- 由于 ubifs 自带 crc 校验，所以不建议 UBI 镜像再使用 static volume 属性来做进一步 CRC 校验

同理：

- 由于 squashfs 文件系统上电没有对整个镜像作完整性校验，所以建议制作的 UBI 镜像时添加 static volume 属性

实测

TEST01：启动时间测试：

RV1106 搭配 spinand，使用 dynamic volume UBI镜像，支持 squashfs

- 启动正常
- 启动过程没有额外的写数据开销（只读）
- 除去冗余的 CRC 校验，加载时间正常

TEST02：squashfs 仅有压缩解压缩，不具备完整性校验，不添加 static 属性

测试方式：

- 制作带 dynamic volume 属性的 squashfs UBI 镜像
- 定位到目标文件，窜改部分数据
- 烧录镜像
- 启动后读取镜像

发现确实破坏文件数据，ubifs 没有完整性校验。

TEST03：squashfs 仅有压缩解压缩，不具备完整性校验，添加 static 属性，增加校验报错

测试方式：

- 制作带 static volume 属性的 squashfs UBI 镜像
- 定位到目标文件，窜改部分数据
- 烧录镜像
- 启动后读取镜像

发现上电过程 UBI 层完成扫描并打印相应出错信息：

```
[    6.358100] ubi0 warning: ubi_eba_read_leb: CRC error: calculated 0xae6dc941, must be 0xf62e2df5
[    6.358212] ubi0 warning: ubi_open_volume: volume 0 on UBI device 0 is corrupted
```

附录：static 与 dynamic 修改点

```
[rootfs-volume]
mode=ubi
image=rootfs.squashfs
vol_id=0
vol_size=60MiB
vol_type=static
vol_name=rootfs
vol_alignment=1
vol_flags=autoresize
```

修改为：

```
[rootfs-volume]
mode=ubi
image=rootfs.squashfs
vol_id=0
vol_size=60MiB
vol_type=dynamic
vol_name=rootfs
vol_alignment=1
vol_flags=autoresize
```

2.14 UBIFS 镜像 4KB 页比 2KB 页颗粒同样大小分区占用冗余空间更多

合理，因为 ubifs 固定是会保留固定数量的 flash block 用作算法处理，所以相同大小的分区，4KB page size 颗粒所占用的冗余会更多。

2.15 UBIFS df 命令空间较小

可以参考 UBIFS 社区描述 “Why does df report too little free space? ” 章节。UBIFS 需要计入冗余、压缩、算法，所以无法准确估计分区可存储空间，且倾向保守估算，所以 df 命令看上去显得较少，但实际传输可能比 df 值更多。

2.16 JFFS2 文件系统挂载速度慢

JFFS2 启动过程会做垃圾回收，特性如此，包括 ls 命令都会比较慢，且越用会越慢。

2.17 JFFS2 分区未擦除

更新升级工具版本，新版本工具会识别 JFFS2 镜像并在升级前擦除分区：

```
mount -t jffs2 /dev/block/by-name/userdata /userdata
[    2.232850] jffs2: Empty flash at 0x002c1204 ends at 0x002c2000
[    2.232931] jffs2: CLEANMARKER node found at 0x002c2000, not first node in
block (0x002c0000)
[    2.233065] jffs2: Empty flash at 0x002c200c ends at 0x002c3000
[    2.233075] jffs2: CLEANMARKER node found at 0x002c3000, not first node in
block (0x002c0000)
[    2.233428] jffs2: Empty flash at 0x002c300c ends at 0x002c4000
```

2.18 JFFS2 与 SPI Nor 4K 擦除冲突

由于 JFFS2 最小擦除对齐为 8KB，而 SPI Nor 内核框架只支持 4KB or 64KB 擦除对齐。

所以如果用户需要使用 JFFS2，SPI Nor 内核不能开启 CONFIG_MTD_SPI_NOR_USE_4K_SECTORS 宏配置，即只能用 64KB 擦除对齐。

2.19 YAFFS 文件系统不建议使用

非常不建议使用 yaffs，开源社区都没有在维护，建议转 UBIFS。

2.20 YAFFS2 文件系统 RK 方案不作支持

主要原因：

- RK spinand 平台在 UBIFS 上的支持较为健全，建议使用 UBIFS
- YAFFS2 涉及 oob 相关操作，page 支持多次写入数据，目前测试无法兼容

自行开发：

- 如果客户需要开发 YAFFS，相关控制器驱动在以下目录，请自行开发:
    - kernel 4.19 或以下版本，drivers/rfkash/
    - kernel 5.10，drivers/spi/spi-rockchip-sfc.c

2.21 EXT 文件系统 Nand 方案无法使用

主要原因：EXT 没有适合 Raw-Nand 的 FTL 算法，不支持磨损均衡、数据刷新、坏块管理等对 Nand 设备极为关键的算法逻辑。

2.22 EXT2 文件系统 SPI Nor 支持

强调：

无磨损均衡，慎用，擦除对齐 4KB

说明：

- 开启 CONFIG_MTD_SPI_NOR_USE_4K_SECTORS 宏
- SPI Nor 支持 ext2，只需挂载将分区挂载在 mtdblock 设备上即可

- 避免频繁做文件系统写操作，SPI Nor 通常 P/E 也就是擦除写寿命为 100K 次，请自行评估产品应用寿命。

2.23 EXT2 不支持 dicard 特性导致文件系统后期读写速率明显降低

ext2 不支持 discard，包括最新的 linux 系统也未作支持，所以 ext2 文件系统下，如果 flash 全空间被使用满过，删除文件仅修改 inode，没有 discard 机制去通知底层存储设备作相应的释放动作，一旦出现存储全空间被文件占用后，就无法释放。

建议：

- 文件重复使用，避免频繁创建，还有避免大文件一直占满全空，如果无法完全规避，可以考虑定期低格对应磁盘，mkfs.ext2 会调用 discard 释放空间
- 或者直接迁移到 ext4，挂载时带 discard 参数

2.24 SquashFS 压缩格式及其效率

以下内容仅作为初步参考，实际压缩效率受数据类型等因素影响：

Method	Ratio	Compression MB/s	Decompression MB/s
gzip	2.92	15	128
lzo	2.64	9.5	217
lz4	2.12	94	218
xz	3.43	5.5	35
xz 256 KB	3.53	5.4	40
zstd 1	2.71	96	210
zstd 5	2.93	69	198
zstd 10	3.01	41	225
zstd 15	3.13	11.4	224
zstd 16 256 KB	3.24	8.1	210

数据来源：squashfs: Add zstd support

3. EMMC 问题

3.1 EMMC Kernel 下概率性出现通信出错

这个问题可能会有多种原因，目前碰到的情况列举如下：

VDDI 外接电容太小

EMMC 颗粒的控制器需要一个 1.2V 左右的工作电压，一般通过 LDO 从 VCCQ 降压得到，VDDI 上的电容就是对 1.2V 工作电压提供滤波。

大部分 EMMC 颗粒封装时内部 VDDI 管脚已经有接一颗小电容，不同厂家和不同型号的 EMMC，由于控制器不同和内部接的电容值不同，对 VDDI 外接电容的要求也会不同。

一般颗粒对外接 VDDI 电容的要求见下表：

运行模式	**VDDI 电容**
HS400	1 - 2.2uF（个别颗粒 2-4.7uF）
非 HS400 模式	0.1uF - 2.2uF

当外接 VDDI 电容太小或者电容失效时，就可能出现用 A 颗粒没有问题，换用 B 颗粒就出现概率性通信出错。

对于 VDDI 电容取值建议（上限值或者接近上限值）：

平台	运行模式	**VDDI 电容值**
RK3399	HS400	2.2uF
其他平台	HS200，DDR50, SDR 50	1uF

主控端 VCCQ 上电容未接或太小

EMMC 5.1 读写速度都非常快，而且容量越大，写速度越快，VCC 和 VCCQ 的瞬时电流可能超过 400mA（大部分颗粒在 200mA 左右），如果主控端 VCCQ 上电容未接或者太小，那么在开机上电或者使用时可能会出现主控 VCCQ 塌陷，造成主控读取 EMMC 端数据出错。

EMMC DATA 线接了 ESD 器件

SD 卡是需要接 ESD 器件，EMMC 是贴在板上，没有热插拔行为，不需要接 ESD。有些 ESD 器件电容太大，严重影响信号，造成 HS400 和 HS200 通信出错。

EMMC CLK/CMD 走线和其他高速信号距离太近

EMMC CLK/CMD 走线和其他高速信号（比如 HDMI）太近，由于信号干扰，造成通信出错。

EMMC 走线不等长并在 CMD 或者 CLK 上串接比较大电阻

EMMC 在 HS200 和 HS400 模式时时钟都是 200Mhz，走线不等长并且 CMD 或者 CLK 上串接比较大电阻，会造成 EMMC 读取数据的有效窗口变小，在电压和温度变化时容易出现都数据出错。

3.2 EMMC 升级固件时找不到存储器件或校验出错

造成这个问题的原因也比较多，目前碰到的情况列举如下：

VCCQ 电压选择错误

VCCQ 供电支持 1.8V 和 3.3V，同时主控的 IO 也需要配置匹配模式，一般通过一个 IO 接 VCC 或地来选择，具体参考原理图。

VCCQ 电压	主控 IO 工作模式	结果
1.8V	1.8V	正常
1.8V	3.3V	没法工作
3.3V	1.8V	低速时可能正常，高速时会出错
3.3V	3.3V	正常

