DDR 开发指南

文件标识：RK-KF-YF-39

发布版本：V2.7.0

日期：2025-02-12

文件密级：□绝密 □秘密 □内部资料 ■公开

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

适用于所有平台的开发指南

概述

产品版本

芯片名称	内核版本
所有芯片	所有内核版本

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	何灿阳	2017-12-21	初始版本
V1.1.0	何灿阳	2018-03-30	增加对 kernel 4.4 DDR 频率相关的描述
V1.2.0	何智欢	2019-01-29	增加调整 loader 的 de-skew 说明
V1.3.0	汤云平	2021-01-21	增加RK356x/RV1109/RV1126说明
V1.4.0	汤云平	2021-07-09	修改RK356x ECC说明
V1.5.0	何智欢	2022-05-06	增加RK3326S/PX30S说明
V1.6.0	何灿阳	2022-09-28	增加"修改DDR容量"的说明
V1.7.0	陈有敏	2022-09-28	增加DMC驱动常见问题说明
V1.8.0	何智欢	2023-05-25	增加“如何查看DDR厂商ID”的说明
V1.9.0	何智欢	2023-05-31	增加LPDDR5 厂商ID 对照表
V2.0.0	陈有敏	2023-08-11	合并其他DDR开发相关说明文档
V2.1.0	何智欢	2023-08-29	增加kernel里获取DDR ECC信息的说明
V2.1.1	姚旭伟	2023-10-12	优化格式
V2.2.0	何智欢	2023-11-01	修改HAL里DDR ECC的描述
V2.3.0	何灿阳	2023-10-31	更新"如何看懂 DDR 打印信息" 更新"如何将RK给的 DDR bin 合成成完整可用的 loader" 增加"DDR bin特殊说明" 增加"修改DDR bin文件" 更新"如何修改 U-Boot 中的 DDR 频率" 更新"如何 enable/disable kernel 中的 DDR 变频功能" 更新"如何让 kernel 一次 DDR 变频都不运行" 更新"如何查看 DDR 的容量" 增加"查看DDR频率" 更新"如何修改 DDR 频率" 更新"如何修改 DDR 某个频率对应的电压" 更新"DDR 如何定频" 更新"如何查看 DDR 带宽利用率" 更新"如何测试 DDR 可靠性" 更新"如何确定 DDR 能运行的最高频率" 增加U-Boot下运行DDR压力测试 更新"如何查看DDR厂商ID" 增加"常见信号相关问题" 增加"高温刷新率（refresh rate）问题" 调整"Rockchip DDR DQ眼图工具指南" 更新"Rockchip DDR 带宽工具使用说明"
V2.4.0	何灿阳	2024-05-17	开发指南增加RK3576的描述及修改一些格式
V2.5.0	何智欢	2024-06-07	开发指南增加kernel里DDR ECC错误注入说明
V2.6.0	何灿阳	2024-09-30	开发指南增加关闭self-refresh、power-down的说明

版本号	作者	修改日期	修改说明
V2.6.1	何灿阳	2024-11-04	优化格式
V2.7.0	姚旭伟	2025-02-12	增加“Rockchip DDR_UserTool（DDR 焊接检测工具）指南”

目录

1. Chapter-1 DDR 开发指南-FAQ

1.1 如何看懂 DDR 打印信息

DDR 打印信息包括 loader 中的打印和 kernel 中的打印，loader 中打印的解析如下：

```
DDR Version 1.05 20170712// DDR 初始化代码的版本信息，用于核对版本。从这行开始，已经进入DDR初始化代码
In
SRX // 有SRX，说明是热重启；没有SRX，说明是冷开机。有的芯片没加这个功能，一直都是没有SRX的
Channel a: DDR3 400MHz // 下面都是DDR容量的详细信息，具体解析可以看"如何查看DDR的容量"章节
Bus Width=32 Col=10 Bank=8 Row=15 CS=1 Die Bus-Width=16 Size=1024MB
Channel b: DDR3 400MHz
Bus Width=32 Col=10 Bank=8 Row=15 CS=1 Die Bus-Width=16 Size=1024MB
Memory OK // DDR自测的结果，第一个Memroy OK是Channel a的自测结果
Memory OK // 是Channel b的自测结果。有报错，说明焊接有问题；没报错，说明当前自测没问题，整个DDR是否能稳定工作，还得看后续阶段的运行结果。
OUT // 这行打印之后，就退出了DDR初始化代码
```

如下是 kernel 3.0 和 kernel 3.10 中打印的 DDR 信息

```
[    0.528564] DDR DEBUG: version 1.00 20150126 //版本信息
[    0.528690] DDR DEBUG: Channel a:    //DDR容量详细信息
[    0.528701] DDR DEBUG: DDR3 Device
[    0.528716] DDR DEBUG: Bus Width=32 Col=10 Bank=8 Row=15 CS=1 Total
Capability=1024MB
[    0.528727] DDR DEBUG: Channel b:
[    0.528736] DDR DEBUG: DDR3 Device
[    0.528750] DDR DEBUG: Bus Width=32 Col=10 Bank=8 Row=15 CS=1 Total
Capability=1024MB
//后面还有其他DDR打印信息，可以不用管，是写DDR驱动人员用的
//DDR DEBUG的字样打印结束，就是kernel中DDR初始化完成了
```

kernel 3.10 还会有如下打印，是 DDR 变频模块的输出信息

```
[    1.473637] ddrfreq: verion 1.2 20140526 //DDR变频模块版本
[    1.473653] ddrfreq: normal 396MHz video_1080p 240MHz video_4k 396MHz
dualview 396MHz idle 0MHz suspend 200MHz reboot 396MHz //DDR各个场景对应的频率，是从dts表格中读取出来的
[    1.473661] ddrfreq: auto-freq=1 //负载变频功能是否开启，1表示开启，0表示关闭
[    1.473667] ddrfreq: auto-freq-table[0] 240MHz    //负载变频的频率表格
[    1.473673] ddrfreq: auto-freq-table[1] 324MHz
[    1.473678] ddrfreq: auto-freq-table[2] 396MHz
[    1.473683] ddrfreq: auto-freq-table[3] 528MHz
//如果在这段打印过程中系统卡死了，很可能是DDR变频有问题
```

kernel 3.10后的版本，kernel中不再有DDR容量信息的打印

1.2 如何将RK给的 DDR bin 合成成完整可用的 loader

1. 将 DDR bin 放在 U-Boot 工程的 rk\rkbin\bin\对应目录下
2. 删除原有的 DDR bin 文件
3. 将新的 DDR bin 改名为删除掉的名字
4. 编译 U-Boot（详见《Rockchip-Developer-Guide-UBoot-nextdev.pdf》），就会生成对应的 loader 文件

==5. 根据 DDR bin 打印的串口信息，确认 loader 已经更新正确==

各平台 DDR bin 对应目录整理如下：

芯片平台	路径	Note
PX30	`rk\rkbin\bin\rk33\px30_ddr_333MHz_vX.XX.bin`	
PX3SE	`rk\rkbin\bin\rk31\px3se_ddr_300MHz_vX.XX_uartX.bin`	
RK1808	`rk\rkbin\bin\rk1x\rk1808_ddr_XXXMHz_vX.XX.bin`	
RK3036	`rk\rkbin\bin\rk30\rk3036_ddr_XXXMHz_vX.XX.bin`	1
RK3126、RK3126B、RK3126C	`rk\rkbin\bin\rk31\rk3126_ddr3_300MHz_vX.XX.bin`	
RK3128	`rk\rkbin\bin\rk31\rk3128_ddr_300MHz_vX.XX.bin`	
RK3288	`rk\rkbin\bin\rk32\rk3288_ddr_400MHz_vX.XX.bin`	
RK322x	`rk\rkbin\bin\rk32\rk322x_ddr_XXXMHz_vX.XX.bin`	
RK322xh	`rk\rkbin\bin\rk33\rk322xh_ddr_333MHz_vX.XX.bin`	
RK3308	`rk\rkbin\bin\rk33\rk3308_ddr_XXXMHz_uartX_mX_vX.XX.bin`	
RK3326	`rk\rkbin\bin\rk33\rk3326_ddr_333MHz_vX.XX.bin`	
RK3328	`rk\rkbin\bin\rk33\rk3328_ddr_XXXMHz_vX.XX.bin`	
RK3368	`rk\rkbin\bin\rk33\rk3368_ddr_600MHz_vX.XX.bin`	
RK3399	`rk\rkbin\bin\rk33\rk3399_ddr_XXXMHz_vX.XX.bin`	2
RK3399PRO	`rk\rkbin\bin\rk33\rk3399pro_ddr_XXXMHz_vX.XX.bin`	
RK3528	`rk\rkbin\bin\rk35\rk3528_ddr_XXXMHz_vX.XX.bin`	3
RK3562	`rk\rkbin\bin\rk35\rk3562_ddr_XXXMHz_vX.XX.bin`	
RK3566	`rk\rkbin\bin\rk35\rk3566_ddr_XXXMHz_vX.XX.bin`	
RK3568	`rk\rkbin\bin\rk35\rk3568_ddr_XXXMHz_vX.XX.bin`	
RK3576	`rk\rkbin\bin\rk35\rk3576_ddr_lp4_XXXXMHz_lp5_XXXXMHz_vX.XX.bin`	
RK3588	`rk\rkbin\bin\rk35\rk3588_ddr_lp4_XXXXMHz_lp5_XXXXMHz_vX.XX.bin`	
RV1106	`rk\rkbin\bin\rv11\rv1106_ddr_XXXMHz_vX.XX.bin`	
RV1108	`rk\rkbin\bin\rv11\rv1108_ddr_vX.XX.bin`	
RV1126	`rk\rkbin\bin\rv11\rv1126_ddr_XXXMHz_vX.XX.bin`	
后续新平台	都是按照类似的命名规则放在rk\rkbin\bin\目录下，可以自行查找	

Note 1：具体用哪个频率由 `rk\rkbin\RKBOOT\RK3036_ECHOMINIALL.ini` 或 RK3036MINIALL.ini 文件中指定。其中 RK3036_ECHOMINIALL.ini 是 ECHO 产品使用的，其他产品都使用 RK3036MINIALL.ini。至于哪个是 ECHO 产品，请联系产品部的人。

Note 2：具体用哪个频率由 `rk\rkbin\RKBOOT\RK3399MINIALL.ini` 文件中指定

Note 3：用于PCB非2层、DDR非4BIT的RK3528硬件设计。2层PCB或4BIT的DDR，参考后面的“DDR bin特殊说明”

1.3 DDR bin包含4个频率点的平台

如下平台的DDR bin文件中，都包含有4个DDR频率，其中最高的那个频率点在DDR bin名称中体现，如 `rv1126_ddr_924MHz_v1.05.bin`，最高频点是924M。

kernel中只能使用这4个频率。如果想修改4个频率，见"修改DDR bin文件"章节

一般默认4个频率如下，

芯片平台	文件	包含频率(MHz)
RK3528	`rk3528_ddr_XXXMHz_vX.XX.bin`	324，528，780，及文件名中的频率
RK3562	`rk3562_ddr_XXXMHz_vX.XX.bin`	324，528，780，及文件名中的频率
RK3566	`rk3566_ddr_XXXMHz_vX.XX.bin`	324，528，780，及文件名中的频率
RK3568	`rk3568_ddr_XXXMHz_vX.XX.bin`	324，528，780，及文件名中的频率

芯片平台	文件	包含频率(MHz)
RK3576	`rk3576_ddr_lp4_XXXXMHz_lp5_XXXXMHz_vX.XX.bin`	LP4/LP4X：528，1068，1560，及文件名中的频率 LP5/LP5X：534，1320，1968，及文件名中的频率
RK3588	`rk3588_ddr_lp4_XXXXMHz_lp5_XXXXMHz_vX.XX.bin`	LP4/LP4X：528，1068，1560，及文件名中的频率 LP5/LP5X：534，1320，1968，及文件名中的频率
RV1126	`rv1126_ddr_XXXMHz_vX.XX.bin`	328，528，784，及文件名中的频率

可以从loader的串口log中查看到这4个频率点，如下,

```
DDR ... v1.14
LPDDR4X, 2112MHz
channel[0] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
channel[1] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
channel[2] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
channel[3] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
...
change to F1: 528MHz  /* 4个DDR频率 */
change to F2: 1068MHz
change to F3: 1560MHz
change to F0: 2112MHz
out
```

也可以通过`rkbin/tools/ddrbin_tool`工具，读取出当前的4个频率，如

`./ddrbin_tool px30 -g gen_param.txt px30_ddr_333MHz_v1.15.bin`

具体使用见ddrbin_tool使用说明

1.4 DDR bin特殊说明

对rk\rkbin\bin下的DDR bin文件，有些存在特殊的地方，需要说明

- RK3399

由于RK3399不同DDR类型支持的频率不同，而DDR bin名字没有按类型来说明频率，导致一些客户出现混乱，下面为统一说明

文件	**DDR3/LP3 频率**		**LP4 频率**	
	退出loader的频率(MHz)	kernel能支持的频率(MHz)	退出loader的频率(MHz)	kernel能支持的频率(MHz)
`rk33\rk3399_ddr_666MHz_v1.30.bin`	666	<=666	416	328、416、666、856
`rk33\rk3399_ddr_800MHz_v1.30.bin`	800	<=800	856	328、416、666、856
`rk33\rk3399_ddr_933MHz_v1.30.bin`	933	<=933	856	328、416、666、856、933

- eyescan后缀的bin

以eyescan后缀的DDR bin，用于合成能获取DDR信号二维眼图的Loader，如

`rk\rkbin\bin\rk35\rk3566_ddr_1056MHz_eyescan_v1.16.bin`

`rk\rkbin\bin\rk35\rk3588_ddr_lp4_2112MHz_lp5_2736MHz_eyescan_v1.11.bin`

`rk\rkbin\bin\rk35\rk3562_ddr_1332MHz_eyescan_v1.04.bin`

`rk\rkbin\bin\rk35\rk3568_ddr_1560MHz_eyescan_v1.16.bin`

- ultra后缀的bin

用于电子书的超级待机，搭配电子书的硬件使用，如

`rk\rkbin\bin\rk35\rk3566_ddr_XXXMHz_ultra_v1.10.bin`

`rk\rkbin\bin\rk35\rk3562_ddr_XXXMHz_ultra_v1.05.bin`

- tb后缀的bin

用于快速开机，使用时必须选择对应DDR类型的bin，如

`rk\rkbin\bin\rv11\rv1126_tpl_XXXMHz_ddr4_tb_v1.08.bin`

`rk\rkbin\bin\rv11\rv1126_tpl_XXXMHz_ddr3_tb_v1.08.bin`

`rk\rkbin\bin\rv11\rv1126_tpl_XXXMHz_lp3_tb_v1.08.bin`

`rk\rkbin\bin\rv11\rv1126_tpl_XXXMHz_lp4_tb_v1.08.bin`

用于RV1106快速开机，用于DDR3

`rk\rkbin\bin\rv11\rv1106_ddr_924MHz_tb_v1.13.bin`

- 3528搭配PCB使用

用于4BIT的DDR设计

`rk\rkbin\bin\rk35\rk3528_ddr_1056MHz_4BIT_PCB_v1.07.bin`

用于2层的PCB设计

`rk\rkbin\bin\rk35\rk3528_ddr_1056MHz_2L_PCB_v1.07.bin`

1.5 修改DDR bin文件

上面章节提到的DDR bin文件，用于开机时，初始化DDR。通过RK提供的工具，可以修改DDR bin文件，达到修改DDR初始化参数、DDR频率、关闭串口、串口波特率等功能。

- rk_ddrBin_tool_windows

优先推荐使用rk_ddrBin_tool_windows，其是有界面的工具，容易上手。

工具在：

https://redmine.rock-chips.com/documents/49 -> rk_ddrBin_tool_windows_Vx.xx.7z

一般有几个压缩包，需要全部下载，再解压。

工具的Help下，有使用指南。

- `rkbin/tools/ddrbin_tool`

此工具是命令行模式，在rkbin工程下，`rkbin/tools/ddrbin_tool`

说明文档是`rkbin/tools/ddrbin_tool_user_guide.txt`

常见修改：

- 改DDR频率

对那些只支持4个DDR频率的SOC，见"DDR bin包含4个频率点的平台"章节

可以按DDR类型，修改对应的4个频率。例如下图是LPDDR4的4个频率

	参数名称	配置值	有效值	Unit	
1	lp4_freq	2112	300-2133	MHz	DDR初始化频率(FSP_0频率)
2	lp4_f1_freq_mhz	528	300-2133	MHz	DDR FSP_1频率，用于变频
3	lp4_f2_freq_mhz	1068	300-2133	MHz	DDR FSP_2频率，用于变频
4	lp4_f3_freq_mhz	1560	300-2133	MHz	DDR FSP_3频率，用于变频

- 改串口号

参数名称	配置值	有效值	Unit	
uart id	0	|	——	串口ID，0xf=关闭串口打印

- 改波特率

参数名称	配置值	有效值	Unit	
uart id	0	|	——	串口ID，0xf=关闭串口打印
uart iomux	0	|	——	串口IOMUX
uart baudrate	1500000	|	bps	串口波特率，支持115200或1500000