VDDI 电容太小

参考 1.1 VDDI 外接电容太小

EMMC 颗粒焊接问题或者损坏

EMMC 颗粒的一些 DNU/NC BALL 是需要保留的，可能是内部是接地或者 VCCQ，如果布板时有和其他 IO 连在一起，可能就会出现个别型号颗粒数据线短路或者电源短路。
手工焊接 EMMC 颗粒时可能存在虚焊，造成主控认不到
贴片过回流焊时温度过高或者颗粒受潮，造成颗粒损坏，主控认不到

3.3 EMMC 使用一段时候后损坏或固件丢失

主要碰到几种情况：

EMMC 颗粒有问题

颗粒问题一般存在两种情况：

1. 个别颗粒存在坏块等原因，在使用一段时间后出现数据损坏（重新升级主控固件可以修复）或者 EMMC 颗粒的 FW 出错（这种需要换颗粒），这种出问题概率一般小于千分之一。
2. EMMC 的 FW 存在问题，造成批量的丢数据或者 FW 出错，这种出错概率会比较大，需要 FFU 升级 EMMC 的固件或者重新开卡。

系统写入数据太多，颗粒寿命用完了损坏

应用软件（包含 android）设计不合理或者配置不合理，写入太多日志（小数据随机写），对于 EMMC 来说，写入放大比较大，比如日志一次写 1KB，但是 EMMC 实际需要写入一个 page（16KB），日志一般是写文件，文件系统部分也有更新，实际总写入可能是 3 次，48KB。

EMMC 5.1 颗粒的 EXT CSD 里面有颗粒寿命信息，DEVICE_LIFE_TIME_EST_TYP_B [269] 、PRE_EOL_INFO [267] 和DEVICE_LIFE_TIME_EST_TYP_B [268] 。

EXT CSD读取方法：

1. 通过mmc-utils工具自己读取，详细参考mmc-utils工具自带的帮助
2. linux控制器cat emmc的ext_csd节点, 不同系统，这个节点的位置可能不同，可以find ext_csd找到。

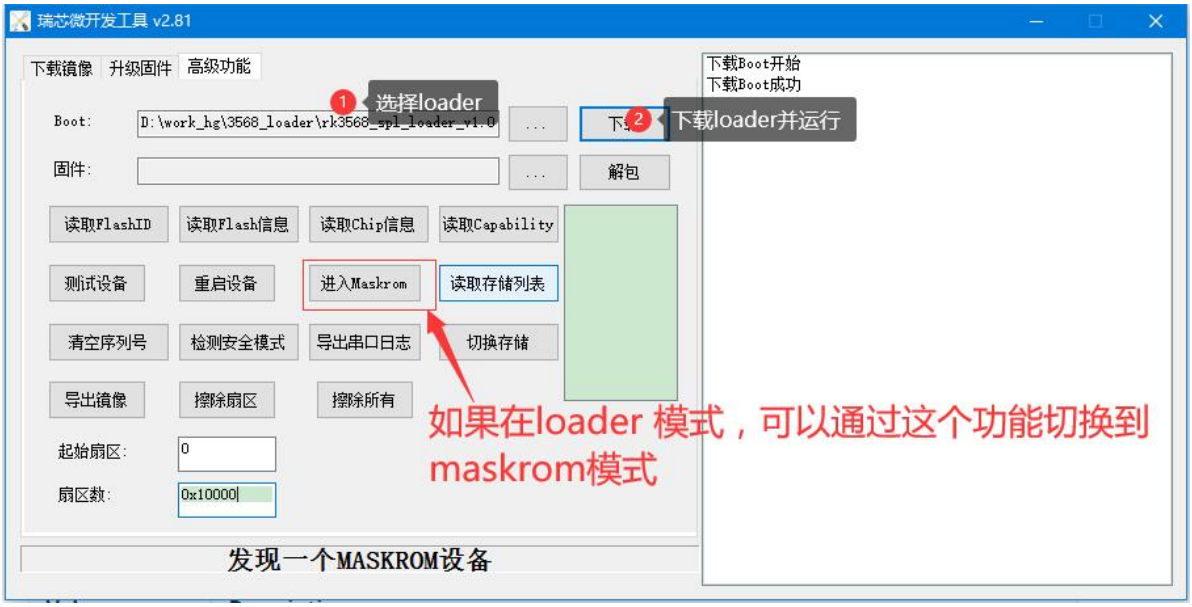
```
cat /sys/kernel/debug/mmc0/mmc0:0001/ext_csd
```

3. maskrom升级模式下，下载文件名带有dump_ext_csd的调试loader，串口会打印出ext csd信息

Linux upgrade tool & Android_Console_Tool

```
db xxxloaderxxx.bin
```

Android tool



EXT CSD寿命信息：

DEVICE_LIFE_TIME_EST_TYP_B [269] 和 DEVICE_LIFE_TIME_EST_TYP_B [268] 记录EMMC颗粒擦写寿命，详细信息见下表：

Value	Description
0x00	Not defined
0x01	0% - 10% device life time used
0x02	10% -20% device life time used
0x03	20% -30% device life time used
0x04	30% - 40% device life time used
0x05	40% - 50% device life time used
0x06	50% - 60% device life time used
0x07	60% - 70% device life time used
0x08	70% - 80% device life time used
0x09	80% - 90% device life time used
0x0A	90% - 100% device life time used
0x0B	Exceeded its maximum estimated device life time
Others	Reserved

PRE_EOL_INFO [267] 记录EMMC颗粒的保留块是否用完，详细信息见下表：

Value	Pre-EOL Info.	Description
0x00	Not Defined	
0x01	Normal	Normal
0x02	Warning	Consumed 80% of reserved block
0x03	Urgent	
0x04 ~ 0xFF	Reserved	

OTA 升级过程异常掉电

ANDROID 7.1 之后 OTA 固件升级时是直接对 system 分区进行块更新（非原来基于文件更新），如果 OTA 升级过程异常掉电，有可能会造成 system 分区数据损坏。

3.4 EMMC 升级固件后启动停留在maskrom模式

EMMC没有焊接好或者PCB有问题

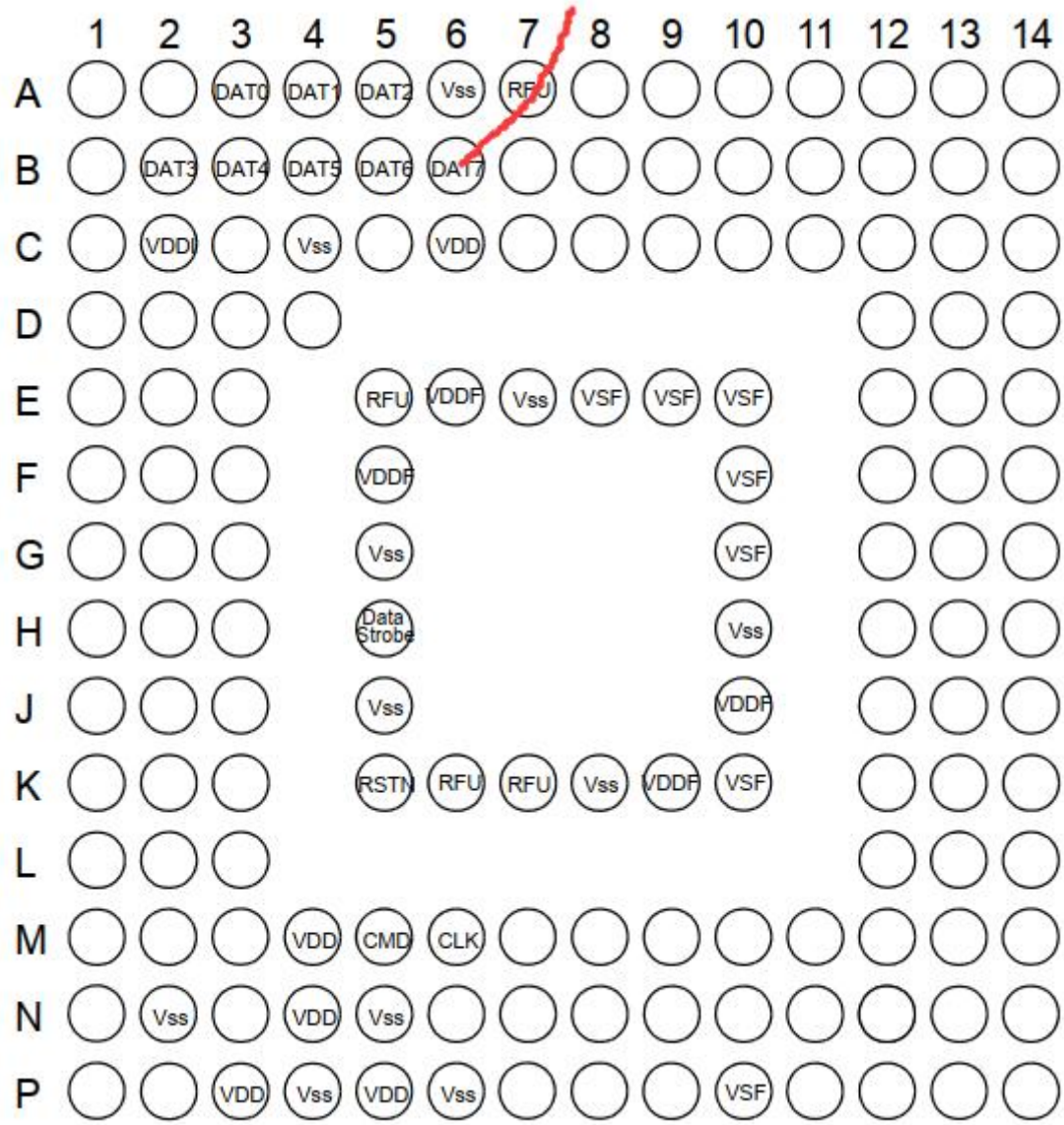
BOOTROM启动会用8线模式，如果EMMC D1-D7有线没有接好，就没法启用8线模式。

焊接不良：

个别样机有问题，一般是焊接不良引起

PCB问题：

所有样机都有问题，一般是PCB问题，比如布线时数据线通过了RFU或者DNU的ball。



未烧录uboot或者trust：

uboot或者trust没有烧录，或者烧录位置错误，miniloader没法启动到uboot时会返回maskrom升级模式。通过串口打印信息可以看到有启动到miniloader，然后在重启。

4. Flash 问题

4.1 Flash 读写速率

通用测试（接近最优）：

dd 工具测试至少 10MB 的空间。

颗粒类型	测试环境	连续读	连续写
SLC Nand	Linux	16MB/s（IO 30MHz）	4MB/s（IO 30MHz）
SPI Nand	Linux	10.8MB/s （IO 80MHz）	4.0 MB/s （IO 48MHz）
SPI Nor	Linux	24.0MB/s（IO 80MHz）	100~200 KB/s（IO 48MHz）
SPI Nor	RTT	35MB/s (IO 80MHz)	100~200KB/s（IO 80MHz）

颗粒类型	测试环境	随机 4K 读	随机 4K 写
SPI Nor	RKOS	13MB/s (IO 80MHz)	100~200 KB/s（IO 48MHz）

特殊测试环境写速率 1（接近饱和）

测试环境：RK3308_V10_EVB

内核：4.4 64 位

用户分区：352MB

FLASH 大小：512MB

测试 cace：

```
Sub Main
    while (1)
        crt.Screen.Send "rm /userdata/test && sync"
        crt.Screen.Send chr (13)
        crt.Sleep 2000
        crt.Screen.Send "time dd if=/dev/zero of=/userdata/test bs=1M count=350
&& sync"
        crt.Screen.Send chr (13)
        crt.Screen.waitForString "real"
        crt.Sleep 2000
    wend
End Sub
```

测试结论：测试半小时， 满载写速率 2.0 MB/s

4.2 Flash 写保护功能

Flash 颗粒的写保护功能

- 软件写保护，颗粒通常支持但实现方式各式各样
- 硬件 wp# 功能脚写保护，部分颗粒支持， RK 主控无法控制该 pin 脚，如果一定要用，必须转 GPIO 控制

RK 存储方案不添加写保护支持

因为存储的读写为统一接口，理论上也不会有异常信号能触发完整读写行为，且不同颗粒保护方案大相径庭，所以不添加相应的写保护处理。

4.3 Flash 软件两片选颗粒无法兼容

两颗同类型的 flash 叠 die 的颗粒如果软件片选，现有的软件框架无法支持。

4.4 Flash 内核存储初始化不成功常见打印

假定，存储初始化异常（异常肯定也不会有对应 mtd 分区创建），系统一样会往下运行，直到在挂载非 ramdisk 的根文件系统时停下来，并报有 waiting for root..... 相关打印。

4.5 Flash 写存储设备节点数据如何确保数据落盘

fopen 添加 O_SYNC 或者 O_DIRECT 符号，或者写数据后添加 fsync，如果使用 dd 命令升级镜像，建议添加以下 flag：

- iflag=nocache 和 oflag=direct

4.6 Flash 更换物料的说明

关于支持列表中 **TA/SA** 物料的区别

RK SPI Flash 控制器 IP 为兼容性 IP，T/A 为经过存储验证平台验证过的样片，S/A 为兼容性平台，有 T/A S/A 的 flash 都可导入产品。

关于 **Nand flash** 新物料

RK 颗粒验证主要验证基础功能、送样样片在测试平台的稳定性，所以具体产品设备建议针对 flash 做进一步测试，建议包括：

- 信号一致性，更换 flash 前后信号无明显差异，符合颗粒 spec 要求
- 异常掉电健壮性，参考文档
  《Rockchip_Developer_Guide_Linux_Flash_Open_Source_Solution_CN.pdf》

4.7 Flash TFTP 下载固件烧录异常定位

初步定位问题：

- 在以下阶段分别添加 md5sum 信息，并对比：
  - 制作升级镜像后添加 md5sum
  - TFTP 下载固件，固件添加 md5sum
    - 如果比对异常，怀疑为网络或 dram 问题
  - flash 烧写固件后在线回读，回读镜像添加 md5sum
    - 如果比对异常，怀疑 flash 工作异常，建议 flash 降速、降为单线传输进一步确认
  - 设备进入 maskrom mode，回读镜像添加 md5sum
    - 结合 “flash 烧写固件后在线回读” 分析

如初步怀疑为 flash 问题：

- 降速降单线有效，建议测量信号确认信号质量、测量 flash 供电是否存在不稳定，调整硬件看是否默认配置能工作
- 如降速降单线无效，联系 RK 负责存储的软件工程师进一步定位

4.8 Flash uboot 阶段 mtd 接口对比 mtd_block 接口

P/E 粒度不匹配问题：

Nor Flash 和 Nand Flash 一样，都是最小擦除（Erase）与最小写粒度（Program）不一致，所以需要考虑 P/E 结合。例如：写 Nor 中某一个 4K sector 中的 256B，需要先擦除 4K ，同时 256B 地址外的 （4KB - 256B） 数据需要预先保存并回写。

mtd 和 block 接口：

- mtd 接口要求上层自行管理 P/E
- block 接口会抽象最小读写逻辑扇区，读写该扇区不会影响其他扇区(内部自行处理 P/E 粒度不匹配)
    - linux 框架有提供一个 mtdblock 接口，将逻辑扇区规划到 512B（可想而至，接口会维护 P/E 关系，避免在写入某一个 512B 扇区时遇到 P/E 粒度不匹配问题）
    - RK SDK u-boot 参考 linux 框架完成相应逻辑

4.9 Flash 低温拷机异常

建议先测量低温下 VCCIO Flash sel 上下拉电平是否异常。

4.10 Flash 开发板 maskrom 按键易于触碰影响数据传输

maskrom 按键应设计在不易触碰的位置，避免误触影响数据传输。

5. Flash 控制器问题

5.1 Controller FSPI 时钟占空比不到 50%

输出时钟由时钟树特定 PLL 分频而来，如为奇数分频，时钟占空比不一定为 50%，将导致外设最高频时序较为临界，而偶数分频占空比为 50%。

5.2 Controller FSPI 非通用 SPI 控制器，仅支持 spi memory 协议

FSPI（旧称 SFC） 仅支持符合 spi memory 时序的应用，即 cmd + addr（可选）+ dummy（可选）+ data（可选），尤其是“不支持cs assert 后直接 input 的操作”，请仔细核对时序。

如需扩展外接例如 FPGA 外设，请自行评估是否符合扩展要求，内核驱动参考 spi-rockchip-sfc.c。

5.3 Controller FSPI 建立保持指标不做测量

uboot 和 kernel 阶段 FSPI 读数据默认都是配置为四线传输，如果速率配置为一致，Host 读采样的建立保持时间要求理论上是一致的。

背景

1. FSPI 读采样基础

FSPI SPI mode 0（CPOL clock 闲时低电，CPHA 采样第一个边沿），在这个基础上推半个周期读采样 —— 负沿采样

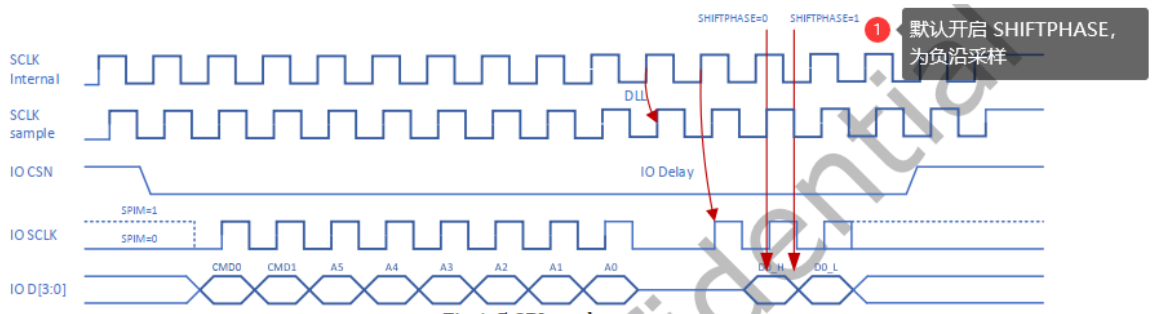


Fig.1-5 SPI mode

2. DLL（delay line ） 需求

在 io 时延（FSPI 时序约束）和绝大部分颗粒 output delay 综合情况下，读数据信号的建保时间较为临界，所以需要通过添加 delay line 支持将获取最佳建立保持时间。

3. DLL 特点

RK host controller FSPI 支持读采样支持 delay line：

- DLL 在芯片内部叠加，所以 IO 上无法测量到该数值
- DLL 受 PVT 和工艺影响，所以非定值，也非锁相位
- 延时效果叠在在 ”FSPI 读采样基础“ 里的负沿采样上

综上，DLL 具体延时无法计算，且不会体现在信号上。

4. DLL 软件策略

且 IO 速率超过 50MHz 时即开启支持，包括以下特征：

- 各个软件阶段都会去 tuning
- 比对 flash id 值

综述

不测量 RK Host FSPI 读数据的建保时间主要是因为：

- FSPI IO 工作时钟大部分会高于 50MHz，启用 DLL，所以实际读数据建保时间无法测量，参考 ”DLL 特点“ 描述
- 低于 50MHz，基于 "FSPI 读采样基础" 有较大冗余值