1.6 如何修改 U-Boot 中的 DDR 频率

- RK322x

如下修改方法只有 RK322x 支持，修改方法是，修改 kernel-3.10 代码中的 arch/arm/boot/dts/rk322x.dtsi

```
dram: dram {
    compatible = "rockchip,rk322x-dram";
    status = "okay";
    dram_freq = <786000000>;
    rockchip,dram_timing = <&dram_timing>;
};
```

修改其中的，dram_freq 就可以了，这里单位是 Hz，频率可以随便选择。

U-Boot 会去解析这个 dts，然后读取这个频率，变频到对应频率。

- RK3576/RK3588

如下修改方法只有RK3576和RK3588支持。这些平台，在loader初始化DDR时，会一起初始化4个DDR频率，供后续的kernel使用，且默认以最高频退出loader，继续后面的程序。

如下的F1，F2，F3，F0就是这4个频率，默认以F0退出。

```
DDR ... v1.14
LPDDR4X, 2112MHz
channel[0] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
channel[1] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
channel[2] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
channel[3] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
...
change to F1: 528MHz  /* 4个DDR频率 */
change to F2: 1068MHz
change to F3: 1560MHz
change to F0: 2112MHz
out
```

当需要修改U-Boot下的DDR频率，即只要修改退出loader的频率。

通过"修改DDR bin文件"所述工具，找到boot_fsp参数，可以选择以F0/F1/F2/F3的哪个频率作为退出loader的频率。从而实现修改U-Boot中的DDR频率。

boot_fsp	0	|	——	完成DDR初始化后进入系统的DDR频率(F0/F1/F2/F3)。

1.7 如何 enable/disable kernel 中的 DDR 变频功能

先确认该芯片支持 kernel 中的 DDR 变频功能，如果支持，可以通过如下方式 enable 或 disable 变频功能。

- 对于 kernel 4.4及以后版本，需要找到 dts 中最终的 dmc 节点，将 status 改成 disabled 就可以 disable kernel 的 DDR 变频功能。相反的，改成 okay，就会 enable DDR 变频功能。

Note 1：RK3576和RK3588平台，通过dts dmc节点enable DDR变频后，根据实际产品硬件使用的电源方案检查DMC节点下的“center-supply”和“mem-supply”属性是否有正确配置。这些属性的配置值为给SOC端DDR模块和LOGIC部分供电的regulator节点名，

RK3576默认为center-supply = <&vdd_ddr_s0>、mem-supply = <&vdd_logic_s0>

RK3588默认为center-supply = <&vdd_ddr_s0>、mem-supply = <&vdd_log_s0>

如果DMC节点缺失该属性会导致dmc驱动加载失败，kernel log如下。

```
rockchip-dmc dmc: Cannot get the regulator "center"
```

Note 2：其他平台，通过dts dmc节点enable DDR变频后，根据实际产品硬件使用的电源方案检查DMC节点下的“center-supply”属性是否有正确配置。该属性的配置值为给SOC端DDR模块供电的regulator节点名，

RK3399默认为center-supply = <&vdd_center>

PX30/RK3568/RV1126等其他平台默认为center-supply = <&vdd_logic>。

如果DMC节点缺失该属性会导致dmc驱动加载失败，kernel log如下。

```
rockchip-dmc dmc: Cannot get the regulator "center"
```

Note 3：由于早期代码，dmc节点有依赖于dfi节点，如果dfi节点为disabled，也会导致dmc节点无效。所以最好dfi节点的status保持跟dmc一致。

举例如下，RK3399 EVB板的最终dmc节点在 arch/arm64/boot/dts/rockchip/rk3399-evb.dtsi 中，

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

&dmc {
    center-supply = <&vdd_center>; /* 这里需要客户根据实际硬件电路来配置 */
    status = "okay"; /* enable kernel DDR变频 */
    ...
};
```

```
&dfi {
    status = "disabled";
};

&dmc {
    status = "disabled";    /* disable kernel DDR变频 */
    ...
};
```

- 对于 kernel 3.10，需要找到 dts 中最终的 clk_ddr_dvfs_table 节点，将 status 改成 disabled 就可以 disable kernel 的 DDR 变频功能。相反的，改成 okay，就会 enable DDR 变频功能。举例如下，RK3288 SDK 板的最终 clk_ddr_dvfs_table 在 arch/arm/boot/dts/rk3288-tb_8846.dts 中，

```
&clk_ddr_dvfs_table {
    ...
    status="okay";  /* enable kernel DDR变频 */
};
```

```
&clk_ddr_dvfs_table {
    ...
    status="disabled";  /* disable kernel DDR变频 */
};
```

- 对于 kernel 3.0，需要在板级的 borad-**.c 文件中修改 dvfs_ddr_table， 让这个表，只留一个 DDR_FREQ_NORMAL 频率，就不会运行 DDR 变频了。举例如下，RK3066 SDK 板的板级文件在 arch/arm/mach-rk30/board-rk30-sdk.c 中

```
/* 这样的表格enable DDR变频 */
static struct cpufreq_frequency_table dvfs_ddr_table[] = {
    {.frequency = 200 * 1000 + DDR_FREQ_SUSPEND, .index = 1050 * 1000},
    {.frequency = 300 * 1000 + DDR_FREQ_VIDEO, .index = 1050 * 1000},
    {.frequency = 400 * 1000 + DDR_FREQ_NORMAL, .index = 1125 * 1000},
    {.frequency = CPUFREQ_TABLE_END},
};
```

```
/* 这样的表格disable DDR变频 */
static struct cpufreq_frequency_table dvfs_ddr_table[] = {
    //{.frequency = 200 * 1000 + DDR_FREQ_SUSPEND, .index = 1050 * 1000},
    //{.frequency = 300 * 1000 + DDR_FREQ_VIDEO, .index = 1050 * 1000},
    {.frequency = 400 * 1000 + DDR_FREQ_NORMAL, .index = 1125 * 1000},
    {.frequency = CPUFREQ_TABLE_END},
};
```

1.8 如何让 kernel 一次 DDR 变频都不运行

上一个标题所讲的修改只是开启或关闭变频功能，即 DDR 不会在平时系统运行时变频，但有一次例外是不受上面修改控制的，即开机的第一次 ddr_init 时，会运行一次变频，用于更新 DDR timing，让性能更高。如果想让 kernel 中一次 DDR 变频都不运行，包括 ddr_init。除了上一个标题讲的“如何 enable/disable kernel 中的 DDR 变频功能”要做以外。还需要修改代码：

- 对于 kernel 4.4及以后版本

  只要按上一个标题讲的“如何 enable/disable kernel 中的 DDR 变频功能”做，就一次 DDR 变频都不会运行了

- 对于 kernel 3.10

  芯片：RK322x

  代码位置：kernel 中无代码

  修改：把 dram 节点改成 disabled，就可以了

  ```
  dram: dram {
      compatible = "rockchip,rk322x-dram";
      status = "disabled";    /* 修改这里 */
      dram_freq = <786000000>;
      rockchip,dram_timing = <&dram_timing>;
  };
  ```

  芯片：RK3188

  代码位置：arch/arm/mach-rockchip/ddr_rk30.c 的 ddr_init()函数

芯片：RK3288

代码位置：arch/arm/mach-rockchip/ddr_rk32.c 的 ddr_init()函数

芯片：RK3126B、RK3126C 不带 trust.img 固件

代码位置：arch/arm/mach-rockchip/ddr_rk3126b.c 的 ddr_init()函数

芯片：RK3126、RK3128

代码位置：./arch/arm/mach-rockchip/ddr_rk3126.c 的 ddr_init()函数

修改：注释掉 ddr_init()函数中，如下几行代码

```
if(freq != 0)
    value = clk_set_rate(clk, 1000*1000*freq);
else
    value = clk_set_rate(clk, clk_get_rate(clk));
```

芯片：RV1108

代码位置：arch/arm/mach-rockchip/ddr_rv1108.c 的 ddr_init()函数

修改：注释掉 ddr_init()函数中，如下几行代码

```
if (freq == 0)
    _ddr_change_freq(ddr_freq_current);
else
    _ddr_change_freq(freq);
```

除了上述几个特殊的芯片，剩下芯片，以及 3126B/3126c 带 trust.img 固件的，只需要按上一个标题讲的“如何 enable/disable kernel 中的 DDR 变频功能”做，就一次 DDR 变频都不会运行了。

- 对于 kernel 3.0

芯片	代码位置
RK3066	arch/arm/mach-rk30/ddr.c 的 ddr_init()函数
RK3026、RK3028A	arch/arm/mach-rk2928/ddr.c 的 ddr_init()函数

修改：注释掉 ddr_init()函数中，如下几行代码

```
if(freq != 0)
  value=ddr_change_freq(freq);
else
  value=ddr_change_freq(clk_get_rate(clk_get(NULL, "ddr"))/1000000);
```

1.9 如何查看 DDR 的容量

如果只是简单的想看 DDR 有多大容量，可以用如下命令，查看 MemTotal 容量，这个容量会比 DDR 实际容量小一点点，自己往上取到标准容量就可以了。

```
root@rk3399:/ # cat /proc/meminfo
MemTotal:        3969804 kB
```

如果需要看 DDR 容量的详细信息，按如下步骤：

在 2 个地方有 DDR 容量的打印，loader 中的 DDR 初始化阶段和 kernel 中 DDR 的初始化阶段。kernel 4.4 及以后版本，中全部没有 DDR 容量信息的打印；kernel 3.10 不全有。loader 中的 DDR 详细信息，所有芯片都有。loader 中的 DDR 容量打印，必须用串口才能抓到，如果使用 adb，是抓不到这部分信息的。

具体情况如下：

芯片	**loader**	**kernel 3.0/3.10/ kernel 4.4 and later**
RK3026	有详细信息	有详细信息
RK3028A	有详细信息	有详细信息
RK3036	有详细信息	没有
RK3066	有详细信息	有详细信息
RK3126	有详细信息	有详细信息
RK3126B、RK3126C 带 trust.img 固件	有详细信息	没有
RK3126B、RK3126C 不带 trust.img 固件	有详细信息	有详细信息
RK3128	有详细信息	有详细信息
RK3188	有详细信息	有详细信息
RK322x	有详细信息	没有
RK322xh	有详细信息	没有
RK3288	有详细信息	有详细信息
RK3328	有详细信息	没有
RK3368	有详细信息	没有
RK3399	有详细信息	没有
RV1108	有详细信息	没有
Other SOC	有详细信息	没有

DDR 的详细信息，会包含有：DDR 类型、DDR 频率、通道信息(Channel a\Channel b)、总线数据位宽(bus width/BW)、行数量(row)、列数量(col/column)、bank 数量(bank/BK)、片选数量(CS)、单颗颗粒的数据位宽(Die Bus width/Die BW)、单个通道的 DDR 总容量(size/Total Capability)。

如果需要整机的总容量，芯片只有一个DDR通道的，整机容量就等于Size或Total Capability。芯片有2个通道的，整机容量等于2个通道的Size或Total Capability相加。

如下是 loader 中打印的 DDR 容量详细信息

```
DDR Version 1.05 20170712
In
Channel a: DDR3 400MHz
Bus Width=32 Col=10 Bank=8 Row=15 CS=1 Die Bus-Width=16 Size=1024MB
Channel b: DDR3 400MHz
Bus Width=32 Col=10 Bank=8 Row=15 CS=1 Die Bus-Width=16 Size=1024MB
Memory OK
Memory OK
OUT
```

如下是 kernel 中打印的 DDR 容量详细信息

```
[    0.528564] DDR DEBUG: version 1.00 20150126
[    0.528690] DDR DEBUG: Channel a:
[    0.528701] DDR DEBUG: DDR3 Device
[    0.528716] DDR DEBUG: Bus Width=32 Col=10 Bank=8 Row=15 CS=1 Total
Capability=1024MB
[    0.528727] DDR DEBUG: Channel b:
[    0.528736] DDR DEBUG: DDR3 Device
[    0.528750] DDR DEBUG: Bus Width=32 Col=10 Bank=8 Row=15 CS=1 Total
Capability=1024MB
[    0.528762] DDR DEBUG: addr=0xd40000
```

1.10 修改DDR容量

目前所有RK平台的DDR容量都是自动识别的，不需要客户配置DDR容量。此处提供修改DDR容量的方法，主要用于客户评估性能或评估减少DDR容量的影响。

在《Rockchip_Developer_Guide_UBoot_Nextdev_CN》-> "CH08调试手段" -> "修改DDR容量"，提供了另一种修改方法。

开机时DDR初始化代码会把DDR容量传递给U-Boot，U-Boot会去除一些安全内存后再传递给内核。用户可以在U-Boot阶段修改传递给内核的DDR容量。

代码位置：

```
./arch/arm/mach-rockchip/param.c
```

修改步骤：

1. 加打印，查看当前的DDR容量信息

```
struct memblock *param_parse_ddr_mem(int *out_count)
{
    ...
    for (i = 0, n = 0; i < count; i++, n++) {
        base = t->u.ddr_mem.bank[i];
        size = t->u.ddr_mem.bank[i + count];
        printf("base:0x%llx, size:0x%llx\n", base, size);     //加这行打印

        /* 0~4GB */
        if (base < SZ_4GB) {
            mem[n].base = base;
            mem[n].size = ddr_mem_get_usable_size(base, size);
```

```
    ...
}
```

按上述修改后，重新编译U-Boot，重新烧录，在新的U-Boot开机log中，有如下输出

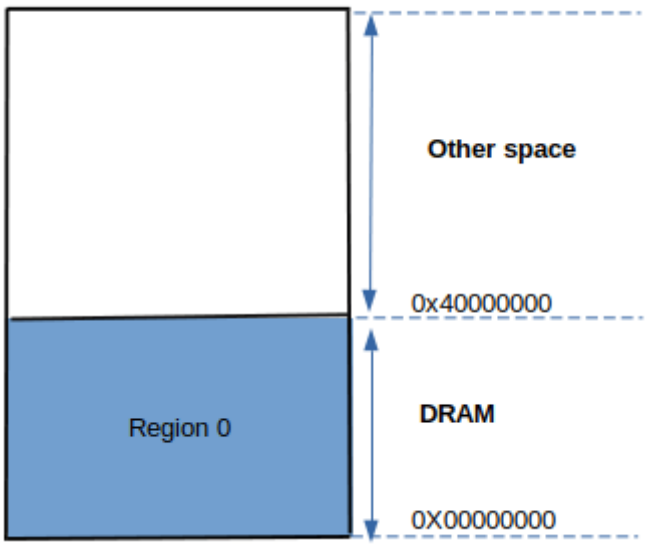
```
U-Boot 2017.09-g0236dc3682-220712-dirty #hcy (Aug 04 2022 - 10:46:07 +0800)

Model: Rockchip RK3568 Evaluation Board
PreSerial: 2, raw, 0xfe660000
DRAM:  base:0x0, size:0x40000000    //这行是我们增加的输出
```

其中

```
base:0x0, size:0x40000000
```

表示如下图，有1块DRAM空间，基地址是0x0，大小是0x40000000



在U-Boot引导kernel前，U-Boot去掉一些安全内存后，传递给内核的DDR容量信息，如下

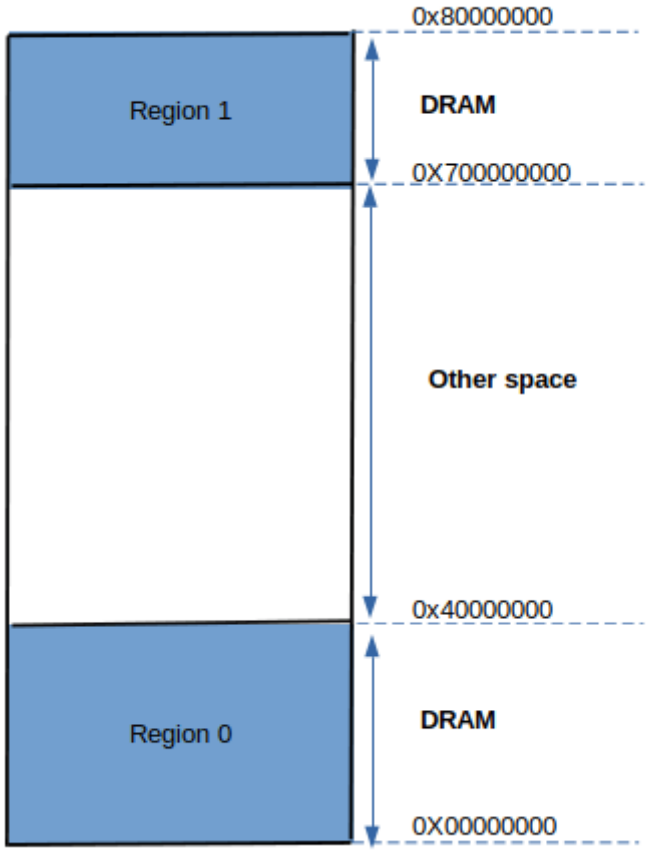
```
Adding bank: 0x00200000 - 0x08400000 (size: 0x08200000)  //去除安全内存块后的
Adding bank: 0x09400000 - 0x40000000 (size: 0x36c00000)  //去除安全内存块后的
Total: 1182.666 ms

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
```

2. 理解了上述DDR容量信息的描述方法，就可以修改

假设，系统中要增加了一块DRAM空间，如下图Region 1的位置，可以做如下修改