5.4 Cotroller FSPI 是否支持两片选分别挂不同器件

如果过该芯片方案下的 FSPI 控制器扩展 CS1N IO，IP 理论上就支持双片选，但仅有 CS0N 片选上的器件为 bootdev，且双片选上的器件复用较多 IO，信号之间存在干扰，对于器件的速率通常存在影响，请做全面的软硬件评估。

6. Nand 问题

6.1 PX30 Nand 设备修改为 rkflash 存储驱动

关于 rk_nand 和 rk_flash 存储框架的介绍请参考 《Rockchip_Developer_Guide_UBoot_Nextdev_CN》 文档 Storage 章节，rk_nand 主要应用于 MLC、TLC 等大容量 Nand flash 产品，rk_flash 应用于 SLC Nand flash 产品，px30 提供的 SDK 多为大容量 Nand 固件，则改用 SLC Nand flash，存储驱动应作相应改动。

主要修改点：

1. rkbin 选用 RKBOOT/PX30MINIALL_SLC.ini 配置打包出来的 loader 固件
2. u-boot 参考 Rockchip_Developer_Guide_UBoot_Nextdev_CN Storage 章节，改 RKNAND 驱动为 RKFLASH
3. kernel 类似 u-boot 修改，改 RK_NAND 为 RK_FLASH，dts 使能 nandc0 节点

可能存在以下修改异常：

- Q1: u-boot 无法正常引导 boot?
    - A1：可能是存储驱动或者 dts 没有配置成功，输入以下命令做相应确认：

```
=> dm tree
...
# 正常设备应有以下类似打印
rksfc [ + ] rksfc |-- sfc@ff4c0000 **
blk [ + ] rkflash_blk | |-- sfc@ff4c0000.spinand.blk
blk [ ] rkflash_blk | `-- sfc@ff4c0000.spinor.blk
```

6.2 Nand uboot 用户自定义读写行为规范

如果擦写次数不频繁，可以考虑直接调用 mtd 接口作擦写读，但部分客户存在 uboot 下频繁记录信息的需求，存在以下情景：

- 裸分区，未使用文件系统
- 频繁记录
- 不想并入 vendor 分区，避免造成影响

为了减少原地擦写对 Nand 产品寿命的影响，建议：

- 新增一个特殊分区，参考 vendor 分区实现读写，源码参考：arch/arm/mach-rockchip/vendor.c，接口参考 mtd_vendor_storage_init 和 mtd_vendor_write。

简介 vendor 分区实现原理：

- 初始化过程探测是否存在 vnvm 分区，如存在注册 vendor 接口

- 一次写两份，每次写入的单位是对齐到页大小，写入位置为块的第一页，擦除全块

6.3 Nand SFTL 空间规划

颗粒	idblock	user data	reserve data	sys block	冗余
128MB	2MB	107MB	11MB	8MB	14.8%
256MB	2MB	218MB	24MB	12MB	14%
512MB	2MB	464MB	31MB	I5MB	9%

其中 11M 为 flash 做存储交换使用，能提交读写效率，sys block 为 FTL 专用。

6.4 Nand 烧录器预烧录不允许合并分立镜像一次烧录

原理上烧录器最终实现肯定是分立烧录：

- flash 软件运行过程坏块管理局限于对应分区，所以预烧录行为应与之一致

部分烧录器支持自行开发或外界开源的的特殊打包方式：

- 将分立的镜像打包成一个 all_in_one 的数据包（可能自带分区信息），然后烧录器解析特有的分区信息，拆包，最终分立烧录
    - 请烧录器厂家提供 demo 或自行开发。

RK 曾解析过一款烧录器 all in one 的打包方式：

- 详细参考 spinand 烧录器制作文档

  20210804_flash_programer

6.5 Nand 开源框架坏块表管理

坏块表管理的需求

Nand flash 几乎所有行为都要感知所要操作的 flash block 是否为坏块，而获取到的坏块信息如不记录，则每次都要从 flash page 中的坏块标记区域去重新判断是否有坏块标记，这样将带来以下几个问题：

- 扫描 flash 坏块标记区域是 io 读行为，需要耗费一定的时间；
- 在使用过程中出现的坏块上记录坏块标记的可靠性有限，flash block 在不稳定情况下不一定能成功标记上可识别的坏块标记。

mtd 框架会将第一次从 io 中获取的坏块信息做一个 bad block table 缓存，然后第二次读取重复位置的 flash block 坏块信息可以直接从 bad block table 缓存中获取坏块信息，但重启上电后，bad block table 缓存又要重新建立，所以除此之外还应将 bad block table 贮存在非易失的存储里。（bbt 缓存参考 drivers/mtd/nand/bbt.c）

SLC Nand bbt in flash

SLC Nand 开源框架中就集成了可写回 flash 的坏块表管理方式，将坏块信息集中管理和更新

SPI Nand bbt in flash

开源框架中并不支持 SPI Nand 可写回 flash 的坏块管理方式，因此，rk 提供了相应的坏块管理支持，且默认开启，详细代码：

u-boot:

```
drivers/mtd/nand/bbt.c
```

kernel:

```
drivers/rkflash/sfc_nand_mtd_bbt.c
```

该坏块表管理方式有以下几个特点：

1. 结合开源框架 drivers/mtd/nand/bbt.c 原有的缓存 flash 坏块信息的 bbt.cache 框架部分；
2. 应预留 Nand flash 最后 4 个 block 为坏块表存储可用块，最后一个分区应有相应调整；
3. 第一次上电在 u-boot 环境下生成坏块表；
4. 每次写回两份坏块表；
5. u-boot 下坏块因为没有真实的 mutex 和多线程情况所以不会涉及死锁问题。

7. PP SLC NAND 问题

7.1 PX30 SLC Nand IDB 制作问题

不同镜像制作方式不同：

1. IDB 镜像制作：烧录器带 oob 方式导出 idb 镜像，地址为 flash block 0，长度通常为 2 flash block。
2. 其他镜像制作：使用 programmer_image_tool 添加 -l 参数来修改链表链接方式

说明：

- px30 IDB 烧录器镜像单独制作，主要是因为旧平台对 slc nand 的 IDB 镜像区域还添加了 1 page 的 flash info 信息，并添加特殊 ECC， programmer_image_tool 没有集成（新平台没有这个 flash info），所以不兼容

7.2 SLC Nand 颗粒原厂分析存在连续大量的坏块

客户或原厂将 SLC Nand flash 拆片后用烧录器或者其他工具检测后认为 RK 产品使用后块出现了标记为坏块的块。

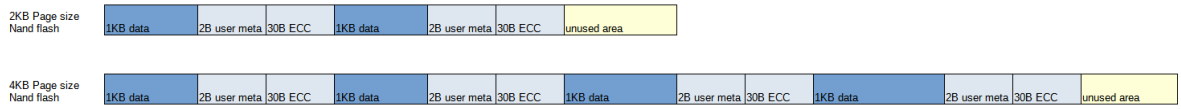
实际上这种判断方式时错误的，RK FTL 使用过的 SLC Nand，RK AP使用坏块表管理坏块，被使用的块不能再去判断 spare 0 的值是否是坏块标记,所以拆下颗粒，用第三方工具判断是不准确的，所以实际使用中遇到问题后视具体情况分析。

7.3 SLC Nand 烧录器烧录不支持母片镜像

假定“镜像” 只有 data 数据：

1.SPI Nand 因依赖颗粒端 ECC，所以用户烧录的镜像仅为 data 数据，ECC 依赖颗粒端

2.SLC Nand 依赖主控端生成的 BCH 码做 ECC 保护，而且 one page 经过 RK BCH 处理后 data 和 oob（including meta 数据和 BCH code）会按照以下规律存放：



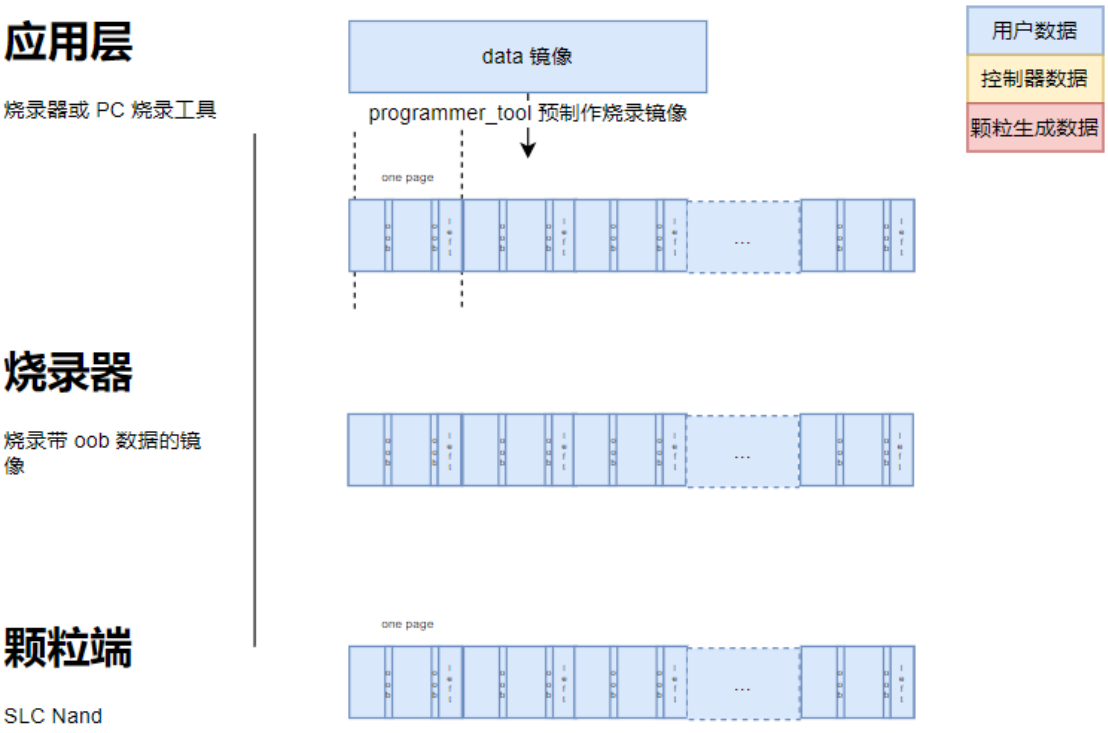
说明：

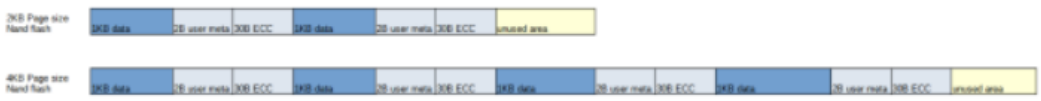
- Re-layout：原本 data + oob 的默认布局会被打算为 data + oob + data + oob ... 的零散布局，原来的 oob first byte 坏块标记处重新布局好处于数据区：
    - 烧录固件后的 SLC Nand 无法重新按照原厂默认坏块标记判断方法判断，也就是 first oob byte
    - RK 软件确认 bad block 的方法为先确认 BCH fail，再确认 first oob byte 为坏块标记