```
struct memblock *param_parse_ddr_mem(int *out_count)
{
    ...
    /* extend top ram size */
    if (t->u.ddr_mem.flags & DDR_MEM_FLG_EXT_TOP)
        gd->ram_top_ext_size = t->u.ddr_mem.data[0];

    //修改DDR容量信息，开始
    count = 2;
    t->u.ddr_mem.count = count;
    t->u.ddr_mem.bank[0] = 0x0;  //Region 0 base address
    t->u.ddr_mem.bank[0 + count] = 0x40000000 - 0x0; //Region 0 size
    t->u.ddr_mem.bank[1] = 0x70000000;  //Region 1 base address
    t->u.ddr_mem.bank[1 + count] = 0x80000000 - 0x70000000;  //Region 1 size
    t->u.ddr_mem.hash = 0;
    //修改DDR容量信息，结束

    /* normal ram size */
    count = t->u.ddr_mem.count;
    mem = calloc(count + MEM_RESV_COUNT, sizeof(*mem));
    if (!mem) {
        printf("Calloc ddr memory failed\n");
        return 0;
    }
    ...
    for (i = 0, n = 0; i < count; i++, n++) {
        base = t->u.ddr_mem.bank[i];
        size = t->u.ddr_mem.bank[i + count];
        printf("base:0x%llx, size:0x%llx\n", base, size);
```

```
        /* 0~4GB */
        if (base < SZ_4GB) {
            mem[n].base = base;
            mem[n].size = ddr_mem_get_usable_size(base, size);
    ...
}
```

重点来分析下如下代码

```
    count = 2;    //表示有2块DRAM空间
    t->u.ddr_mem.count = count;    //必须这么写
    t->u.ddr_mem.bank[0] = 0x0;
    t->u.ddr_mem.bank[0 + count] = 0x40000000 - 0x0; //数组index必须有+count
    //Region base address加上Region size为一组，有多少组，必须跟上面的count一样
    t->u.ddr_mem.bank[1] = 0x70000000;
    t->u.ddr_mem.bank[1 + count] = 0x80000000 - 0x70000000;  //数组index必须有
+count
    t->u.ddr_mem.hash = 0;    //必须这么写
```

重新编译U-Boot，重新烧录，在新的开机log中，看到修改已生效

```
...
U-Boot 2017.09-g0236dc3682-220712-dirty #hcy (Aug 04 2022 - 10:46:07 +0800)

Model: Rockchip RK3568 Evaluation Board
PreSerial: 2, raw, 0xfe660000
DRAM:  base:0x0, size:0x40000000
base:0x70000000, size:0x10000000    //我们新增的
...
Adding bank: 0x00200000 - 0x08400000 (size: 0x08200000)
Adding bank: 0x09400000 - 0x40000000 (size: 0x36c00000)
Adding bank: 0x70000000 - 0x80000000 (size: 0x10000000)  //我们新增的
Total: 1182.666 ms

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
```

如果客户还是只有一块DRAM，但是size，想改大到0x80000000，可以增加如下代码

```
    count = 1;    //表示有1块DRAM空间
    t->u.ddr_mem.count = count;    //必须这么写
    t->u.ddr_mem.bank[0] = 0x0;
    t->u.ddr_mem.bank[0 + count] = 0x80000000 - 0x0; //数组index必须有+count
    t->u.ddr_mem.hash = 0;    //必须这么写
```

==Note：DRAM容量并不能随意改大的，因为没有真正的储存器，系统会异常。一般都是size改小，用于评估。此处举例加Region1，是为了方便理解==

1.11 查看DDR频率

- 通过devfreq

有开启DDR变频功能的，可以通过如下节点获取到当前DDR频率

`cat /sys/class/devfreq/dmc/cur_freq`

```
console:/ # cat /sys/class/devfreq/dmc/cur_freq
780000000
```

- 通过clk_summary

`cat /sys/kernel/debug/clk/clk_summary | grep scmi_clk_ddr`

或

`cat /sys/kernel/debug/clk/clk_summary | grep sclk_ddrc`

或

`cat /sys/kernel/debug/clk/clk_summary | grep sclk_ddr`

具体看kernel版本

```
console:/ # cat /sys/kernel/debug/clk/clk_summary | grep scmi_clk_ddr
scmi_clk_ddr                          0        0        0  2736000000          0
    0  50000 /*2736000000为单位Hz的DDR频率，即2736MHz*/
```

1.12 如何修改 DDR 频率

kernel 中改变 DDR 频率的，有 2 种情况，一种是场景变频，一种是负载变频。对于这 2 种变频策略，kernel 4.4 及以后版本和 kernel 3.10 的实现有些不同。

kernel 4.4及以后版本：

场景变频指：进入指定场景，如果此时负载变频功能关闭，则 DDR 频率就变到对应 SYS_STATUS_XXX 定义的频率。如果此时负载变频功能开启的，则 SYS_STATUS_XXX 定义的频率作为最低频率，再根据实际 DDR 利用率状况结合 upthreshold、downdifferential 定义来提频或降频，但是最低频率始终不会低于 SYS_STATUS_XXX 定义的频率。

负载变频指：所有场景都会根据负载来变频。但是会保证频率不低于 SYS_STATUS_XXX 场景定义的频率。唯一特例的是 SYS_STATUS_NORMAL，如果负载变频功能开启，SYS_STATUS_NORMAL 完全被负载变频替换，连最低频率都是由 auto-min-freq 决定，而不是 SYS_STATUS_NORMAL 决定。

kernel 3.10：

场景变频指：进入指定场景，DDR 频率就变到对应 SYS_STATUS_XXX 定义的频率，不在变化，即使此时负载变频功能是打开的，也不会根据负载来变频。

负载变频指：仅仅是用来替换 SYS_STATUS_NORMAL 场景的。即只有在 SYS_STATUS_NORMAL 场景下，DDR 频率才会根据负载情况来变频。

要修改 DDR 频率，还是得按分支和芯片型号来分开处理

- PX30S/RK3326S，见后面的"PX30S/RK3326S DDR频率选择"
- "DDR bin包含4个频率点的平台"章节所列平台，见后面的"DDR bin包含4个频率点的平台，其DDR频率的选择"
- 对于 kernel 4.4，需要找到 dts 中 dmc 节点。举例如下，RK3399 EVB 板的 dmc 节点在 arch/arm64/boot/dts/rockchip/rk3399-evb.dtsi 和 arch/arm64/boot/dts/rockchip/rk3399.dtsi，

```
&dmc {
    status = "okay";
    center-supply = <&vdd_center>;
    upthreshold = <40>;
    downdifferential = <20>;
    system-status-freq = <
        /*system status         freq(KHz)*/
        SYS_STATUS_NORMAL       800000
        SYS_STATUS_REBOOT       528000
        SYS_STATUS_SUSPEND      200000
        SYS_STATUS_VIDEO_1080P  200000
        SYS_STATUS_VIDEO_4K     600000
        SYS_STATUS_VIDEO_4K_10B 800000
        SYS_STATUS_PERFORMANCE  800000
        SYS_STATUS_BOOST        400000
        SYS_STATUS_DUALVIEW     600000
        SYS_STATUS_ISP          600000
    >;
    /* 每一行作为一组数据，其中的min_bw、max_bw表示vop发出的带宽需求，落在min_bw、max_bw范围内，则DDR频率需要再提高freq指定的频率，auto-freq-en=1时有效 */
    vop-bw-dmc-freq = <
    /* min_bw(MB/s) max_bw(MB/s) freq(KHz) */
        0       577      200000
        578     1701     300000
        1702    99999    400000
    >;
    auto-min-freq = <200000>;
};
```

```
dmc: dmc {
    compatible = "rockchip,rk3399-dmc";
    devfreq-events = <&dfi>;
    interrupts = <GIC_SPI 1 IRQ_TYPE_LEVEL_HIGH 0>;
    clocks = <&cru SCLK_DDRCLK>;
    clock-names = "dmc_clk";
    ddr_timing = <&ddr_timing>;
    /* DDR利用率超过40%，开始升频；auto-freq-en=1时才有效 */
    upthreshold = <40>;
    /* DDR利用率低于20%，开始降频；auto-freq-en=1时才有效 */
    downdifferential = <20>;
    system-status-freq = <
    /*system status         freq(KHz)*/
    /* 只有auto-freq-en=0时有效。表示除了下列场景以外的，都用这个场景 */
    SYS_STATUS_NORMAL       800000
    /* reboot前的DDR频率。当auto-freq-en=1时，会以此频率作为min值，根据负载状况往上升频 */
    SYS_STATUS_REBOOT       528000
    /* 一级待机时的DDR频率。当auto-freq-en=1时，会以此频率作为min值，根据负载状况往上升频 */
    SYS_STATUS_SUSPEND      200000
    /* 1080P视频时的DDR频率。当auto-freq-en=1时，会以此频率作为min值，根据负载状况往上升频 */
    SYS_STATUS_VIDEO_1080P  300000
    /* 4K视频时的DDR频率。当auto-freq-en=1时，会以此频率作为min值，根据负载状况往上升频 */
    SYS_STATUS_VIDEO_4K     600000
    /* 4K 10bit视频时的DDR频率。当auto-freq-en=1时，会以此频率作为min值，根据负载状况往上升频 */
```

```
    SYS_STATUS_VIDEO_4K_10B 800000
    /* 性能模式时的DDR频率。当auto-freq-en=1时，会以此频率作为min值，根据负载状况往上升频
*/
    SYS_STATUS_PERFORMANCE  800000
    /* 触屏时的DDR频率，用于从低频提高上来，改善触屏响应。当auto-freq-en=1时，会以此频率作为
min值，根据负载状况往上升频 */
    SYS_STATUS_BOOST        400000
    /* 双屏显示时的DDR频率。当auto-freq-en=1时，会以此频率作为min值，根据负载状况往上升频
*/
    SYS_STATUS_DUALVIEW     600000
    /* ISP时的DDR频率。当auto-freq-en=1时，会以此频率作为min值，根据负载状况往上升频 */
    SYS_STATUS_ISP          600000
    >;
    /* auto-freq-en=1时有效，表示normal场景的最低频率 */
    auto-min-freq = <400000>;
    /* 等于1，表示开启负载变频功能；等于0，表示关闭负载变频功能。开启负载变频功能后，
SYS_STATUS_NORMAL场景将完全被负载负载变频接替，且最低频率以auto-min-freq为准，而不是以
SYS_STATUS_NORMAL定义的为准。而且开启负载变频后，其他场景定义的频率将作为最低频率，系统根据
DDR带宽的利用率状况，依据upthreshold、downdifferential来提高、降低DDR频率 */
    auto-freq-en = <1>;
    status = "disabled";
};
```

==注意==

1. kernel 4.4 的频率电压跟 kernel 3.10 不同，只有频率等于 dmc_opp_table 所列 opp-hz 的，才会按该频率运行；小于 opp-hz，频率会向上取；如果频率超过这个表格的最大 opp-hz，只会按最大 opp-hz 工作。所以，如果不想频率被向上取，或被限制到最大 opp-hz，dmc_opp_table 也是需要关注的。

```
dmc_opp_table: opp-table3 {
    opp-200000000 {
        /* 当DDR频率等于200MHz时，使用这个电压;小于200MHz，就按200MHz运行 */
        opp-hz = /bits/ 64 <200000000>;
        opp-microvolt = <825000>;    //vdd_center电压
    };
    ...
    opp-800000000 {
        opp-hz = /bits/ 64 <800000000>;
        opp-microvolt = <900000>;
    };
};
```

明白了每个配置的含义后，根据需要修改的场景，来修改对应的频率定义，就可以了。如果 auto-freq-en=1，就不好控制频率，这时候降频如果是为了查找定位问题，可以把 auto-freq-en 设置为 0，然后修改各场景定义的频率值来达到目的。

2. RK3399 LPDDR4和RV1126平台，DDR变频支持的频率在loader阶段已确定，ddr初始化阶段会通过串口打印出来，dmc_opp_table定义的频率需要跟其对应。

   以RV1126为例，loader阶段打印的DDR频率为328MHz，528MHz，784MHz，924MHz，则dmc_opp_table也只能定义这4个频率。

```
change to: 328MHz
change to: 528MHz
change to: 784MHz
change to: 924MHz(final freq)
```

```
dmc_opp_table: dmc-opp-table {
    compatible = "operating-points-v2";

    opp-328000000 {
        opp-hz = /bits/ 64 <328000000>;
        opp-microvolt = <800000>;
    };
    opp-528000000 {
        opp-hz = /bits/ 64 <528000000>;
        opp-microvolt = <800000>;
    };
    opp-784000000 {
        opp-hz = /bits/ 64 <784000000>;
        opp-microvolt = <800000>;
    };
    opp-924000000 {
        opp-hz = /bits/ 64 <924000000>;
        opp-microvolt = <800000>;
    };
};
```

如果dmc_opp_table和loader打印的频率不一致，kernel dmc驱动执行DDR变频时会出现频率不匹配问题，log 如下。

```
rockchip-dmc dmc: Get wrong frequency, Request 1056000000, Current 924000000
```

- 对于 kernel 3.10，需要找到 dts 中的 clk_ddr_dvfs_table 节点。举例如下，RK3288 SDK 板的最终的 clk_ddr_dvfs_table 在 arch/arm/boot/dts/rk3288-tb_8846.dts 中，

```
&clk_ddr_dvfs_table {
    /* DDR频率对应的logic电压，如果freq-table或bd-freq-table中的频率比这里的最大频率还大，就无法找到对应电压，因此无法切换到对应频率。这时候需要在这里增加频率电压表格 */
    operating-points = <
        /* KHz    uV */
        200000 1050000
        300000 1050000
        400000 1100000
        533000 1150000
        >;

    freq-table = <
        /*status        freq(KHz)*/
        /* 只有auto-freq-en=0时有效。表示除了下列场景以外的，都用这个场景 */
        SYS_STATUS_NORMAL   400000
        /* 一级待机时的DDR频率 */
        SYS_STATUS_SUSPEND  200000
        /* 1080P视频时的DDR频率 */
        SYS_STATUS_VIDEO_1080P  240000
        /* 4K视频时的DDR频率 */
        SYS_STATUS_VIDEO_4K     400000
```

```
        /* 4K视频60fps时的DDR频率 */
        SYS_STATUS_VIDEO_4K_60FPS     400000
        /* 性能模式时的DDR频率 */
        SYS_STATUS_PERFORMANCE  528000
        /* 双屏显示时的DDR频率 */
        SYS_STATUS_DUALVIEW 400000
        /* 触屏时的DDR频率，用于从低频提高上来，改善触屏响应 */
        SYS_STATUS_BOOST    324000
        /* ISP时的DDR频率 */
        SYS_STATUS_ISP      400000
        >;
    bd-freq-table = <
        /* bandwidth   freq */
        5000           800000
        3500           456000
        2600           396000
        2000           324000
    >;
    /* 负载变频开启后，当处于SYS_STATUS_NORMAL场景时，将按照DDR带宽利用率情况，在这个表格的
几个频率之间切换 */
    auto-freq-table = <
        240000
        324000
        396000
        528000
        >;
    /* 等于1，表示开启负载变频功能；等于0，表示关闭负载变频功能。开启负载变频功能后，
SYS_STATUS_NORMAL场景将完全被负载负载变频接替 */
    auto-freq=<1>;
    /*
     * 0: use standard flow
     * 1: vop dclk never divided
     * 2: vop dclk always divided
     */
    vop-dclk-mode = <0>;
    status="okay";
};
```

明白了每个配置的含义后，根据需要修改的场景，来修改对应的频率定义，就可以了。如果 auto-freq=1，就不好控制频率，这时候降频如果是为了查找定位问题，可以把 auto-freq 设置为 0，然后修改各场景定义的频率值来达到目的。

==注意：修改频率，一定要注意对应的电压，是否能正常工作。==如何修改电压，见"如何修改 DDR 某个频率对应的电压"章节

- 对于 kernel 3.0，需要在板级的 borad-**.c 文件中修改 dvfs_ddr_table。举例如下，RK3066 SDK 板的板级文件在 arch/arm/mach-rk30/board-rk30-sdk.c 中

```
static struct cpufreq_frequency_table dvfs_ddr_table[] = {
    /* 一级待机时的DDR频率 */
    {.frequency = 200 * 1000 + DDR_FREQ_SUSPEND, .index = 1050 * 1000},
    /* 播放视频时的DDR频率 */
    {.frequency = 300 * 1000 + DDR_FREQ_VIDEO, .index = 1050 * 1000},
    /* 除了上述2个场景，其他时候的DDR频率 */
    {.frequency = 400 * 1000 + DDR_FREQ_NORMAL, .index = 1125 * 1000},
    {.frequency = CPUFREQ_TABLE_END},
};
```

kernel 3.0 只有 3 个场景，所要修改的 DDR 频率在.frequency 的 200 * 1000 这里，频率单位是 KHz。“ + DDR_FREQ_SUSPEND”这串可以不用管。

==注意：修改频率，一定要注意对应的电压，是否能正常工作。==如何修改电压，见"如何修改 DDR 某个频率对应的电压"章节

1.12.1 RK3399 LPDDR4支持928MHz的修改方法

1. 修改rkbin目录下的RKBOOT/RK3399MINIALL.ini文件，选择933MHz的DDR bin文件，进行loader打包。