最终导致，烧录器回读数据时无法正常 skip bad block。

7.4 SLC Nand 烧录器烧录镜像为有效数据及冗余 oob 数据

oob 数据（meta data 和 BCH）在制作镜像时这一层添加，烧录烧录镜像设置”烧录镜像为 data 带 oob”

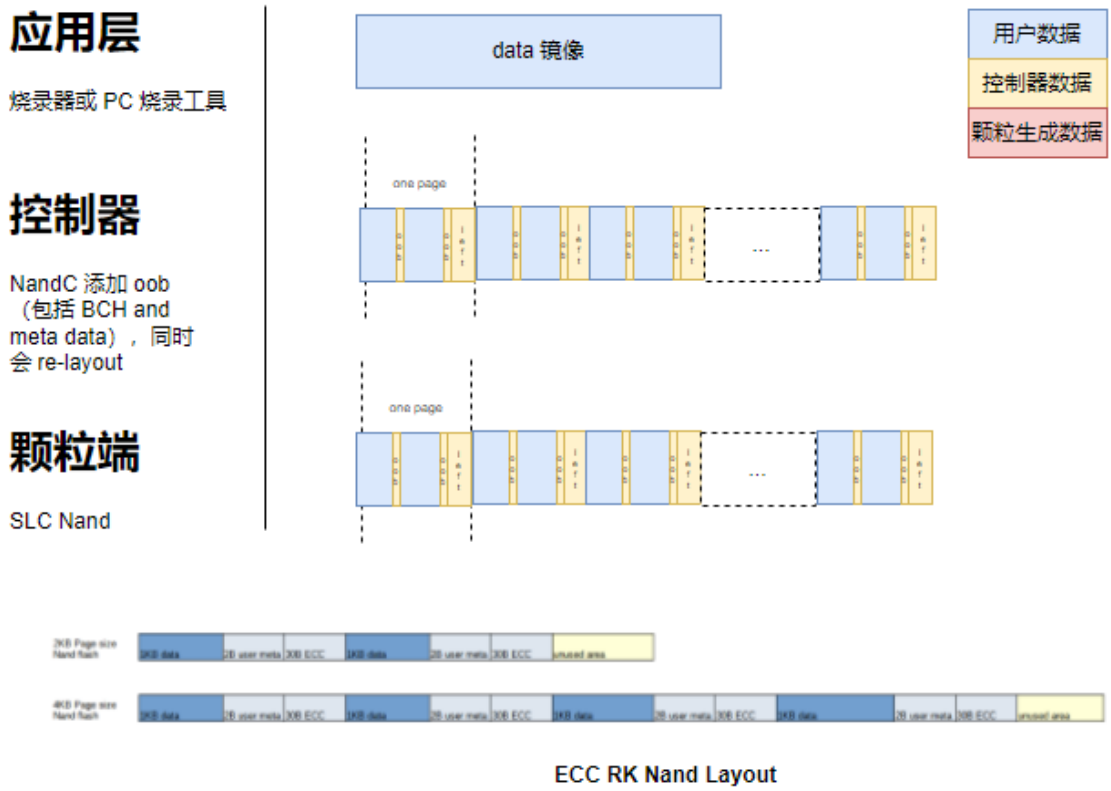




ECC RK Nand Layout

7.5 SLC Nand SDK 输出镜像为有效数据（不包含冗余 oob 数据）

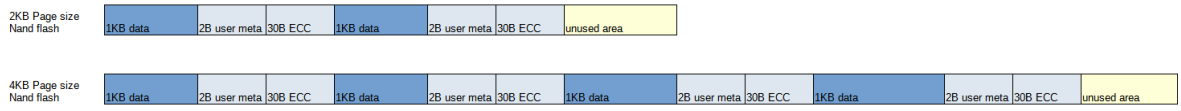
oob 数据（meta data 和 BCH）在控制器这一层添加，用户用来烧录的镜像只包含 data。



ECC RK Nand Layout

7.6 SLC Nand 软件中 ECC Layout

RK SDK slc nand（并口）使用的是 host controller 集成的 bch 模块做 ECC，并设置为 16bits/1KB，布局如图:



7.7 SLC Nand rkflash 方案内核阶段读写速率优化

牺牲 cpu 调度:

1. 更新存储补丁，链接：

   存储补丁

2. 修改 drivers/rkflash/nandc.c 驱动中的 usleep_range 从 20-30 到 5-10 —— 调整 usleep 粒度以优化 CPU 线程调度（但是会牺牲部分传输速率）

反之，如果增大 usleep 时长，则降低 IO 吞吐率高负载情况下的 CPU 负载。

7.8 SLC Nand mtd 方案 spl 阶段读传输速率优化

背景

由于 PP Nand 颗粒标准不一，所以 NandC 控制器配置会相对保守，追求稳定性，如果客户产品对速率有较高的要求，可以做适当的提速优化。

更新补丁确认

以 RV1126 为例，uboot 添加补丁，编译更新 spl 支持 30MHz IO 速率：

```
diff --git a/arch/arm/mach-rockchip/rv1126/rv1126.c b/arch/arm/mach-
rockchip/rv1126/rv1126.c
index 84df82ab138..e3ae330e413 100644
--- a/arch/arm/mach-rockchip/rv1126/rv1126.c
+++ b/arch/arm/mach-rockchip/rv1126/rv1126.c
@@ -77,6 +77,7 @@ DECLARE_GLOBAL_DATA_PTR;
 #define CRU_CLKSEL_CON50       0x1c8
 #define CRU_CLKSEL_CON51       0x1cc
 #define CRU_CLKSEL_CON54       0x1d8
+#define CRU_CLKSEL_CON59       0x1ec
 #define CRU_CLKSEL_CON61       0x1f4
 #define CRU_CLKSEL_CON63       0x1fc
 #define CRU_CLKSEL_CON65       0x204
@@ -749,6 +750,11 @@ int arch_cpu_init(void)
        udelay(1);
        writel(0x00100000, CRU_BASE + CRU_SOFTRST_CON10);

+#ifdef CONFIG_SPL_BUILD
+       writel(0x00ff004, CRU_BASE + CRU_CLKSEL_CON59);
+       printf("NandC gpll=768,div=(%d+1)\n", readl(CRU_BASE + CRU_CLKSEL_CON59)
& 0xff);
+#endif
+
        return 0;
 }
 #endif
diff --git a/drivers/mtd/nand/raw/rockchip_nand_spl.c
b/drivers/mtd/nand/raw/rockchip_nand_spl.c
index 6e39287ef39..d6ed7038914 100644
--- a/drivers/mtd/nand/raw/rockchip_nand_spl.c
+++ b/drivers/mtd/nand/raw/rockchip_nand_spl.c
@@ -92,7 +92,7 @@ static u32 nand_block_num;

 static void nandc_init(struct rk_nand *rknand)
 {
-       writel(0x1081, rknand->regs + NANDC_REG_V6_FMWAIT);
+       writel(0x1061, rknand->regs + NANDC_REG_V6_FMWAIT);
 }

 static void rockchip_nand_wait_dev_ready(void __iomem *regs)
```

要求

由于 spl 阶段 NandC 提速，建议做一下确认：

1. 信号质量，确认输出信号是否满足器件 spec 要求
2. 拷机压测，可以直接拷机 reboot 测试

7.9 SLC Nand 尾部过多坏块 MTD 方案设备无法正常工作

如果内核为 4.19 版本，请先确认是否更新以下补丁：

```
commit e323080273b1e6ad2862158c2c634bf9f36671b2
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Tue Aug 29 15:56:56 2023 +0800

    mtd: rawnand: rockchip: Modify the judgment of the last page

    Incorrect judgment without affecting bad block judgment logic.

    Change-Id: I123be12cc50e52e6a24c4276a5ac68208239af42
    Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
```

8. SPI Nand 常见问题

8.1 SPI Nand 颗粒 plane select 物料兼容问题

问题：

RV1126/RV1109 芯片对于部分 2 plane 物理结构的 spinand 颗粒的支持存在兼容性问题，需要裁剪 IDB size 才能 bring up，所以不推荐使用该类颗粒，这一点 support list 上会体现。

兼容方案：

如果一定要用到这种颗粒，RK 有做特殊兼容处理，即裁剪 spl 做相应支持，裁剪后的 defconfig 为 configs/rv1126-spl-spi-nand_defconfig。

主要裁剪：

1.其他存储类型支持：并口 nand、emmc、spinor、SD 卡

2.gpt 支持

3.AB 分区

如果，有以上功能需求，可以考虑，再做裁剪，以便开启以上功能，裁剪方向，去除以下没有用到的原厂颗粒定义：

```
CONFIG_MTD_SPI_NAND=y
CONFIG_SPI_NAND_GIGADEVICE=y
CONFIG_SPI_NAND_MACRONIX=y
CONFIG_SPI_NAND_MICRON=y
CONFIG_SPI_NAND_TOSHIBA=y
CONFIG_SPI_NAND_WINBOND=y
CONFIG_SPI_NAND_DOSILICON=y
CONFIG_SPI_NAND_ESMT=y
CONFIG_SPI_NAND_XTX=y
CONFIG_SPI_NAND_HYF=y
CONFIG_SPI_NAND_FMSH=y
CONFIG_SPI_NAND_FORESEE=y
CONFIG_SPI_NAND_BIWIN=y
```

确认 **spl.bin + ddr.bin** 合并的 **IDB** 大小是否满足要求：

uboot 目录下输入以下命令，其中 ddr.bin 改为目标

```
./tools/mkimage -n rv1126 -T rksd -d
./../rkbin/bin/rv11/rv1126_ddr_924MHz_v1.05.bin:./spl/u-boot-spl.bin
rv1126_idb1.bin
```

已知存在该问题颗粒：

颗粒丝印	**ID0**	**ID1**
MT29F2G01ABA, XT26G02E, F50L2G41XA	2C	24
DS35Q2GA-IB	E5	72
DS35M2GA-IB	E5	22
FM25S02A	A1	E5
DS35Q2GB-IB	E5	F2
DS35Q4GM	E5	F4

进一步解释

bootrom 没法扫描到这些颗粒的 block 1 3 5... 的奇数 block 上的数据

一个 block 是 128KB

所以一旦 IDB size 超过 128KB，必然跨 block，bootrom 肯定是扫不到 IDB 部分数据，无法正确加载 IDB

8.2 SPI Nand 颗粒 1V8 物料选型

1V8 的颗粒不会太难找，原厂通常都会有 1V8 的产品。RK 已经支持有 7 家颗粒原厂，以最新 flash 支持列表为准。

1V8 颗粒支持，除了 domain 配置和 io 供电需要转好，存储驱动上和 3V3 兼容，无特殊配置：

20230705	**XTX**	**XT26Q04DWSIGA**	**4GBits**	**512MB**	**1.7V-1.95V**	**SPI NAND**
20230705	XTX	XT26Q01DWSIGA	1GBits	128MB	1.7V-1.95V	SPI NAND
20230705	XTX	XT26Q02DWSIGA	2GBits	256MB	1.7V-1.95V	SPI NAND
20230705	FORESEE	F35UQA002G-WWT	2GBits	256MB	1.7V-1.95V	SPI NAND
20230705	FORESEE	F35UQA001G-WWT	1GBits	128MB	1.7V-1.95V	SPI NAND
20230317	HYF	HYF1GQ4IDACAE	1GBits	128MB	1.70V-1.98V	SPI NAND
20230317	HYF	HYF2GQ4IAACAE	2GBits	256MB	1.70V-1.98V	SPI NAND
20230217	Dosilicon	DS35M12B-IB	512MBits	64MB	1.7V-1.95V	SPI NAND
20220901	MXIC	MX35UF1GE4AD-Z4I	1GBits	128MB	1.7V-1.95V	SPI NAND
20220818	MXIC	MX35UF4GE4AD-Z4I	4GBits	512MB	1.7V-1.95V	SPI NAND
20220818	MXIC	MX35UF2GE4AD-Z4I	2GBits	256MB	1.7V-1.95V	SPI NAND
20210918	GigaDevice	GD5F2GM7REYIGY	2GBits	256MB	1.7V-2.0V	SPI NAND
20210911	Dosilicon	DS35M2GA-IBR	2GBits	256MB	1.7V-2.0V	SPI NAND
20210911	Dosilicon	DS35M1GA-IBR	1GBits	128MB	1.7V-1.95V	SPI NAND
20210911	Dosilicon	DS35M1GB-IB	1GBits	128MB	1.7V-1.95V	SPI NAND
20210911	GigaDevice	GD5F2GQ5REYIG	2GBits	256MB	1.7V-2.0V	SPI NAND
20210911	GigaDevice	GD5F1GQ5REYIG	1GBits	128MB	1.7V-2.0V	SPI NAND
200701	Dosilicon	DS35M1GA-IBR	1GBits	128MB	1.65V-1.95V	SPI NAND
200831	MXIC	MX35UF1GE4AC-Z4I	1GBits	128MB	1.7V-1.95V	SPI NAND
200831	MXIC	MX35UF2GE4AC-Z4I	2GBits	256MB	1.7V-1.95V	SPI NAND

8.3 SPI Nand 颗粒 HYF1GQ4UTXCAE 物料异常掉电问题

原厂送样的 HYF1GQ4UTXCAE 颗粒异常掉电 fail，确认为系列颗粒 upstream 缺少调整 HWP_EN 支持

```
commit 72986913fc4ba3e0c1c276aa63e2040ff8290201
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Mon Aug 14 14:52:36 2023 +0800

    mtd: spinand: Enable HWP_EN for skyhigh devices

    HWP_EN must be enabled first before block unlock region is set.

    Change-Id: I6b107d97de48bb2644da865f353d2adace95224e
    Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
```

8.4 SPI Nand 软件 Soft ECC 方案预研

首先阐明，软件 ECC 对于纠错几乎没有帮助，对于数据有效性的判断有所帮助，但 RK 方案只使用颗粒自带的硬件 ECC。

关于 ECC 简介：

- spinand 的数据需要 ECC 校验，因为会 Nand 介质有一定概率出现位翻转
- 通常为硬件 ECC，可能是集成在颗粒（大部分颗粒都有集成），可能集成在 host（例如海思芯片）

关于 软件 ECC 的需求：

如果 SPI Nand 没有集成硬件 ECC，RK 方案默认不支持该颗粒，因为其可靠性极低，如果颗粒已经有硬件 ECC，还考虑做软件 ECC：

- 对纠错几乎没有帮助，如果颗粒自身的硬件 ECC 出现纠错 fail 那么通常为大面积异常，软件的少量纠错就没有帮助
- 可作为数据是否有效的判断标志，因为有可能在极限情况，传输通路上出现异常（比如异常掉电），这种情况硬件 ECC 未必报错，所以要有类似 UBIFS 内的数据校验机制

软件 ECC 的应用场景：

对于 misc 等无文件系统的分区，如果进行 SPI Nand 读写行为，可以考虑添加 ECC 机制，做极限状况的数据防护，抑或是采用其他行之有效的数据保护机制。

8.5 SPI Nand 软件 BBT store in flash 保护机制

背景

RK SPI Nand SDK 支持 bbt 表存储 flash 坏块信息，详细参考《Rockchip_Developer_FAQ_Storage_CN.pdf》“坏块表管理的需求” 章节。

早期方案

早期对于坏块表的校验主要依靠：

- oob first byte
- 贮存在 bbt 中有效数据尾部的 magic

对应补丁版本：

u-boot:

```
commit 78cac1dffdc0bd64e47294e41d2432b7760e7186
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Sun Jun 28 20:13:39 2020 +0800

    mtd: nand: Remove bbt option property if scan fail

    Change-Id: Ifb5b500b6ffee551aea5b6aecea629b3d0ea6207
    Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
```

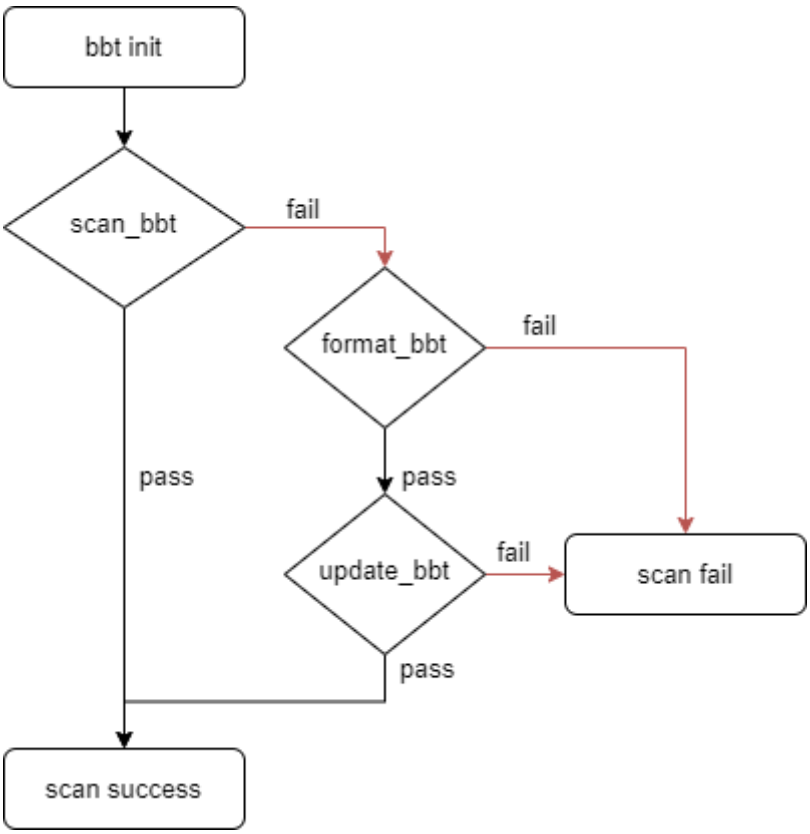
kernel:

```
commit fe246b156492ee509bb1d071c176d324542d3193
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Thu Dec 17 16:05:27 2020 +0800

    drivers: rkflash: Fixed bbt operation calculation error

    Fixes: 008340a82 ("drivers: rkflash: Change to use the api which the oob area
available")
    Change-Id: I140aa76a2acb73271ba04b7060030dc06a2353e6
    Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
```