```
diff --git a/RKBOOT/RK3399MINIALL.ini b/RKBOOT/RK3399MINIALL.ini
index d8e71dd7..3e20f255 100755
--- a/RKBOOT/RK3399MINIALL.ini
+++ b/RKBOOT/RK3399MINIALL.ini
@@ -5,7 +5,7 @@ MAJOR=1
 MINOR=26
 [CODE471_OPTION]
 NUM=1
-Path1=bin/rk33/rk3399_ddr_800MHz_v1.27.bin
+Path1=bin/rk33/rk3399_ddr_933MHz_v1.27.bin
 Sleep=1
 [CODE472_OPTION]
 NUM=1
@@ -14,7 +14,7 @@ Path1=bin/rk33/rk3399_usbplug_v1.26.bin
 NUM=2
 LOADER1=FlashData
 LOADER2=FlashBoot
-FlashData=bin/rk33/rk3399_ddr_800MHz_v1.27.bin
+FlashData=bin/rk33/rk3399_ddr_933MHz_v1.27.bin
 FlashBoot=bin/rk33/rk3399_miniloader_v1.26.bin
 [OUTPUT]
 PATH=rk3399_loader_v1.27.126.bin
```

将打包出来的loader文件烧写到机器后，可以通过串口打印的log，看到该loader支持416MHz，856MHz， 328MHz, 666MHz, 928MHz五个频率间变频，当前运行DDR频率为856MHz。

```
DDR Version 1.27 20211018
In
...
support 416 856 328 666 928 MHz, current 856MHz
OUT
```

2. 修改kernel dts dmc_opp_table节点，将opp-928000000表内的“status = "disabled";”改为“status = "okay";”，或者直接删除。

```
                opp-928000000 {
                        opp-hz = /bits/ 64 <928000000>;
                        opp-microvolt = <900000>;
-                       status = "disabled";
                };
```

3. 修改kernel dts dmc节点，将system-status-freq表内的“856000“全部改为”928000“，或者参考本文档《如何修改 DDR 频率》章节关于场景变频的SYS_STATUS_XXX相关描述，根据实际项目不同场景的DDR带宽需求进行选择性修改。

1.12.2 PX30S/RK3326S DDR频率选择

对于RK3326S/PX30S平台，只支持4个频率点，由两个地方共同决定：kernel 的 `arch/arm64/boot/dts/rockchip/px30s-dram-default-timing.dtsi` 对应DDR类型的 ddrx_params 节点和 `arch/arm64/boot/dts/rockchip/px30.dtsi` 的 px30s_dmc_opp_table 节点。

其中ddrx_params节点，对于DDR3，则要配置ddr3_params下的freq_0、freq_1、freq_2、freq_3；对于DDR4，则是ddr4_params下的freq_0、freq_1、freq_2、freq_3；其他DDR类型依次类推。

对于px30s_dmc_opp_table 节点，不区分不同DDR类型，其使能的频率，要对应ddrx_params节点的freq_0、freq_1、freq_2、freq_3。例如对于LPDDR4，则 px30s_dmc_opp_table 所使能的频率，要对应lpddr4_params下的freq_0、freq_1、freq_2、freq_3。具体配置如下：

```
    px30s_dmc_opp_table: px30s-dmc-opp-table {
        compatible = "operating-points-v2";

        opp-328000000 {
            opp-hz = /bits/ 64 <328000000>;
            opp-microvolt = <1000000>;
        };
        opp-666000000 {
            opp-hz = /bits/ 64 <666000000>;
            opp-microvolt = <1000000>;
        };
        opp-786000000 {
            opp-hz = /bits/ 64 <786000000>;
            opp-microvolt = <1000000>;
        };
        opp-924000000 {
            opp-hz = /bits/ 64 <924000000>;
            opp-microvolt = <1000000>;
        };
        /* 1056M only for LP4 */
        opp-1056000000 {
            opp-hz = /bits/ 64 <1056000000>;
            opp-microvolt = <1000000>;
            status = "disabled";
        };
    };
```

```
/ {
    ...
    lpddr4_params: lpddr4-params {
        ...
        /* freq info, freq_0 is final frequency, unit: MHz */
        freq_0 = <924>;
        freq_1 = <328>;
        freq_2 = <666>;
        freq_3 = <786>;
        ...
    };
};
```

若 LPDDR4 需要跑1056MHz，则需要修改 lpddr4_params 的freq_0、freq_1、freq_2、freq_3其中一个频率值改为1056，并且在 px30s_dmc_opp_table 节点里 disabled 旧频点，新增加使能 1056000000 的频率点。

改为如下配置：

```
    px30s_dmc_opp_table: px30s-dmc-opp-table {
        compatible = "operating-points-v2";

        opp-328000000 {
            opp-hz = /bits/ 64 <328000000>;
            opp-microvolt = <1000000>;
        };
        opp-666000000 {
            opp-hz = /bits/ 64 <666000000>;
            opp-microvolt = <1000000>;
        };
        opp-786000000 {
            opp-hz = /bits/ 64 <786000000>;
            opp-microvolt = <1000000>;
        };
        opp-924000000 {
            opp-hz = /bits/ 64 <924000000>;
            opp-microvolt = <1000000>;
            status = "disabled";
        };
        /* 1056M only for LP4 */
        opp-1056000000 {
            opp-hz = /bits/ 64 <1056000000>;
            opp-microvolt = <1000000>;
        };
    };
```

```
/ {
    ...
    lpddr4_params: lpddr4-params {
        ...
        /* freq info, freq_0 is final frequency, unit: MHz */
        freq_0 = <1056>;
        freq_1 = <328>;
        freq_2 = <666>;
        freq_3 = <786>;
        ...
    };
};
```

1.12.3 DDR bin包含4个频率点的平台，其DDR频率的选择

见"DDR bin包含4个频率点的平台"章节所列平台，除去RV1126，其他平台都适应本节所讲的方法。

这些平台loader有变4次频率，保存着相应频率的training结果。

如下的F1，F2，F3，F0就是这4个频率，默认以F0退出。

```
DDR ... v1.14
LPDDR4X, 2112MHz
channel[0] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
channel[1] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
channel[2] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
channel[3] BW=16 Col=10 Bk=8 CS0 Row=16/0 CS1 Row=16/0 CS=2 Die BW=16
Size=2048MB
...
change to F1: 528MHz  /* 4个DDR频率 */
change to F2: 1068MHz
change to F3: 1560MHz
change to F0: 2112MHz
out
```

这些平台，虽然DDR变频支持的频率也是在loader阶段就确定，但这些平台的kernel dmc驱动会自动获取支持的DDR频率值，因此不存在dmc_opp_table频率匹配问题。

这些平台的kernel dmc节点如下

```
dmc: dmc {
    ...
    system-status-freq = <
    /*system status         freq(KHz)*/
    SYS_STATUS_NORMAL       DMC_FREQ_LEVEL_MID_HIGH
    SYS_STATUS_REBOOT       DMC_FREQ_LEVEL_MID_LOW
    ...
    SYS_STATUS_SUSPEND      DMC_FREQ_LEVEL_LOW
    ...
    SYS_STATUS_PERFORMANCE  DMC_FREQ_LEVEL_HIGH
    ...
    >;
```

```
    ...
};
```

各个场景对应的频率不再是直接的频率，而是DMC_FREQ_LEVEL_HIGH等定义，这些定义，能自动对应到loader的4个频率。如果用"修改DDR bin文件"的方法，修改了4个频率，DMC_FREQ_LEVEL_HIGH这些定义能自动变为修改后的4个频率。

以上面开机loader的4个频率为例，说明对应关系如下，

宏定义	频率	说明
DMC_FREQ_LEVEL_HIGH	F0的2112MHz	最高频
DMC_FREQ_LEVEL_MID_HIGH	F3的1560MHz	次高频
DMC_FREQ_LEVEL_MID_LOW	F2的1068MHz	次低频
DMC_FREQ_LEVEL_LOW	F1的528MHz	最低频

想修改F0、F1、F2、F3对应的频率值，见"修改DDR bin文件"

想修改SYS_STATUS_NORMAL、SYS_STATUS_REBOOT等场景对应的频率，可以将对应的频率换成DMC_FREQ_LEVEL_HIGH、DMC_FREQ_LEVEL_MID_HIGH、DMC_FREQ_LEVEL_MID_LOW、DMC_FREQ_LEVEL_LOW中的一个。

1.13 如何修改 DDR 某个频率对应的电压

如果是为了定位问题，想通过命令来修改电压，可以采用如下方式

kernel 4.4以后版本：可以通过`/sys/kernel/debug/regulator/`节点直接调整某一路的电压。要求先对DDR进行定频。否则这里调整电压，与DDR变频的调整电压，会冲突。DDR定频参考"DDR 如何定频"章节

以3588调整电压为例：

```
cat /sys/kernel/debug/regulator/vdd_ddr_s0/voltage
echo 700000 > /sys/kernel/debug/regulator/vdd_ddr_s0/voltage
```

各平台修改DDR频率对应电压需要调整的regulator如下

平台	需要调整的节点
RK3308/RK3308B	vdd_core或vdd_log
RK3576	vdd_logic_s0和vdd_ddr_s0
RK3588	vdd_log_s0和vdd_ddr_s0
Other SOC	vdd_center或vdd_logic

kernel 4.4：需要编译 kernel 时，打开 pm_tests 选项(make ARCH=arm64 menuconfig ->Device Drivers -> SOC （System On Chip) specific Drivers -> Rockchip pm_test support)

kernel 3.10：需要编译 kernel 时，打开 pm_tests 选项(make menuconfig ->System Type -> /sys/pm_tests/ support)。调整 DDR 电压的命令是：

RK3399

```
echo set vdd_center  900000 > /sys/pm_tests/clk_volt
```

其他平台

```
echo set vdd_logic  1200000 > /sys/pm_tests/clk_volt
```

如果没有 pm_tests 或者命令无法满足要求，就需要改 kernel 固件，按如下步骤

- 对于 kernel 4.4，需要找到 dts 中 dmc_opp_table 节点。举例如下，RK3399 EVB 板的在 arch/arm64/boot/dts/rockchip/rk3399-opp.dtsi、RK3368 在 arch/arm64/boot/dts/rockchip/rk3368.dtsi，以 RK3399 为例。

```
/* 只有频率等于dmc_opp_table所列opp-hz的，才会按该频率运行；小于opp-hz，频率会向上取；如果频率超过这个表格的最大opp-hz，只会按最大opp-hz工作;这是跟kernel 3.10不同的地方 */
dmc_opp_table: opp-table3 {
    compatible = "operating-points-v2";

    opp-200000000 {
        /* 当DDR频率等于200MHz时，使用这个电压;小于200MHz，就按200MHz运行 */
        opp-hz = /bits/ 64 <200000000>;
        opp-microvolt = <825000>;    //vdd_center电压
    };
    opp-300000000 {
        opp-hz = /bits/ 64 <300000000>;
        opp-microvolt = <850000>;
    };
    opp-400000000 {
        opp-hz = /bits/ 64 <400000000>;
        opp-microvolt = <850000>;
    };
    opp-528000000 {
        opp-hz = /bits/ 64 <528000000>;
        opp-microvolt = <900000>;
    };
    opp-600000000 {
        opp-hz = /bits/ 64 <600000000>;
        opp-microvolt = <900000>;
    };
    opp-800000000 {
        opp-hz = /bits/ 64 <800000000>;
        opp-microvolt = <900000>;
    };
};
```

RK3368 为例：

```
/* 只有频率等于dmc_opp_table所列opp-hz的，才会按该频率运行；小于opp-hz，频率会向上取；如果频率超过这个表格的最大opp-hz，只会按最大opp-hz工作;这是跟kernel 3.10不同的地方 */
dmc_opp_table: opp_table2 {
    compatible = "operating-points-v2";

    opp-192000000 {
        /* 当DDR频率等于200MHz时，使用这个电压;小于200MHz，就按200MHz运行 */
        opp-hz = /bits/ 64 <192000000>;
        opp-microvolt = <1100000>; //vdd_logic电压
    };
```

```
    opp-300000000 {
        opp-hz = /bits/ 64 <300000000>;
        opp-microvolt = <1100000>;
    };
    opp-396000000 {
        opp-hz = /bits/ 64 <396000000>;
        opp-microvolt = <1100000>;
    };
    opp-528000000 {
        opp-hz = /bits/ 64 <528000000>;
        opp-microvolt = <1100000>;
    };
    opp-600000000 {
        opp-hz = /bits/ 64 <600000000>;
        opp-microvolt = <1100000>;
    };
};
```

可以修改对应频率对应的电压。因为频率电压表采用的是小于等于指定频率，就使用相应电压。如果新增的频率超出了这张表格的最高频率，使得找不到对应电压，会导致 DDR 无法切换到该新增频率。这时候，就需要增加一项该频率对应的电压表。

- 对于 kernel 3.10，需要找到 dts 中的 clk_ddr_dvfs_table 节点。举例如下，RK3288 SDK 板的最终的 clk_ddr_dvfs_table 在 arch/arm/boot/dts/rk3288-tb_8846.dts 中，

```
&clk_ddr_dvfs_table {
    /* 频率电压表是这个 */
    operating-points = <
        /* KHz    uV */
        /* 表示DDR频率小于等于200MHz，logic使用1050mV，其他行以此类推 */
        200000 1050000
        300000 1050000
        400000 1100000
        533000 1150000
        >;

    ...
    status="okay";
};
```

可以修改对应频率对应的电压。因为频率电压表采用的是小于等于指定频率，就使用相应电压。如果新增的频率超出了这张表格的最高频率，使得找不到对应电压，会导致 DDR 无法切换到该新增频率。这时候，就需要增加一项该频率对应的电压表。

- 对于 kernel 3.0，需要在板级的 borad-**.c 文件中修改 dvfs_ddr_table。举例如下，RK3066 SDK 板的板级文件在 arch/arm/mach-rk30/board-rk30-sdk.c 中

```
static struct cpufreq_frequency_table dvfs_ddr_table[] = {
    {.frequency = 200 * 1000 + DDR_FREQ_SUSPEND, .index = 1050 * 1000},
    {.frequency = 300 * 1000 + DDR_FREQ_VIDEO, .index = 1050 * 1000},
    {.frequency = 400 * 1000 + DDR_FREQ_NORMAL, .index = 1125 * 1000},
    {.frequency = CPUFREQ_TABLE_END},
};
```

dvfs_ddr_table 表格中的.index 就是对应的电压，单位是 uV。

1.14 如何关闭 DDR 的负载变频功能，只留场景变频

- 对于 kernel 4.4及以后版本，需要找到 dts 中 dmc 节点的 auto-freq-en。举例如下，RK3399 EVB 板的在 arch/arm64/boot/dts/rockchip/rk3399.dtsi

```
dmc: dmc {
    compatible = "rockchip,rk3399-dmc";
    ...
    auto-min-freq = <400000>;
    /* 这个设为0，就关闭DDR负载变频功能，只留场景变频 */
    auto-freq-en = <0>;
    ...
};
```

- 对于 kernel 3.10，需要找到 dts 中的 clk_ddr_dvfs_table 节点。举例如下，RK3288 SDK 板的最终的 clk_ddr_dvfs_table 在 arch/arm/boot/dts/rk3288-tb_8846.dts 中，

```
&clk_ddr_dvfs_table {
    ...
    /* 这个设为0，就关闭DDR负载变频功能，只留场景变频 */
    auto-freq=<0>;
    ...
    status="okay";
};
```

- 对于 kernel 3.0，本身就不支持负载变频

1.15 DDR 如何定频

如果是为了定位问题，想通过命令来定频，可以采用如下方式

kernel 4.4及以后版本：

获取固件支持的 DDR 频率：
`cat /sys/class/devfreq/dmc/available_frequencies`

设置频率：

`echo userspace > /sys/class/devfreq/dmc/governor`
`echo 300000000 > /sys/class/devfreq/dmc/min_freq` //这条是防止要设置的频率低于min_freq，导致设置失败
`echo 300000000 > /sys/class/devfreq/dmc/userspace/set_freq`

这里的频率单位是 Hz，可用的值，必须在`cat /sys/class/devfreq/dmc/available_frequencies`结果中

kernel 3.10：

需要编译 kernel 时，打开 pm_tests 选项(make menuconfig ->System Type -> /sys/pm_tests/ support)。DDR 定频的命令是：

`echo set clk_ddr 300000000 > /sys/pm_tests/clk_rate`

这里的频率单位是 Hz，具体要固定要什么频率可以根据需求改命令的参数。

如果上面的方法不可行，只能修改代码或 dts 了

- 对于 kernel 4.4及以后版本，如果上述方法不行，一般都是因为你需要定频的目标频率，在`cat /sys/class/devfreq/dmc/available_frequencies`里找不到。

  添加频率的方法是，找到板级 dts 文件，在 dmc_opp_table 中增加你需要的频率。举例如下，RK3399 EVB 板的在 arch/arm64/boot/dts/rockchip/rk3399-opp.dtsi，假设你需要增加的是 666MHz

```
dmc_opp_table: opp-table3 {
compatible = "operating-points-v2";

opp-200000000 {
    opp-hz = /bits/ 64 <200000000>;
    opp-microvolt = <825000>;
};
...
opp-666000000 {
    /* 当DDR频率等于666MHz时，使用这个电压 */
    opp-hz = /bits/ 64 <666000000>;
    opp-microvolt = <900000>;    //vdd_center电压
};
opp-800000000 {
    opp-hz = /bits/ 64 <800000000>;
    opp-microvolt = <900000>;
};
};
```

  dts 添加完后，就可以用前面的命令，进行固定频率了。

  如果不想通过开机输入命令来固定频率，而是希望开机后一直运行在一个固定频率，可以像如下一样修改 dts。假设你需要开机后固定的频率是 666MHz，举例如下，RK3399 EVB 板的 dmc 节点在 arch/arm64/boot/dts/rockchip/rk3399-evb.dtsi

```
/* dfi必须为okay，由于早期代码，dmc节点有依赖于dfi节点。如果dfi节点为disabled，也会导致dmc节点无效。所以最好dfi节点的status保持跟dmc一致 */
&dfi {
    status = "okay";
};

&dmc {
status = "okay";
...
system-status-freq = <
    /*system status         freq(KHz)*/
    SYS_STATUS_NORMAL       666000
    /* 去掉其他所有场景 */
    /*
    SYS_STATUS_REBOOT       528000
    SYS_STATUS_SUSPEND      200000
    SYS_STATUS_VIDEO_1080P  200000
    SYS_STATUS_VIDEO_4K     600000
    SYS_STATUS_VIDEO_4K_10B 800000
    SYS_STATUS_PERFORMANCE  800000
    SYS_STATUS_BOOST        400000
    SYS_STATUS_DUALVIEW     600000
    SYS_STATUS_ISP          600000
    */
```

```
>;
...
auto-min-freq = <666000>;
/* auto-freq-en要为0，否则会动态变频 */
auto-freq-en = <0>;
};
```

- 对于 kernel 3.10，需要找到 dts 中的 clk_ddr_dvfs_table 节点。举例如下，RK3288 SDK 板的最终的 clk_ddr_dvfs_table 在 arch/arm/boot/dts/rk3288-tb_8846.dts 中，

```
&clk_ddr_dvfs_table {
    operating-points = <
        /* KHz    uV */
        /* 3，如果要固定的频率，超出了这个表的最大频率，还需要添加该频率的电压表 */
        200000 1050000
        300000 1050000
        400000 1100000
        533000 1150000
        >;

    freq-table = <
        /*status        freq(KHz)*/
        /* 2，其他场景都注释掉，只留SYS_STATUS_NORMAL，并把他定义为你需要固定的频率 */
        SYS_STATUS_NORMAL   400000
        /*
        SYS_STATUS_SUSPEND  200000
        SYS_STATUS_VIDEO_1080P  240000
        SYS_STATUS_VIDEO_4K     400000
        SYS_STATUS_VIDEO_4K_60FPS     400000
        SYS_STATUS_PERFORMANCE  528000
        SYS_STATUS_DUALVIEW 400000
        SYS_STATUS_BOOST    324000
        SYS_STATUS_ISP      400000
        */
        >;
    bd-freq-table = <
        /* bandwidth   freq */
        5000            800000
        3500            456000
        2600            396000
        2000            324000
    >;
    auto-freq-table = <
        240000
        324000
        396000
        528000
        >;
    /* 1，负载变频的，要设置为0 */
    auto-freq=<0>;
    /*
     * 0: use standard flow
     * 1: vop dclk never divided
     * 2: vop dclk always divided
     */
    vop-dclk-mode = <0>;
    status="okay";
```

```
};
```

只要上面 3 步，就可以完成固定频率的固件，

1. 负载变频的，要设置为 0
2. 注释其他场景，只留 SYS_STATUS_NORMAL，并把他定义为你需要固定的频率
3. 如果要固定的频率，超出了频率电压表的最大频率，还需要添加该频率的电压表

- 对于 kernel 3.0，需要在板级的 borad-**.c 文件中修改 dvfs_ddr_table。举例如下，RK3066 SDK 板的板级文件在 arch/arm/mach-rk30/board-rk30-sdk.c 中

```
static struct cpufreq_frequency_table dvfs_ddr_table[] = {
    /*  */
    /* 1,注释掉其他场景，只留DDR_FREQ_NORMAL */
    //{.frequency = 200 * 1000 + DDR_FREQ_SUSPEND, .index = 1050 * 1000},
    //{.frequency = 300 * 1000 + DDR_FREQ_VIDEO, .index = 1050 * 1000},
    /* 2，把DDR_FREQ_NORMAL定义成你需要的频率，对应的电压有可能需要调整 */
    {.frequency = 400 * 1000 + DDR_FREQ_NORMAL, .index = 1125 * 1000},
    {.frequency = CPUFREQ_TABLE_END},
};
```

只要上面 2 步，就可以完成固定频率的固件，

1. 注释其他场景，只留 DDR_FREQ_NORMAL
2. 把 DDR_FREQ_NORMAL 定义成你需要的频率，对应的电压有可能需要调整

1.16 如何查看 DDR 带宽利用率

kernel 4.4 及以后版本，提供了一个命令，可以简单查看整个 DDR 带宽利用率，但是没有详细每个端口的数据量信息。

```
rk3288: # cat /sys/class/devfreq/dmc/load
11@396000000Hz
```

11 表示 DDR 的当前带宽利用率是 11%

RK还提供一个专门用于观测DDR带宽的可执行文件，详见"Rockchip DDR 带宽工具使用说明"章节

1.17 如何测试 DDR 可靠性

详见“DDR 颗粒验证流程说明”章节

1.18 如何确定 DDR 能运行的最高频率

1. 先添加各种高频的频率电压表，如何添加见“如何修改 DDR 频率”和“如何修改 DDR 某个频率对应的电压”章节
2. 见"Rockchip DDR DQ 眼图工具指南"章节，对有提供眼图工具的平台，进行眼图确认
3. 从高频到低频，运行 google stressapptest，碰到报错，就降低频率，再运行。没报错，可以多运行一段时间，还是没报错，进入下一步。

google stressapptest 在

https://redmine.rock-chips.com/documents/49 -> DDR相关资料_VerX.XX.7z -> DDR颗粒验证_DDR测试资源文件 -> static_stressapptest

如何使用，详见“DDR 颗粒验证流程说明”章节。

4. 上一步已经大概摸清了最高频率的位置，这里再运行一个memtester。方法一样，碰到报错，就降低频率，再运行。没报错，可以多运行一段时间，还是没报错，就可以确认最高频率点。

memtester 在

https://redmine.rock-chips.com/documents/49 -> DDR相关资料_VerX.XX.7z -> DDR颗粒验证_DDR测试资源文件 -> static_memtester

如何使用，详见“DDR 颗粒验证流程说明”章节。

google stressapptest 是一个粗筛选的过程，他能快速报错。而 memtester 是一个细的筛选过程，他报错比较慢，但是主要是针对信号的测试，能覆盖到 google stressapptest 没覆盖到的面。

当然，以上方法，都是基于软件的测试方法，用于快速确认最高频率，实际 DDR 信号和电源是否在最高频能否达标，就不一定了，需要信号测量及拷机。

1.19 怎么判断 DDR 已经进入自刷新（self-refresh 省电模式）

可以通过测量 CKE 信号来判断，而且不需要带宽很高的示波器就可以。

CKE 状态	解释
低电平(时间大于 7.8us)	处于自刷新状态
高电平	处于正常工作状态

如果测到的 CKE 是一会儿为高一会儿为低，也是按上表的解释来理解，即一会儿进入了自刷新，一会儿退出自刷新，处于正常工作状态。

注意：CKE 为低电平时间一定要大于 7.8us 才算进入自刷新，因为 power-down 状态也是 CKE 为低，但时间小于 7.8us，不要混淆。

1.20 怎么判断 DDR 已经进入 auto power-down 省电模式

可以通过测量 CKE 信号来判断，而且不需要带宽很高的示波器就可以。

CKE 状态	解释
低电平(时间小于 7.8us)	处于 auto power-down 状态
高电平	处于正常工作状态

auto power-down 状态，测到的 CKE 状态，是 CKE 保持低电平将近 7.8us（DDR3、DDR4）或 3.9us(LPDDR2、LPDDR3、LPDDR4），然后拉高一小段时间，再进入低电平近 7.8us 或 3.9us，如此一直往复。

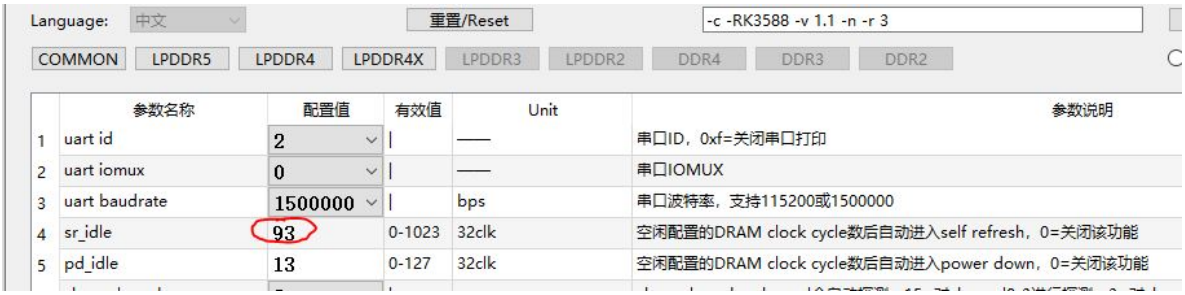
注意：CKE 为低电平时间一定要小于 7.8us（DDR3、DDR4）或 3.9us(LPDDR2、LPDDR3、LPDDR4）才是 auto power-down，如果时间大于 7.8us，就是自刷新，不要混淆。

1.21 怎么关闭self-refresh、 power-down省电模式

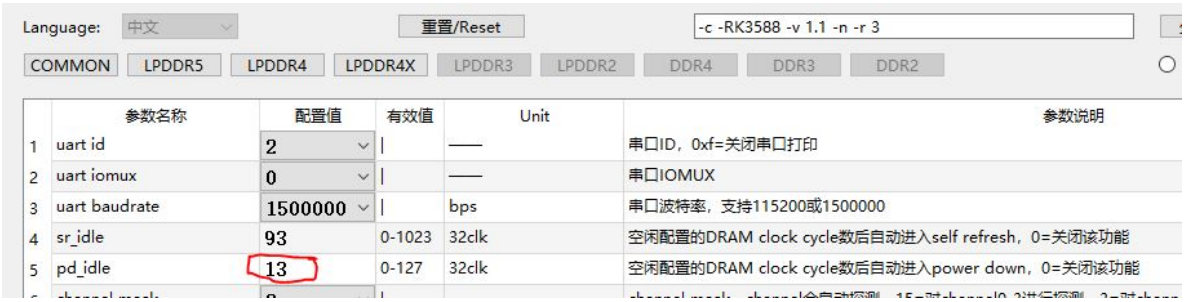
所有DDR颗粒都有self-refresh和power-down这些基本省电功能，软件上也会开启这些功能，用于省电，当DDR处于空闲时，会根据空闲时间，进入self-refresh或power-down省电模式。

某些客户不需要省电功能；或者只在乎性能，不在乎功耗；或者DDR颗粒self-refresh功能或power-down功能有问题，可以通过如下方法关闭这些省电模式。

将下图sr_idle的值设置为0，就会关闭根据DDR空闲，让DDR颗粒进入self-refresh的功能。



将下图pd_idle的值设置为0，就会关闭根据DDR空闲，让DDR颗粒进入power-down的功能。



1.22 如何调整 DQ、DQS、CA、CLK 的 de-skew

主要由于硬件 PCB 中 DDR 走线不等长，可以通过调整 de-skew，达到类似 DDR 走线等长的效果。de-skew 功能就是 DDR PHY 内部串联在信号线上的一个个延迟单元，可以通过 de-skew 寄存器控制各个信号线上串联的延迟单元的个数来改变每个信号线的延迟。

1.22.1 调整 kernel 中的 de-skew

要调整 kernel 中的 de-skew，目前也只有 RK322xh、RK3328 支持。需要改对应的 dts 文件

芯片：RK322xh、RK3328

代码位置：

arch/arm64/boot/dts/rk322xh-dram-default-timing.dtsi

arch/arm64/boot/dts/rk322xh-dram-2layer-timing.dtsi

如果产品有覆盖这 2 个的定义，以新定义的为准。

修改：

根据发布的“deskew 自动扫描工具”扫出来的结果，选择 mid 值，修改到对应 dts 定义中。

“deskew 自动扫描工具”的使用请按照《3228H deskew 自动扫描工具使用说明.pdf》来做

1.22.2 调整 loader 中的 de-skew

要调整 loader 中的 de-skew，目前只有 RK3308 支持。

芯片：RK3308

需要的文件：

deskew 自动扫描工具，3308_deskew.exe，RK3308_DDRXPXXXXXX_Template_VXX_de-skew.txt，rk3308_ddr_XXXMHz_uartX_mX_vX.XX.bin

修改：

根据发布的“deskew 自动扫描工具”扫出来的结果，选择 mid 值，修改到 RK3308_DDRXPXXXXXX_Template_VXX_de-skew.txt 对应定义中。使用 3308_deskew.exe 将 RK3308_DDRXPXXXXXX_Template_VXX_de-skew.txt 定义的 de-skew 修改到 rk3308_ddr_XXXMHz_uartX_mX_vX.XX.bin 中。

“deskew 自动扫描工具”的使用请按照《deskew 自动扫描工具使用说明.pdf》来做。

1.23 U-Boot下运行DDR压力测试

U-Boot下我们移植了2个常用DDR压力测试程序stressapptest和memtester

主要用于：

- kernel无法启动时，在U-Boot下对DDR进行压力测试，以排除DDR问题
- kernel不具备运行stressapptest或memtester的条件。

  如：kernel中有较多模块不完善，会引起系统不稳定，导致stressapptest或memtester无法长时间运行。
- 测试更大的DDR空间

  kernel由于系统占用，只能对剩余DDR空间进行测试。在U-Boot下测试，占用空间小很多。可以测到更大的DDR空间。
- 指定物理地址测试

1.23.1 stressapptest

开启

```
make ARCH=arm64 menuconfig ->Command line interface -> DDR Tool -> Enable DDR Tool
```

再开启

```
make ARCH=arm64 menuconfig ->Command line interface -> DDR Tool -> Enable DDR Tool -> Enable stressapptest for ddr
```

编译U-Boot，烧录uboot.img后。

重启设备，在PC串口终端上，持续按住CTRL+C键，直到U-Boot进入命令行模式，如下

```
  pclk_top_root 100000 KHz
  aclk_low_top_root 396000 KHz
Net:   No ethernet found.
Hit key to stop autoboot('CTRL+C'):  0
=> <INTERRUPT>
=> <INTERRUPT>
```

输入`stressapptest -h`，按照参数说明来测试。

1.23.2 memtester

开启

`make ARCH=arm64 menuconfig ->Command line interface -> DDR Tool -> Enable DDR Tool`

再开启

`make ARCH=arm64 menuconfig ->Command line interface -> DDR Tool -> Enable DDR Tool -> Enable memtester for ddr`

编译U-Boot，烧录uboot.img后。

重启设备，在PC串口终端上，持续按住CTRL+C键，直到U-Boot进入命令行模式，如下

```
  pclk_top_root 100000 KHz
  aclk_low_top_root 396000 KHz
Net:   No ethernet found.
Hit key to stop autoboot('CTRL+C'):  0
=> <INTERRUPT>
=> <INTERRUPT>
```

输入`memtester -h`，按照参数说明来测试。

1.24 RK3568 ECC的使能

ECC 指 Error Correcting Code ， 而DDR ECC 是对DDR数据进行错误检查和纠正的。

RK3568支持的是SideBand ECC，即在DDR数据通道旁增加一个专门存放ECC数据的DDR通道。

其ECC具有纠错1bit，发现2bit错误的能力，即SEC/DED ECC（Single Error Correction/Double Error Detection）。

1. 使能ECC：只要DDR_ECC_DQ0-7有贴颗粒，ECC会自动enable。
2. DDR_ECC_DQ0-7的作用：ECC是32bit的DQ数据+7bit的ECC数据。DDR_ECC_DQ0-7用于存储DQ0-DQ31计算得到的ECC数据。所以需要多贴一颗用于存放ECC数据的颗粒。
3. 多贴的一颗ECC颗粒的要求：颗粒类型，Row/Col/Bank需要与DQ0-31上所贴的颗粒一致。
4. 支持的颗粒类型：所有颗粒类型均支持ECC。

1.25 如何在 kernel 获取 DDR ECC 的信息

DDR ECC信息在kernel的edac架构里，对于支持DDR ECC功能的平台，如RK3568，且硬件上支持使能DDR ECC（详见 “RK3568 ECC的使能”章节），在dtsi里开启edac模块。若平台dtsi无edac节点，则表明此平台不支持。

```
    edac: edac {
        ...
        status = "okay";
    };
```

RK EDAC将注册到系统位置：

```
sys/devices/system/edac/mc/mc0
或
sys/bus/edac/devices/mc/mc0/
```

获取RK EDAC名称：

```
# cat sys/devices/system/edac/mc/mc0/mc_name
rk_edac_ecc
```

获取本次开机后累计的单bit可检测可纠正性错误correctable error (ce)数量：