优化方案

考虑到 SPI Nand 存在不稳定时，虽然根源并非来源于坏块表自身逻辑，但坏块表可能遭到破坏性的影响，所以对其作针对性的边界问题保护机制。

针对 BBT 框架：



增加以下保护：

- 增加 hash 值，校验坏块表信息
- format_bbt 扫描 flash 全盘坏块信息时，对于识别为坏块的可疑坏增加二次校验机制
- update_bbt 更新坏块表时，写坏块表增加回读校验机制，校验失败擦除冗余信息

优化方案注错

uboot/kernel 阶段注入以下错误：

- 模拟上电扫描坏块表时异常
- 模拟 bbt 低格过程扫描 flash 全盘坏块信息异常
- 模拟 bbt write 过程写入异常、hash 值异常、回读异常、回读 hash 值异常

其他说明：

- 由于 spl 阶段坏块表不支持低格，所以不存在破坏性动作，故不作注入错误测试
- 拷机压测后坏块表保持正常无错误坏块信息

优化方案补丁

uboot:

```
commit 55ef8700b677919efa06f3bfe1dcab3a4554a92d
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Sun Sep 24 20:11:20 2023 +0800

    mtd: nand: bbt: Add mechanisms to protect bad block tables

    1.Add hash check
    2.Add anti-shake mechanism to avoid damaging the bad block tables
    3.Add anti-shake mechanism to optimize the reliability of bad block table

    Change-Id: I6ac9554ae0f642d6a89d21a8f79a2a87d607f993
    Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
```

kernel:

```
commit cc1f433fb9ade372b06e9d62651ed8d719aaac94
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Sun Sep 24 20:24:00 2023 +0800

    drivers: rkflash: Add mechanisms to protect bad block tables

    1.Add hash check
    2.Add anti-shake mechanism to avoid damaging the bad block tables
    3.Add anti-shake mechanism to optimize the reliability of bad block table

    Change-Id: I888ceba54e5bbc55283850316e27560c484a9cf5
    Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
```

8.6 SPI Nand 烧录器烧录支持母片不支持一次性导出全部镜像

建议使用 RK programmer_image_tool 工具制作烧录器镜像，而非通过母片方式制作烧录镜像。不支持一次性导出 SPI Nand 镜像作为母片镜像的原因是因为 linux 的 mtd 框架\ubifs 等方案对坏块的管理是按照分区来探测的。

例如：
分区定义：8MB a 分区（中间有一个坏块 128KB）| 8MB b 分区

然后别烧录： 7MB A 镜像 和 7MB B镜像，

假定制作为 **7MB_A + 1MB 冗余 + 7MB_B 的 total 镜像**，烧录下去，那么在 skip bad block 后：

A 镜像有效部分起始地址，0

B 镜像有效部分起始地址，8MB + 128KB

但是软件定义的分区是 8MB 起始，那么他就会在 8MB flash offset 的起始地址开始加载数据，这样镜像就错位了。

真正做法：

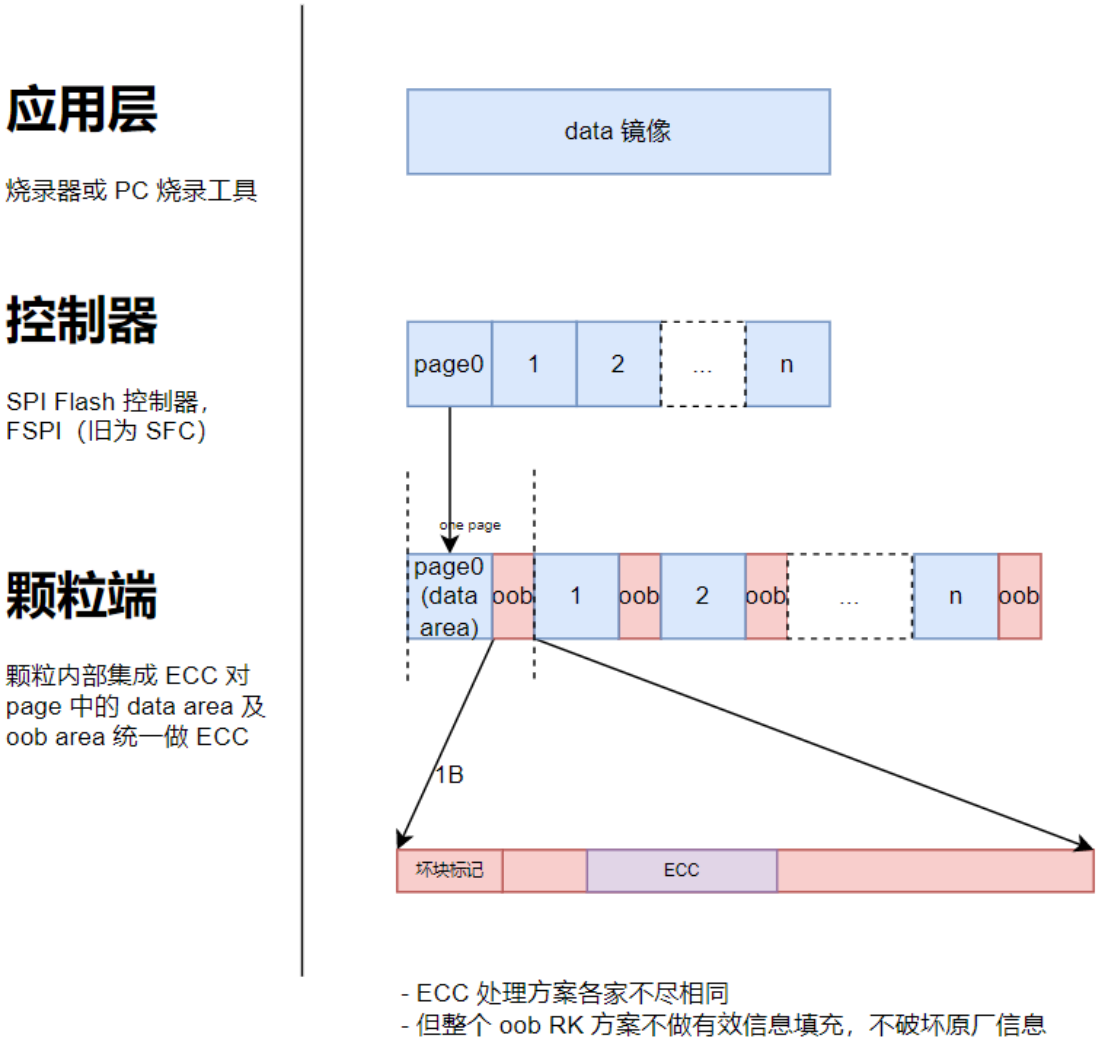
0 地址烧录 A 分立镜像，8MB 地址烧录 B 分立镜像，也就是从分区起始 write skip bad blcok 烧录镜像

软件各自分区探测分区镜像的策略，从分区起始 read skip bad block加载数据

烧录策略和软件策略达到一致。

8.7 SPI Nand SDK 输出镜像为有效数据（不包含冗余 oob 数据）

主要原因：SPI Flash 控制器没有集成 ECC，所以只能依赖颗粒端的 ECC 模块。



8.8 SPI Nand rkflash 方案内核阶段读写速率优化

IO 有较大的数据量时，实际上确实是会有较高的 CPU 占用率，但是可以通过牺牲一些 flash 速率来改善 CPU 调度。

```
diff --git a/drivers/rkflash/sfc_nand.c b/drivers/rkflash/sfc_nand.c
index 0bc1c6b43fd7..ca190c115ad3 100644
--- a/drivers/rkflash/sfc_nand.c
+++ b/drivers/rkflash/sfc_nand.c
@@ -274,7 +274,7 @@ static int sfc_nand_wait_busy(u8 *data, int timeout)

        *data = 0;

-       for (i = 0; i < timeout; i++) {
+       for (i = 0; i < timeout; i+= 20) {
                ret = sfc_nand_read_feature(0xC0, &status);

                if (ret != SFC_OK)
@@ -285,7 +285,7 @@ static int sfc_nand_wait_busy(u8 *data, int timeout)
                if (!(status & (1 << 0)))
                        return SFC_OK;

-               sfc_delay(1);
```

```
+                usleep_range(20, 30);
        }

        return SFC_NAND_WAIT_TIME_OUT;
```

8.9 SPI Nand mtd 方案内核阶段 flash 信息及坏块信息

坏块信息：

```
cat /sys/devices/virtual/mtd/mtd5/bad_blocks
```

其他信息：

```
# ls /sys/devices/virtual/mtd/mtd4/
bad_blocks          ecc_step_size       numeraseregions     subsystem/
bbt_blocks          ecc_strength        offset              type
bitflip_threshold   erasesize           oobsize             uevent
corrected_bits      flags               power/              writesize
dev                 mtdblock4/          size
ecc_failures        name                subpagesize
```

8.10 SPI Nand mtd 方案内核阶段 mtd write fail

确认是否为以下颗粒：

- S35ML01G3, ANV1GCP0CLG, HYF1GQ4UTXCAE, YX25G1E
- S35ML02G3, ANV2GCP0CLG, HYF2GQ4UTXCAE, YX25G2E

该颗粒对于 84h 命令支持的兼容性问题，已添加补丁支持，且申请原厂更正文档说明。

```
commit 328145662f6d6154fbf4329a0d53f9c152673648
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Fri Jul 7 12:06:40 2023 +0800

    mtd: spinand: skyhigh: The vendor requires the devices to be patched

    1.Double OIP=0 after page 13H
    2.The nand flash does not support 84H and 34H command

    Change-Id: Ie805f42a36e1a864115988087bdc43592cc94ded
    Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
```

8.11 SPI Nand mtd 方案根文件系统自动挂载兼容 EMMC 方案

默认 SDK 不支持，如果一定要兼容 EMMC，建议是 uboot 下识别颗粒类型是 EMMC 还是 spi-nand 后，将 root 的参数传递给 kernel 的 bootargs。

8.12 SPI Nand mtd 方案 env 分区表升级方式兼容 4K pagesize 颗粒 idb 镜像

4KB pagesize 颗粒需使用特殊命令生成 idblock 镜像用以升级，命令参考：

```
./tools/programmer_image_tool -i rv1106_download_v1.14.108.bin -b 128 -p 4 -2 -t
spinand -o ./
```

说明：

- 输出 idblock.img 目标镜像
- 最新的 SDK 默认输出该镜像

8.13 SPI Nand 快启方案内核未建立坏块表

SPI Nand page read 速率异常与优化，spl 快启方案没有经过 uboot，内核阶段的坏块表建立存在兼容问题没有建立

```
commit c7270465b2a699211dfd8ad3c2f882f22247c40c
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Thu Dec 17 16:05:27 2020 +0800

    drivers: rkflash: Fixed bbt operation calculation error

    Fixes: 008340a82 ("drivers: rkflash: Change to use the api which the oob area
available")
    Change-Id: I140aa76a2acb73271ba04b7060030dc06a2353e6
    Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
```

8.14 SPI Nand 客户自定义 MTD 升级方案升级带坏块分区存在异常

问题来源：

通常为存在分区内随机地址写入行为，例如：2M左右大小的镜像，以512K为单位用nandwrite分次写入，每写一次下一次nandwrite偏移512K，如果此时中间某次碰到有坏块，就会导致uboot这边mtd read失败。

建议：

要求一次性写入，否则就要自行做坏块管理。

为何要求一次性写入：

如果不对齐分区头一次性写入会存在无法有效识别坏块的可能，假定 bad block 8：

- write 7_8，实际 write 7_9
- write 9，实际 write9 这样 9 就会重复 write，导致 9 里的数据回读 ECC fail。

8.15 SPI Nand 裸分区用户读写行为应关注分区寿命

SPI Nand 颗粒有擦写次数，当磨损达到一定程度后，分区数据保持能力会下降，最终导致数据错误，所以分区应根据实际需求评估寿命。

使用天数 = PE_cycles * Copies / PE_each_data

假定：

- 通常颗粒擦写寿命为 100K 次，以实际手册为准，PE_cycles = 100000
- 裸分区数据如果实现双备份，Copies = 2
- 每天擦除 100 次擦写裸分区 PE_each_data = 100

则使用天数 = 100000 * 2 / 100 = 2000 天。

9. SPI Nor 问题

9.1 SPI Nor 颗粒 1V8 256Mbits 及以上容量物料选型

MX25U25645GZ4I-00 256MBits 32MB

MX25U51245GZ4I00 512Mbits 64MB

GD25LQ256CW1G 256MBits 32MB

W25Q256JWEQ 256MBits 32MB

芯天下 XT25W512BSFHGU 512Mbit 1.65~3.6V 未验证

dosilicon DS25M4BA-1AIB1 256Mb 未验证

dosilicon DS25M4BA-1AIB4 256Mb 未验证

可以和这些厂家交流是否有符合客户需求的颗粒，包括 RK 未验证的颗粒可以让提供样片验证，已支持的颗粒列表以最新 flash 支持列表为准。

9.2 SPI Nor rkflash 方案与内核 4.4 及更早版本的 mtd 开源框架对比

旧版本内核开源框架支持颗粒较有限，且 mtd 框架结构较老，所以 RK 未做适配。相比之下，rkflash 存储方案实现并兼容较多 spiflash 颗粒。

9.3 SPI Nor mtd 方案 u-boot 下 sf 命令及相关接口实现

RK SDK u-boot 使用的是 DM_SPI_FLASH 框架，支持 spi_flash.h 下的：

- spi_find_bus_and_cs 和 spi_flash_probe_bus_cs 的初始化接口
- spi_flash_read 等读写 接口

如何使用相关接口，建议参考 cmd/sf.c 源码及 u-boot 命令

cmd_sf

以 RV1106 为例：

补丁：

```
diff --git a/arch/arm/dts/rv1106-u-boot.dtsi b/arch/arm/dts/rv1106-u-boot.dtsi
index b0632e9ce9..03bc933125 100644
--- a/arch/arm/dts/rv1106-u-boot.dtsi
+++ b/arch/arm/dts/rv1106-u-boot.dtsi
@@ -167,6 +167,7 @@
                spi-tx-bus-width = <1>;
                spi-rx-bus-width = <4>;
                spi-max-frequency = <80000000>;
+               status = "disabled";
        };

        spi_nor: flash@1 {
diff --git a/configs/rv1106_defconfig b/configs/rv1106_defconfig
index 9b20bcfb65..b1f3933a11 100644
--- a/configs/rv1106_defconfig
+++ b/configs/rv1106_defconfig
@@ -148,3 +148,4 @@ CONFIG_SPL_LZMA=y
 CONFIG_SPL_GZIP=y
 CONFIG_ERRNO_STR=y
 # CONFIG_EFI_LOADER is not set
+CONFIG_CMD_SF=y
```

命令：

```
=> sf probe 2:0 10000000 3
SF: Detected sfc_nor with page size 256 Bytes, erase size 4 KiB, total 16 MiB

=> sf read 0x400000 0x200 0x200
device 0 offset 0x200, size 0x200
SF: 512 bytes @ 0x200 Read: OK
```

9.4 SPI Nor mtd 方案注册 mtdparts 耗时长

每一个 mtd partition 都是一个 mtd 设备，如果开启 mtd block 同时还要注册 block 设备，整体注册时间就是比较长：

add_mtd_device:

- device_register 0.15ms
- mtd_nvmem_add 0.08ms
- device_create 0.12ms
- mtdpstore_notify_add 0.5ms

9.5 SPI Nor mtd 方案 uboot 擦写速率慢

uboot 环境如不适用 JFFS2 文件系统，可以去掉 SPI_FLASH_USE_4K_SECTORS 宏，改用 64KB 对齐擦除。

9.6 SPI Nor 单线和四线传输性能差异

测试环境：IO 时钟 80MHz。

IO 使用	io0~3 双向	io0~1 双向	io0 收，io1 发
文件系统读	35MB/s	19MB/s	11MB/s
文件系统擦除写	150KB/s	150KB/s	150KB/s

注意：SPI flash 单线实际上并非指硬件上用于传输的 IO 只有一个，而是指一次单向传输。

9.7 SPI Nor 提高速率拷机异常

以标称 104 MHz 某颗粒为例

- 颗粒标称是 104MHz，但是有小部分特性约束在 83MHz
- 对该颗粒进行测试，运行80Mhz条件下不稳定

测试目标

- 80 MHz IO 速率下拷机通过

信号质量确认

建议先确认信号质量：

- 测试内核阶段 SPI Nor 运行过程中的关键信号，包括 clk、d0、d1、cs，最好同时抓取来确认内核阶段信号质量是否有明显不足的点

测试

信号没有明显问题，建议：

- 测试A：问题颗粒降频到 60MHz，做一些高低温的 auto reboot 测试，看是否压力下能 pass
- 测试B：选择另一个原厂的标称同样为 100MHz，做 auto reboot 测试，作为对照，来作证是否板子自身存在质量问题
- 根据测试结果，确定 60M 是否稳定：
    - 如果 AB 都 fail，很大概率是板子的硬件存在问题，导致 spinor 速率上不去，进一步确认信号
    - 如果 AB 都 pass，综合默认固件 80MHz fail 的情形来看，较大概率是问题颗粒虚标
    - 如果 A fail B pass，则继续降低 A 的频率到 40M，继续测试，如果 40MHz pass，则 A 虚标很严重，或者是兼容性极差

根据测试结果做进一步处理

- 如果测试相对良好，且确定要用这个颗粒，再评估 60MHz 是否满足产品性能要求

10. 工具问题

10.1 upgrade_tool 工具单独制作 GPT 镜像

使用 linux upgrade tool，版本 v1.67及以上

```
./upgrade_tool gpt parameter.txt  gpt.img
```

10.2 boot_merger 解包 loader

```
./tools/boot_merger unpack -i rk356x_spl_loader_v1.01.102-528M-1115.bin -o .
```

10.3 安卓升级工具 LoaderToDDR 功能

工具上的 loader 项有两种配置方式：

1. loader —— 引导升级的功能，并且从 loader 镜像中解析出 idb.img 升级
2. loaderToDDR —— 仅有引导升级的功能

10.4 idb_bootconfig 工具介绍

romcode 默认以单线传输方式加载 SPI Nor 里面的固件，支持 idb 打包镜像头中添加 bootconfig 来提速修改为四线加载固件。
idb_bootconfig 脚本就是在 idb 镜像中添加 nor flash 启动加速配置的脚本。

由于 spinor bootconfig 参数不同的 spinor flash 配置不同，所以相较于预制作镜像填充数据，更建议使用 idb_bootconfig 中添加，否则就需要为颗粒的每一个 spinor 选型制定对应的 bootconfig 参数。