```
# cat sys/devices/system/edac/mc/mc0/ce_count
0
```

获取本次开机后累计的多bit可检测不可纠正性错误uncorrectable error (ue)数量：

```
# cat sys/devices/system/edac/mc/mc0/ue_count
0
```

当系统重新启动时，内核会重新初始化相关的EDAC模块和计数器，即之前记录的错误数量会被清零，从零开始计数。

1.26 如何在 kernel 里软件注入 DDR ECC 错误

需要确认kernel里对应平台dts文件里的edac节点已开启，且硬件上支持DDR ECC。

```
    edac: edac {
        ...
        status = "okay";
    };
```

系统起来后，通过如下节点选择注入DDR ECC错误类型

```
echo ce > /sys/module/rockchip_edac/parameters/edac_poison_mode
echo ue > /sys/module/rockchip_edac/parameters/edac_poison_mode
```

edac_poison_mode只支持如下两种参数：

1. ce，表示注入单bit可检测可纠正性错误correctable error；
2. ue，表示注入多bit可检测不可纠正性错误uncorrectable error；

再通过如下节点触发DDR ECC错误（其中echo进去的数字是触发DDR ECC错误的数量）：

```
echo 10 > /sys/module/rockchip_edac/parameters/rockchip_edac_trigger
```

Note：

1. 若出现如下异常打印，需确认edac已开启，且硬件支持DDR ECC。

```
echo 1 > /sys/module/rockchip_edac/parameters/rockchip_edac_trigger
sh: echo: write error: Operation not permitted
```

2. 多bit可检测不可纠正性错误uncorrectable error（ue）的默认处理是重启系统，单bit可检测可纠正性错误correctable error(ce)的默认处理是kernel报告warning，并将出错的数量记录在文件`/sys/devices/system/edac/mc/mc0/ce_count`里。

在HAL里注入DDR ECC错误，请参考章节“Rockchip_Developer_Guide_HAL_DDR_ECC”。

1.27 如何查看DDR厂商ID

只有LPDDR类型（如LPDDR2，LPDDR3，LPDDR4，LPDDR4X，LPDDR5，LPDDR5X）才有厂商的ID，DDR类型（如DDR2，DDR3，DDR4）是没有厂商ID的。

1.27.1 通过kernel的dmcdbg节点

如下平台，支持通过`proc/dmcdbg/dmcinfo`获取DDR厂商ID

- RK3566
- RK3568
- RK3326
- RK3326S

需要先在对应平台dtsi开启dmcdbg节点（若节点不存在，则此平台不支持）。如RK356X，在`arch/arm64/boot/dts/rockchip/rk3568.dtsi`文件，开启dmcdbg节点：

```
dmcdbg: dmcdbg {
    compatible = "rockchip,rk3568-dmcdbg";
    status = "okay";
};
```

重新编译kernel固件，烧录，开机后，通过 cat 对应节点`proc/dmcdbg/dmcinfo`获取DDR厂商ID和其他DDR信息。

```
console:/ # cat proc/dmcdbg/dmcinfo
DramType:       LPDDR4
Dram ID:        MR5=0x1,MR6=0x0,MR7=0x1/* 其中MR5是指厂商ID，MR6，MR7是预留给厂商定义的version ID */

DramCapacity:
CS Count:       1
Bus Width:      32 bit
Column:         10
Bank:           8
```

```
CS0_Row:        16
CS1_Row:        0
DieBusWidth:    16 bit
TotalSize:      2048 MB

devfreq/dmc:    Enable
governor:       dmc_ondemand

cur_freq:       780000000

NOTE:
more information about dmc can get from /sys/class/devfreq/dmc.
```

1.27.2 通过loader输出信息

RK3576/RK3588 loader输出中含有厂商ID

```
DDR ... v1.14
LPDDR5, 2736MHz
channel[0] BW=16 Col=10 Bk=16 CS0 Row=15 CS1 Row=15 CS=2 Die BW=16 Size=2048MB
channel[1] BW=16 Col=10 Bk=16 CS0 Row=15 CS1 Row=15 CS=2 Die BW=16 Size=2048MB
channel[2] BW=16 Col=10 Bk=16 CS0 Row=15 CS1 Row=15 CS=2 Die BW=16 Size=2048MB
channel[3] BW=16 Col=10 Bk=16 CS0 Row=15 CS1 Row=15 CS=2 Die BW=16 Size=2048MB
Manufacturer ID:0xff /* 这就是厂商ID */
...
change to F0: 2736MHz
out
```

RK3399 loader输出中含有厂商ID

```
DDR Version ...
In
channel 0
...
MR5=0xFF    /* 这就是厂商ID */
...
change freq to 416MHz 0,1
Channel 0: LPDDR4,416MHz
Bus Width=32 Col=10 Bank=8 Row=16 CS=1 Die Bus-Width=16 Size=2048MB
Channel 1: LPDDR4,416MHz
Bus Width=32 Col=10 Bank=8 Row=16 CS=1 Die Bus-Width=16 Size=2048MB
...
OUT
```

1.27.3 注意点

应当注意如下几点：

1. 只有LPDDR类型（如LPDDR2，LPDDR3，LPDDR4，LPDDR4X，LPDDR5，LPDDR5X）才有厂商的ID，DDR类型（如DDR2，DDR3，DDR4）是没有厂商ID的。
2. 厂商ID显示的是晶圆来自哪个厂商，并不是封装后的DDR品牌信息。因为很多DDR品牌并不具备生产晶圆的能力。

3. 所有LPDDR，都无法获得封装丝印上的型号。这跟Nand Flash或eMMC不同。
4. 晶圆厂商ID的解析，遵循JEDEC标准的JEP166。

1.27.4 厂商ID对照表

如下是截止本文档完成时，最新的ID对照表。

后续有更新，可以自行在https://www.jedec.org搜索JEP166，找到最新的ID对照表

LPDDR2 ，LPDDR3 厂商 ID 对照表：

MR5值	厂商名称
0x1	Samsung
0x2	Qimonda
0x3	Elpida
0x4	Etron
0x5	Nanya
0x6	SK hynix
0x7	Mosel
0x8	Winbond
0x9	ESMT
0xa	Zentel
0xb	Spansion
0xc	SST
0xd	ZMOS
0xe	Intel
0x12	Being Advanced Memory Corp
0x1a	Xi'an UniIC Semiconductors Co., Ltd
0x1b	ISSI
0x1c	JSC
0xaa	Tezzaron
0xc2	Macronix
0xf8	Fidelix
0xfc	eveRAM
0xfd	AP Memory
0xfe	Numonyx
0xff	Micron

LPDDR4 厂商 ID 对照表：

MR5值	厂商名称
0x1	Samsung
0x5	Nanya
0x6	SK hynix
0x8	Winbond
0x9	ESMT
0x13	CXMT
0x1a	Xi'an UniIC Semiconductors Co., Ltd
0x1c	JSC
0xf8	Fidelix
0xf9	Ultra Memory
0xfd	AP Memory
0xff	Micron

LPDDR5 厂商 ID 对照表：

MR5值	厂商名称
0x1	Samsung
0x5	Nanya
0x6	SK hynix
0x8	Winbond
0x9	ESMT
0x13	CXMT
0xe5	Dosilicon
0xff	Micron

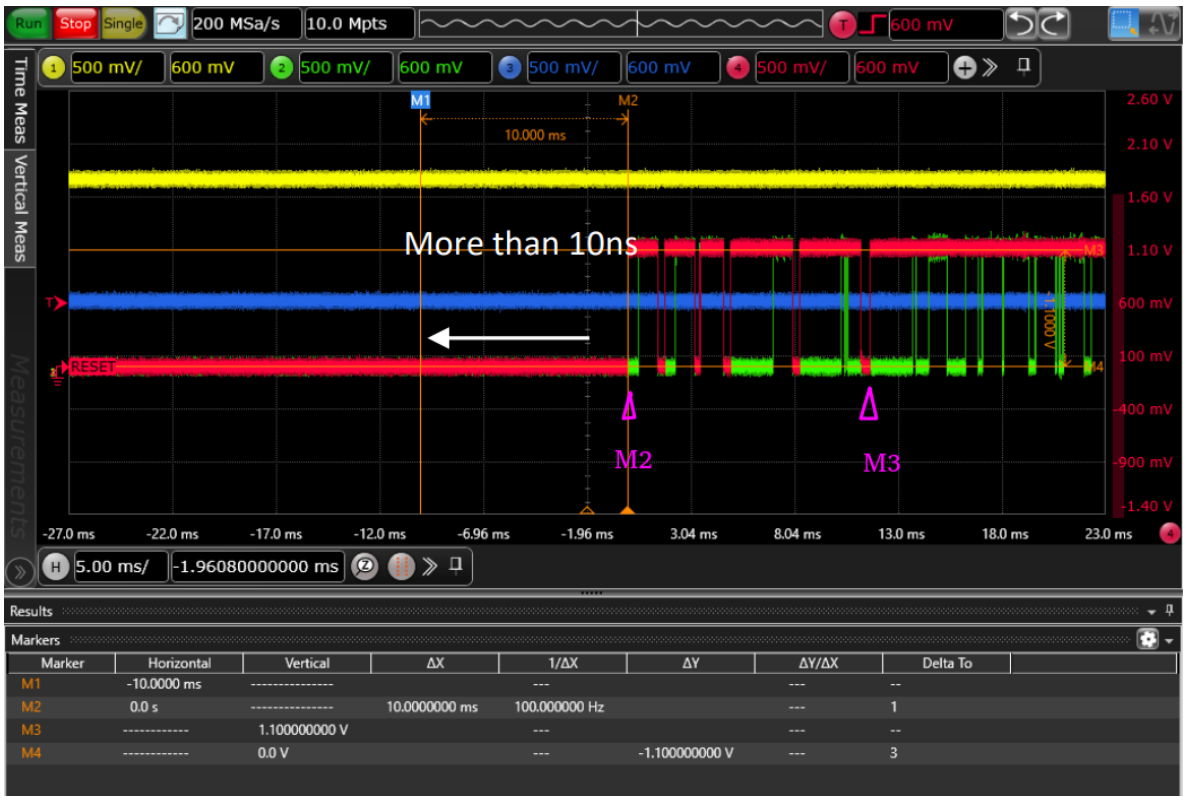
1.28 常见信号相关问题

1.28.1 tINIT3不满足协议

原因：RK平台做DDR初始化时，RESET信号会有多次reset行为，测量厂商误将第一次RESET信号高电平作为tINIT3的测量点，所以导致测到的tINIT3不满足协议。

正确测量方法，应该以最后一个RESET信号高电平作为tINIT3的测量点。如下图，

不能以M2为测量点，而应该以M3作为测量点。



1.28.2 颗粒ODT开启注意事项

- 按JEDEC标准，LPDDR4/LPDDR4X颗粒在800MHz以下不支持开ODT
- 按JEDEC标准，DDR4颗粒在625MHz以下不支持开ODT

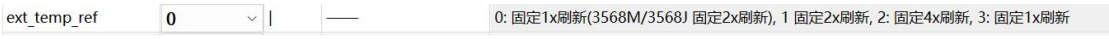
1.29 高温刷新率（refresh rate）问题

当温度高于85°C，需要对DDR加倍刷新。

此要求，对工规、车规产品尤其需要注意。

RK平台有如下几个方法可以加倍刷新：

- LPDDR4、LPDDR4X、LPDDR5、LPDDR5X颗粒自身带有温度传感器，能根据温度，通过MR4告知需要多少刷新率，RK有些平台实现了一套Automatic Temperature Derating的功能，简称derate。可以根据颗粒提供的MR4信息，自动调整刷新率。从而实现刷新率和温度匹配。
- LPDDR3由于一些硬件原因不支持derate功能，采用加倍刷新。
- 对DDR3、DDR4颗粒，因为自身不带温度传感器，也没有MR4告知需要多少刷新率的功能，所以无法使用derate。这类颗粒，采用加倍刷新。
- 客户可以修改DDR bin的ext_temp_ref，自行加倍刷新

  修改方法见"修改DDR bin文件"

  Note：RK的工规、车规、宽温芯片，都有对刷新率进行修改，无需通过这个工具自行加倍。

  工具中，固定2x/4x/1x刷新的对应关系如下

ext_temp_ref	DDR2/DDR3/DDR4	LPDDR3/LPDDR4/LPDDR4x/LPDDR5
固定1x刷新	刷新周期7.8us	刷新周期3.9us

ext_temp_ref	DDR2/DDR3/DDR4	LPDDR3/LPDDR4/LPDDR4x/LPDDR5
固定2x刷新	刷新周期3.9us	刷新周期1.95us
固定4x刷新	刷新周期1.95us	刷新周期0.975us

- 对于非工规、车规、宽温的常规平台除开启derate功能的颗粒类型外，默认均采用1倍刷新率。

目前RK工规、车规、宽温平台处理方法及DDR bin版本要求，如下：

平台	DDR类型	处理方法及DDR bin版本要求
RK3308J/M RK3308K RK3308GK	所有DDR类型	使用`rk3308_ddr_XXXMHz_uartX_mX_v1.26.bin`及以后版本 默认4倍刷新(即0.25x tREFI)
RK3568J/M	LPDDR3 DDR3 DDR4	使用`rk3568_ddr_XXXXMHz_v1.16.bin`及以后版本 默认2倍刷新(即0.5x tREFI)
RK3568J/M	LPDDR4 LPDDR4X	使用`` `rk3568_ddr_XXXXMHz_v1.18.bin ``及以后版本 默认开启derate功能
RK3588J/M	LPDDR4 LPDDR4X LPDDR5 LPDDR5X	使用`rk3588_ddr_XXXXMHz_v1.13.bin`及以后版本 默认开启derate功能
RV1126K	所有DDR类型	使用`rv1126_ddr_XXXXMHz_v1.03.bin`及以后版本 默认2倍刷新(即0.5x tREFI)

普通平台默认DDR bin有开启derate功能的，如下：

平台	DDR类型	处理方法及DDR bin版本要求
RK3528	LPDDR4 LPDDR4X	使用`rk3528_ddr_XXXXMHz_v1.07.bin` 或`rk3528_ddr_XXXXMHz_2L_PCB_v1.07.bin` 或rk3528_ddr_XXXXMHz_4BIT_PCB_v1.07.bin` 及以后版本 默认开启derate功能
RK3562	LPDDR4 LPDDR4X	使用`rk3562_ddr_XXXXMHz_v1.05.bin` 或`rk3562_ddr_XXXXMHz_ultra_v1.05.bin` 及以后版本 默认开启derate功能
RK3568	LPDDR4 LPDDR4X	使用`rk3568_ddr_XXXXMHz_v1.18.bin`及以后版本 默认开启derate功能
RK3576	LPDDR4 LPDDR4X LPDDR5 LPDDR5X	使用`rk3576_ddr_XXXXMHz_v1.00.bin`及以后版本 默认开启derate功能
RK3588	LPDDR4 LPDDR4X LPDDR5 LPDDR5X	使用`rk3588_ddr_XXXXMHz_v1.13.bin`及以后版本 默认开启derate功能

2. Chapter-2 DDR 问题排查手册

2.1 怎么确认是不是 DDR 问题

1. 查看串口 log
    1. 如果串口 log 是在 loader 中的 DDR 初始化部分报错的话一定是 DDR 问题。
    2. 查看 loader 中 DDR 初始化部分 log 中的 DDR 容量行列 bank 及颗粒类型位宽信息是否正确。如果信息错误可能引起 DDR 问题。
    3. 如果串口 log 是系统中的 panic log 的话，可以多尝试几次看多次 panic 的地址是否一致，如果一致的话基本不可能是 DDR 问题，如果不一致的话有可能是 DDR 问题，也可能是电源问题。
2. 看显示是否正常。如果显示异常是 DDR 问题的概率比较大。
3. 做排查试验：
    1. arm gpu 降频，定频适当抬压，如果有效果的话就不是 DDR 问题。基本上能确认是电源问题。
    2. 关闭 DDR 变频功能，有效果则 DDR 变频导致的问题概率比较大。
    3. 降低 DDR 频率到稳妥频率(如 200M)，如果有效果那很大概率是 DDR 信号质量有问题。

2.2 引起 DDR 问题的几个主要原因

1. 电源问题：
    1. layout 上电容不够，电容摆放位置离芯片太远，电容分布不合理
    2. 电源 feedback 回路没按要求从末端引回到 PMU/DC-DC 端
    3. 敷铜有没有按照 RK 的 layout 规则处理导致电源路径太窄
    4. LQFP 封装的芯片正下方的 GND 需要堆锡保证良好接地，否则会影响芯片内部电源质量以及散热。
2. 信号质量问题
    1. 不等长的走线。RK 大部分平台是不带各种 eye training 的， 不等长的走线会直接牺牲 DDR 的 setup/hold time。
    2. 过窄的线间距。过窄的线间距将会导致严重的串扰问题。
    3. T 型拓扑结构分支不等长。不等长的分支会恶化信号边沿，使边沿非单调。
    4. 信号参考层回路上的不完整。在敷铜时，间隙设置过大导致过孔直接隔断参考层，会导致信号质量下降引起兼容性问题。
3. 颗粒问题
    1. 白牌颗粒，由于没经过测试甚至一些可能是原厂测试淘汰下来的颗粒，良率上无法保证。
    2. 一些特殊渠道的颗粒，可能存在驱动强度偏弱等问题。
    3. hynix 4Gb C die DDR3，如 H5TQ4G63**C**FR ，kernel 3.10 早期代码需要打补丁才能用。（后期修复了，判断标准是：只要进入 kernel 能正常就是 fix 过的，否则就需要补丁）。

2.3 解决 DDR 问题的一些手段

解决 DDR 问题总的办法就是找规律，尝试是否能找到死机的规律，如都在某个频率下死机，休眠唤醒死机的是否是和休眠时间多久有关等。尝试各种方法如定频，尝试不同频率，抬压，改驱动强度等逐个排查可能性缩小问题范围。

1. 对于在 DDR 初始化中报错的问题
    1. 如果有"rd addr 0x... = 0x..."的报错基本上是焊接问题。焊接问题可以用“Rockchip 平台 DDR 测试工具”直接找到问题点。
    2. 如果报”16bit error!!!“，”W FF != R“的话表明 DDR 基本的读写都是错误的。这种情况焊接问题概率比较大。
    3. 打印"unknow device"说明颗粒基本的读写都不对，无法探测到 dram 类型。此时应该检查焊接问题。
    4. 对于个别容量不是 2 的 n 次幂的颗粒，如 768MB,1.5GB,3GB 等特殊颗粒有些版本的代码可能没做好兼容工作，可以更新到最新 loader，如果还有异常的话可以联系 DDR 相关工程师分析。
    5. 对于 DDR loader 中报错的问题，大部分会是焊接问题，可以尝试使用 ddr 测试工具焊接专项选择对应容量的测试项测试分析。
2. 查看 loader 中 DDR 初始化部分 log 中的 DDR 容量行列 bank 及颗粒类型位宽信息是否正确。如果信息错误可能引起 DDR 问题。

    如下图第一行为 DDR 版本号，第 3 行 DDR 频率，第 4 行 DDR 类型，第五行从左到右分别为系统的位宽数，列数，bank 数，行数，片选数，颗粒的位宽数和总容量。第 7 行“OUT"打印出来后表面 DDR 初始化成功并退出，再下面就是 usbplug 或者 miniloader 打印的 log。这中间 Die Bus-Width 比实际的大不会有问题，但是比实际的小会引起死机。

```
DDR Version V1.06 20171026
In
300MHz
DDR3
Bus Width=32 Col=10 Bank=8 Row=15 CS=2 Die Bus-Width=16 Size=2048MB
mach:14
OUT
Boot1 Release Time: 2017-06-12, version: 2.37
```

3. 看显示是否正常。

    当系统死机时虽然 cpu 停下来了，但是 vop 依然会重复着从 DDR 中取数据并显示在屏幕上。所以死机时可以直接观察显示的情况来初步判断 DDR 这时候的状态。

    1. 如果显示正常的话，说明这时候 DDR 的是能够正常访问的， 但是并不能说明死机和 DDR 无关。
    2. 如果显示异常。
        - 如下图，我们称为“花屏”，有可能是 DDR 变频的过程中死机了导致 DDR 处于不可访问状态，这时候可以定频试试。或者可能是电源问题导致 DDR 控制器逻辑异常。

- 如下图，我们称为“重影”，之前遇到类似的情况是由于板子参考层不完整，导致 DDR3 异常。可尝试提高 VCC_DDR 电压到 1.6V 或者将颗粒的 dll bypass 掉解决。

4. 排查是否是电源问题
    1. 固定 cpu/gpu 到一个较低的频率，适当提高 arm/logic 电压看看是否有改善。有改善的话可能是电源问题。
    2. 审核 layout 看是否电源上存在问题。
    3. 测量电源纹波是否存在问题。
5. 排查是否信号质量问题
    1. 降低 DDR 频率看看是否有明显改善，有改善的话很可能就是信号质量问题。
    2. 让硬件同事审核 layout 和 gerber 文件，检查走线是否合理，参考层是否完整。
    3. 适当加强减弱驱强度/odt 强度，看是否有改善。
    4. 改变 RZQ 的阻值看看是否有改善。遇到过个别 220ball 的 lpddr3 需要将 RZQ 改小或者去掉才能够恢复正常。
6. 对于白牌颗粒

对于白牌颗粒，如果排查过电源，信号质量等都没问题的情况下只能怀疑可能是存储单元有问题，可以尝试现有遇到过的白牌颗粒的处理方法。

1. 尝试关闭 pd_idle，sr_idle 看看是否有效果。
2. 对于一些死机时屏幕有“重影”的颗粒可以尝试 bypass DRAM DLL 看是否有效果。
3. 一些存储单元有问题的颗粒可以通过 DDR 测试工具测试出来， 目前遇到的比较多的是 DDR 测试工具 March 专项能测出来的概率比较大。
   需要注意的是 DDR 测试工具仅仅是作为一个辅助工具，测试工具测试 pass 并不代表颗粒或者板子稳定性一定没问题。

3. Chapter-3 DDR 颗粒验证流程说明

NOTE

1. RV1108、RK3308 平台 DDR 颗粒验证流程与其它平台不同，RV1108 请详见本文档的"RV1108 DDR 颗粒验证流程说明"章节；RK3308 请详见本文档的"RK3308 DDR 颗粒验证流程说明"章节。其他平台，请根据 Linux Kernel 版本选择对应章节进行参考。
2. 本文中所述颗粒验证过程需要的 DDR 测试资源文件随该文档提供。

3.1 Linux 3.10 DDR 颗粒验证流程说明

3.1.1 Linux 3.10 测试固件编译

配置 kernel 代码的 menuconfig，进入 System Type，选择打开 DDR Test 和 pm_tests。

```
menuconfig
System Type  --->
  [*]   /sys/pm_tests/ support
  [*]   DDR Test
```

如果 menuconfig 中没有`[] /sys/pm_tests/ support`选项，请参考《Rockchip-Developer-Guide-DDR-CN》的"DDR 如何定频"和"如何 enable/disable kernel 中的 DDR 变频功能"章节，分别编译定频固件和变频固件。

3.1.2 Linux 3.10 测试环境搭建

3.1.2.1 固件烧写

测试开始前，需要先明确测试过程需要的如下信息：

1. 测试固件操作系统位数（eg: 32bit or 64bit）
2. 测试机器 DDR 总容量（eg: 512MB or 1GB or 2GB ...）
3. 定频测试，DDR 要跑的最高频率（eg: 456MHz or 533MHz ...）
4. 变频测试，DDR 要跑的频率范围（eg: 200MHz - 456MHz or 200MHz-533MHz ...）

3.1.2.2 自动搭建测试环境

进入 DDR 测试资源文件的"linux3.10_ddr_test_files"目录，直接双击 push_files.bat 脚本，根据脚本提示和固件类型信息进行选择输入 1 或者 2，自动完成测试环境搭建。自动搭建无异常，可以跳过下面的"手动搭建测试环境"这一章节。

3.1.2.3 手动搭建测试环境

如果自动搭建测试环境失败，可以通过手动搭建来完成。请选择"linux3.10_ddr_test_files"目录里的测试文件进行安装。

1. 安装捕鱼达人APK

Note: 捕鱼达人为第三方APK，请自行获取，也可选择其他能在测试期间自动运行的APK或者循环播放视频。

```
<adb_tool> adb.exe install fishingjoy1.apk
```

2. push google stressapptest

请根据测试机器的固件为 linux 64bit 或 linux 32bit，在 “static_stressapptest” 目录下选择对应的 stressapptest 进行 push。

Eg:

- 如测试机器的固件为 linux 32bit，则选择对应的 stressapptest_32bit

```
<adb_tool> adb.exe push stressapptest_32bit /data/stressapptest
<adb_tool> adb.exe shell chmod 0777 /data/stressapptest
```

- 如测试机器的固件为 linux 64bit，则选择对应的 stressapptest_64bit

```
<adb_tool> adb.exe push stressapptest_64bit /data/stressapptest
<adb_tool> adb.exe shell chmod 0777 /data/stressapptest
```

3. push memtester

请根据测试机器的固件为 linux 64bit 或 linux 32bit，在 “static_memtester” 目录下选择对应的 memtester 进行 push。
Eg：

- 如测试机器的固件为 linux 32bit，则选择对应的 memtester_32bit

```
<adb_tool> adb.exe push memtester_32bit /data/memtester
<adb_tool> adb.exe shell chmod 0777 /data/memtester
```

- 如测试机器的固件为 linux 64bit，则选择对应的 memtester_64bit

```
<adb_tool> adb.exe push memtester_64bit /data/memtester
<adb_tool> adb.exe shell chmod 0777 /data/memtester
```

4. sync

```
<adb_tool> adb.exe shell sync
```

3.1.3 Linux 3.10 确认容量是否正确

通过`<rkxxxx:/ $>  cat /proc/meminfo`查看 MemTotal 容量是否与测试机器实际容量相符。

log eg：

```
<rkxxxx:/ $> cat /proc/meminfo
MemTotal:        2038904 kB
```

512MB 约等于 533504kB

1GB 约等于 1048576kB

1.5GB 约等于 1582080kB

2GB 约等于 2097152kB

3GB 约等于 3145728kB

4GB 约等于 4194304kB

由于系统内存分配管理差异的原因，得到的容量有些偏差，属于正常。

3.1.4 Linux 3.10 定频测试

1. 开启捕鱼达人 APK
2. 串口控制台上输入 su 命令

```
<rkxxxx:/ $> su
```

3. DDR 定频到拷机频率

请根据测试机器所支持的 DDR 最高频率，进行设置。

Eg：

如果测试机器所支持的 DDR 最高频率为 533MHz。

```
<rkxxxx:/ #> echo set clk_ddr 533000000 > /sys/pm_tests/clk_rate
```

4. google stressapptest 拷机，拷机时间 12 小时以上

- 执行 stressapptest 程序

stressapptest 关键参数说明：

-M mbytes，指定申请测试的内存空间大小，单位为MB。一般申请总容量的八分之一进行测试，如果总容量是 1GB 则申请 128MB 进行 测试，如果总容量是 2GB 则申请 256MB 进行测试。

-s seconds，指定测试运行时间，单位是秒。运行时间12小时则参数为43200。

Eg：

内存总容量 1GB 则申请 128MB 进行 stressapptest，运行时间12小时(43200秒)，执行命令如下：

```
<rkxxxx:/ #> /data/stressapptest -s 43200 -i 4 -C 4 -W --stop_on_errors -M 128
```

- 确认拷机结果

拷机结束，确认机器是否正常，捕鱼达人是否正常运行，stressapptest 结果是 PASS 还是 FAIL。stressapptest 每隔 10 秒会打印一条 log，log 显示测试剩余时间。测试完成后会打印测试结果，如果测试通过打印 "Status: PASS - please verify no corrected errors"，如果测试失败打印 "Status: FAIL - test discovered HW problems"。

6. memtester 拷机，拷机时间 memtester 12 小时以上

- 执行 memtester 程序

memtester 关键参数说明：

<mem>m，指定申请测试的内存空间大小，单位为 MB。一般申请总容量的八分之一进行测试，如果总容量是 1GB 则申请 128MB 进行 测试，如果总容量是 2GB 则申请 256MB 进行测试。

Eg:

内存总容量 1GB 则申请 128MB 进行 memtester，执行命令如下：

```
<rkxxxx:/ #> /data/memtester 128m
```

- 确认拷机结果

拷机结束，确认机器是否正常，捕鱼达人是否正常运行，memtester 是否在正常运行。DDR 测试资源文件目录里的 memtester 程序进行过修改，测试过程如果有发现错误会自动停止测试，如果持续测试 12 小时以上后，memtester 仍在继续运行，说明测试过程没有发现错误。

- memtester 运行过程如果没有发现错误，会持续打印如下 log：

```
Loop 10:
  Stuck Address       : ok
  Random Value        : ok
  Compare XOR         : ok
  Compare SUB         : ok
  Compare MUL         : ok
  Compare DIV         : ok
  Compare OR          : ok
  Compare AND         : ok
  Sequential Increment: ok
  Solid Bits          : ok
  Block Sequential    : ok
  Checkerboard        : ok
  Bit Spread          : ok
  Bit Flip            : ok
  Walking Ones        : ok
  Walking Zeroes      : ok
```

- memtester 运行过程如果有发现错误，会自动停止测试并退出，退出时打印如下 log：

```
FAILURE: 0xffffffff != 0xffffbfff at offset 0x03b7d9e4.
EXIT_FAIL_OTHERTEST
```

3.1.5 Linux 3.10 变频测试

如果机器前面做过定频测试，要重启机器，否则后续的变频命令会无法进行。

1. 开启捕鱼达人 APK
2. 串口控制台上输入 su 命令

```
<rkxxxx:/ $> su
```

3. 后台执行 memtester

memtester 关键参数说明：

<mem>m，指定申请测试的内存空间大小，单位为MB。一般申请总容量的八分之一进行测试，如果总容量是 1GB 则申请 128MB 进行测试，如果总容量是 2GB 则申请 256MB 进行测试。

Eg:

如果总容量是 1GB 则申请 128MB 进行 memtester，执行命令如下：

```
<rkxxxx:/ #> /data/memtester 128m > /data/memtester_log.txt &
```

4. 执行变频命令

根据测试机器支持的 DDR 频率范围进行设置。

Eg：

如果测试机器支持的 DDR 频率范围为 200MHz 到 533MHz。

```
<rkxxxx:/ #> echo 'a:200M-533M-1000000T' > proc/driver/ddr_ts
```

Note: 变频测试测试过程，由于是强制变频，机器可能会出现由于带宽不足等原因所致的屏幕闪烁等现象，属正常现象。

5. 确认拷机结果，拷机时间 12 小时以上

- 确认捕鱼达人是否正常运行，机器是否正常
- 确认变频程序运行是否正常，变频 log 是否在正常打印
- 确认 memtester 是否正常运行

在串口输入`<rkxxxx:/ #> ps | grep memtester`，看 memtester 进程是否存在。

Eg：

```
<rkxxxx:/ #>  ps | grep memtester
root      14309 1730  74332  68156           0 5e980bf564 R /data/memtester
```

3.1.6 Linux 3.10 reboot 拷机

开启计算器 Calculator，输入 "83991906="，点击"RebootTest"，拷机时间 12 小时以上。

3.2 Linux 4.xx DDR 颗粒验证流程说明

3.2.1 Linux 4.xx 测试固件编译

使能 DDR 变频功能，打开测试机器对应的板级 DTS 文件，找到 dfi 和 dmc 节点，配置 status = "okay"。

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

&dmc {
    status = "okay";
    ...
};
```

关于测试固件编译这里只做简单说明，详细介绍请参考《Rockchip-Developer-Guide-DDR-CN》的"如何enable/disable kernel 中的 DDR 变频功能"章节。

3.2.2 Linux 4.xx 测试环境搭建

3.2.2.1 烧写固件

测试开始前，需要先明确测试过程需要的如下信息：

1. 测试固件操作系统位数（32bit or 64bit）
2. 测试机器 DDR 总容量（512MB or 1GB or 2GB ...）
3. 定频测试，DDR 要跑的最高频率（456MHz or 533MHz ...）

3.2.2.2 自动搭建测试的环境

1. 进入 DDR 测试资源文件的"linux4.xx_ddr_test_files"目录，双击 push_files.bat 脚本文件，根据脚本提示和固件类型信息进行选择输入 1 或者 2，自动完成测试环境搭建。

Note：运行脚本后，需要通过打印的 log 检查是否每项都有被正常执行，确认是否有报错信息。

2. 自动搭建无异常，可以跳过下面的"手动搭建测试环境"这一章节。

3.2.2.3 手动搭建测试的环境

如果自动搭建测试环境失败，可以通过手动搭建来完成。请选择"linux4.xx_ddr_test_files"目录里的测试文件进行安装。

1. 安装捕鱼达人APK

Note：捕鱼达人为第三方 APK，请自行获取，也可选择其他能在测试期间自动运行的APK或者循环播放视频。未运行 Android 系统的设备请跳过此步骤。

```
<adb_tool> adb.exe install fishingjoy1.apk
```

2. push google stressapptest

请根据测试机器的固件为 linux 64bit 或 linux 32bit，在 “static_stressapptest” 目录下选择对应的 stressapptest 进行 push。

Eg:

- 如测试机器的固件为 linux 32bit，则选择对应的 stressapptest_32bit

```
<adb_tool> adb.exe push stressapptest_32bit /data/stressapptest
<adb_tool> adb.exe shell chmod 0777 /data/stressapptest
```

- 如测试机器的固件为 linux 64bit，则选择对应的 stressapptest_64bit

```
<adb_tool> adb.exe push stressapptest_64bit /data/stressapptest
<adb_tool> adb.exe shell chmod 0777 /data/stressapptest
```

3. push memtester

请根据测试机器的固件为 linux 64bit 或 linux 32bit，在 “static_memtester” 目录下选择对应的 memtester 进行 push。

Eg：

- 如测试机器的固件为 linux 32bit，则选择对应的 memtester_32bit

```
<adb_tool> adb.exe push memtester_32bit /data/memtester
<adb_tool> adb.exe shell chmod 0777 /data/memtester
```

- 如测试机器的固件为 linux 64bit，则选择对应的 memtester_64bit

```
<adb_tool> adb.exe push memtester_64bit /data/memtester
<adb_tool> adb.exe shell chmod 0777 /data/memtester
```

4. push ddr_freq_scan.sh 脚本

```
<adb_tool> adb.exe push ddr_freq_scan.sh /data/ddr_freq_scan.sh
<adb_tool> adb.exe shell chmod 0777 /data/ddr_freq_scan.sh
```

5. sync

```
<adb_tool> adb.exe shell sync
```

3.2.2.4 通过 U 盘和串口搭建测试环境

如果测试机器的 ADB 无法连接，可以通过 U 盘将测试过程需要用到文件拷贝到测试板，然后通过串口搭建测试环境。

1. 准备工作

- 开机后，添加 wake_lock 防止机器进入二级待机`echo 1 > /sys/power/wake_lock`，或者通过设置 Setting->Dsiplay->Sleep->Never sleep 让机器保持唤醒状态
- U 盘接入电脑，将"linux4.xx_ddr_test_files" ， “static_memtester” 和 “static_stressapptest” 目录拷贝到 U 盘
- 测试板串口控制台上输入 su 命令

```
<rk3399:/ $> su
```

- U 盘接入测试板，将 "linux4.xx_ddr_test_files" ， “static_memtester” 和 “static_stressapptest” 目录拷贝到机器 /data 目录。U 盘正常是加载在/mnt/media_rw/***目录下（***：每块 U 盘的挂载节点名有不同，请用 tab 键补全）。

Eg:

如果U盘的挂载节点为/mnt/media_rw/B4FE-5315，执行命令如下：

```
<rk3399:/ #> cp -rf /mnt/media_rw/B4FE-5315/linux4.xx_ddr_test_files   /data/
<rk3399:/ #> cp -rf /mnt/media_rw/B4FE-5315/static_memtester   /data/
<rk3399:/ #> cp -rf /mnt/media_rw/B4FE-5315/static_stressapptest   /data/
```

2. 自动搭建测试环境

- 如测试机器的固件为 linux 32bit，则选择对应的 test_files_install_32bit.sh 脚本

```
<rk3399:/ #> chmod 0777
/data/linux4.xx_ddr_test_files/test_files_install_32bit.sh
<rk3399:/ #> /data/linux4.xx_ddr_test_files/test_files_install_32bit.sh
```

- 如测试机器的固件为 linux 64bit，则选择对应的 test_files_install_64bit.sh 脚本

```
<rk3399:/ #> chmod 0777
/data/linux4.xx_ddr_test_files/test_files_install_64bit.sh
<rk3399:/ #> /data/linux4.xx_ddr_test_files/test_files_install_64bit.sh
```

需要通过打印的 log 检查是否每项都有被正常执行。自动搭建无异常，可以跳过下面的"手动搭建测试环境"这一部分。

3. 手动搭建测试环境

如果自动搭建测试环境失败，可以通过手动搭建来完成。

- 如测试机器的固件为 linux 32bit，则拷贝对应的测试所需文件

```
<rk3399:/ #> cp /data/static_stressapptest/stressapptest_32bit
/data/stressapptest
<rk3399:/ #> cp /data/static_memtester/memtester_32bit /data/memtester
<rk3399:/ #> cp /data/linux4.xx_ddr_test_files/ddr_freq_scan.sh
/data/ddr_freq_scan.sh
```

- 如测试机器的固件为 linux 64bit，则拷贝对应的测试所需文件

```
<rk3399:/ #> cp /data/static_stressapptest/stressapptest_64bit
/data/stressapptest
<rk3399:/ #> cp /data/static_memtester/memtester_64bit /data/memtester
<rk3399:/ #> cp /data/linux4.xx_ddr_test_files/ddr_freq_scan.sh
/data/ddr_freq_scan.sh
```

- 更改文件权限

```
<rk3399:/ #> chmod 777 /data/memtester /data/stressapptest
/data/ddr_freq_scan.sh
```

- 安装捕鱼达人 APK

Note：捕鱼达人为第三方APK，请自行获取，也可选择其他能在测试期间自动运行的APK或者循环播放视频。未运行 Android 系统的设备请跳过此步骤。

```
<rk3399:/ #> pm install /data/fishingjoy1.apk
```

- sync

```
<rk3399:/ #> sync
```

3.2.3 Linux 4.xx 确认颗粒容量

通过`<rkxxxx:/ #>  cat /proc/meminfo`查看 MemTotal 项所示容量是否与测试机器 DDR 总容量一致。

log eg：

```
rkxxxx:/ # cat /proc/meminfo
MemTotal:        2038904 kB
```

512MB 约等于 533504kB

1GB 约等于 1048576kB

1.5GB 约等于 1582080kB

2GB 约等于 2097152kB

3GB 约等于 3145728kB

4GB 约等于 4194304kB

由于系统内存分配管理差异的原因，得到的容量有些偏差，属于正常。

3.2.4 Linux 4.xx 定频拷机

1. 开启捕鱼达人 APK（未运行 Android 系统的设备请跳过此步骤）
2. 先输入 su 命令

```
<rkxxxx:/ $> su
```

3. 定频到拷机频率

根据测试机器 DDR 要跑的最高频率进行设置。

Eg：

- 如果是跑 928MHz

```
<rkxxxx:/ #> /data/ddr_freq_scan.sh 933000000
```

- 如果是跑 800MHz

```
<rkxxxx:/ #> /data/ddr_freq_scan.sh 800000000
```

- 如果是跑 600MHz

```
<rkxxxx:/ #> /data/ddr_freq_scan.sh 600000000
```

4. 通过 log 确认频率是否正确

log eg:

```
130|rkxxxx:/ # /data/ddr_freq_scan.sh 800000000
already change to 800000000 done.
change frequency to available max frequency done.
```

5. google stressapptest 拷机，拷机时间 12 小时以上

- 执行 stressapptest 程序

stressapptest 关键参数说明：

-M mbytes，指定申请测试的内存空间大小，单位为MB。一般申请总容量的八分之一进行测试，如果总容量是 1GB 则申请 128MB 进行 测试，如果总容量是 2GB 则申请 256MB 进行测试。

-s seconds，指定测试运行时间，单位是秒。运行时间12小时则参数为43200。

Eg：

内存总容量 1GB 则申请 128MB 进行 stressapptest，运行时间12小时(43200秒)，执行命令如下：

```
<rkxxxx:/ #> /data/stressapptest -s 43200 -i 4 -C 4 -W --stop_on_errors -M 128
```

- 确认拷机结果

拷机结束，确认机器是否正常，捕鱼达人是否正常运行，stressapptest 结果是 PASS 还是 FAIL。stressapptest 每隔 10 秒会打印一条 log，log 显示测试剩余时间。测试完成后会打印测试结果，如果测试通过打印 Status: PASS - please verify no corrected errors，如果测试失败打印 Status: FAIL - test discovered HW problems。

6. memtester 拷机，拷机时间 12 小时以上

- 执行 memtester 程序

memtester 关键参数说明：

<mem>m，指定申请测试的内存空间大小，单位为MB。一般申请总容量的八分之一进行测试，如果总容量是 1GB 则申请 128MB 进行 测试，如果总容量是 2GB 则申请 256MB 进行测试。

Eg:

内存总容量 1GB 则申请 128MB 进行 memtester，执行命令如下：

```
<rkxxxx:/ #> /data/memtester 128m
```

- 确认拷机结果

拷机结束，确认机器是否正常，捕鱼达人是否正常运行，memtester 是否在正常运行。DDR 测试资源文件目录里的 memtester 程序进行过修改，测试过程如果有发现错误会自动停止测试，如果持续测试 12 小时以上后，memtester 仍在继续运行，说明测试过程没有发现错误。

- memtester 运行过程如果没有发现错误，会持续打印如下 log：

```
Loop 10:
  Stuck Address       : ok
```

```
  Random Value        : ok
  Compare XOR         : ok
  Compare SUB         : ok
  Compare MUL         : ok
  Compare DIV         : ok
  Compare OR          : ok
  Compare AND         : ok
  Sequential Increment: ok
  Solid Bits          : ok
  Block Sequential    : ok
  Checkerboard        : ok
  Bit Spread          : ok
  Bit Flip            : ok
  Walking Ones        : ok
  Walking Zeroes      : ok
```

- memtester 运行过程如果有发现错误，会自动停止测试并退出，退出时打印如下 log：

```
FAILURE: 0xffffffff != 0xffffbfff at offset 0x03b7d9e4.
EXIT_FAIL_OTHERTEST
```

3.2.5 Linux 4.xx 变频拷机

1. 开启捕鱼达人 APK（未运行 Android 系统的设备请跳过此步骤）
2. 先输入 su 命令

```
<rkxxxx:/$> su
```

3. 后台执行 memtester

memtester 关键参数说明：

<mem>m，指定申请测试的内存空间大小，单位为MB。一般申请总容量的八分之一进行测试，如果总容量是 1GB 则申请 128MB 进行 测试，如果总容量是 2GB 则申请 256MB 进行测试。

Eg:

如果总容量是 1GB 则申请 128MB 进行 memtester，执行命令如下：

```
<rkxxxx:/ #> /data/memtester 128m > /data/memtester_log.txt &
```

4. 执行测试脚本

```
<rkxxxx:/ #> /data/ddr_freq_scan.sh
```

Note: 变频测试测试过程，由于是强制变频，机器可能会出现由于带宽不足等原因所致的屏幕闪烁等现象，属正常现象。

5. 确认拷机结果，拷机时间 12 小时以上

- 确认捕鱼达人是否正常运行，机器是否正常
- 确认 memtester 是否正常运行

在串口输入`<rkxxxx:/ #> ps | grep memtester`，看 memtester 进程是否存在。

Eg：

```
<rkxxxx:/ #>  ps | grep memtester
root      14309 1730  74332  68156           0 5e980bf564 R /data/memtester
```

- 确认变频脚本运行是否正常运行，变频 log 是否在正常打印

log eg:

```
DDR freq will change to 600000000 8786
already change to 600000000 done
DDR freq will change to 800000000 8787
already change to 800000000 done
DDR freq will change to 200000000 8788
already change to 200000000 done
```

3.2.6 Linux 4.xx reboot 拷机

3.2.6.1 Android 系统下通过计算器开启 reboot 拷机

==为防止做 reboot 过程机器进入休眠，影响测试，请通过设置 setting->security->set screen lock->None，让机器一开机跳过锁屏界面，直接进入主界面。同时通过设置 Setting->Dsiplay->Sleep->Never sleep 让机器保持唤醒状态。==

开启计算器 Calculator，输入 "83991906="，点击"RebootTest"，拷机时间 12 小时以上。

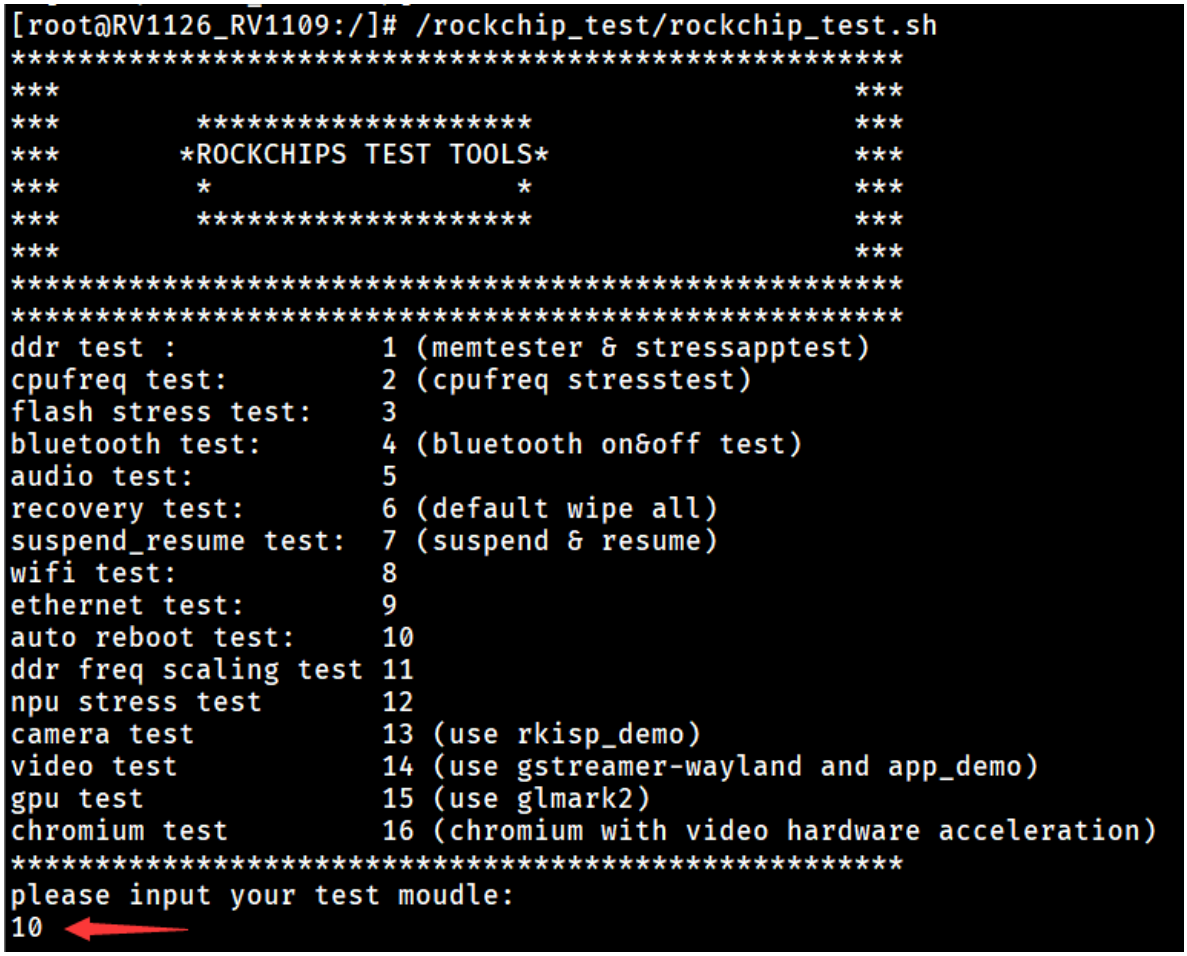
3.2.6.2 非 Android 系统使用 rockchip_test.sh 脚本进行 reboot 拷机

1. 输入 su 命令

```
[root@RV1126_RV1109:/]# su
```

2. 运行 /rockchip_test/rockchip_test.sh，选择 auto reboot test 对应的数字

```
[root@RV1126_RV1109:/]# /rockchip_test/rockchip_test.sh
```



3. 拷机 12 小时以上，确认设备是否正常。可以通过以下命令关闭 reboot test

```
[root@RV1126_RV1109:/]# echo off > /data/cfg/rockchip_test/reboot_cnt
```

3.2.7 Linux 4.xx sleep 拷机

RK3399 使用 LPDDR4 时，必须进行这项测试；其他 DDR 类型及其他平台是 optional。

3.2.7.1 Android 系统下通过计算器开启 sleep 拷机

拔掉连接 ADB 的 USB 线，开启计算器 Calculator,输入 "83991906=",点击"SleepTest"，拷机时间 12 小时以上。

3.2.7.2 非 Android 系统使用 rockchip_test.sh 脚本进行 sleep 拷机

1. 输入 su 命令

```
[root@RV1126_RV1109:/]# su
```

2. 运行 /rockchip_test/rockchip_test.sh，选择 suspend_resume test 对应的数字，再选择 auto suspend (resume by rtc) 对应的数字

```
[root@RV1126_RV1109:/]# /rockchip_test/rockchip_test.sh
```

```
[root@RV1126_RV1109:/]# /rockchip_test/rockchip_test.sh
*******************************************************
***                                                 ***
***        ********************                     ***
***       *ROCKCHIPS TEST TOOLS*                    ***
***        *                  *                     ***
***        ********************                     ***
***                                                 ***
*******************************************************
*******************************************************
ddr test :              1 (memtester & stressapptest)
cpufreq test:           2 (cpufreq stresstest)
flash stress test:      3
bluetooth test:         4 (bluetooth on&off test)
audio test:             5
recovery test:          6 (default wipe all)
suspend_resume test:    7 (suspend & resume)
wifi test:              8
ethernet test:          9
auto reboot test:       10
ddr freq scaling test   11
npu stress test         12
camera test             13 (use rkisp_demo)
video test              14 (use gstreamer-wayland and app_demo)
gpu test                15 (use glmark2)
chromium test           16 (chromium with video hardware acceleration)
*******************************************************
please input your test moudle:
7
************************************
auto suspend:                       1
suspend (resume by key):            2
auto suspend (resume by rtc):       3
************************************
3
```

3. 拷机 12 小时以上，确认设备是否正常

3.3 RV1108 DDR 颗粒验证流程说明

3.3.1 RV1108 测试固件编译

打开 DDR Test 和 pm_tests 配置。配置 kernel 代码的 menuconfig，进入 System Type，选择打开 DDR Test 和 pm_tests。

```
menuconfig
System Type  --->
  [*]   /sys/pm_tests/ support
  [*]   DDR Test
```

如果 menuconfig 中没有`[] /sys/pm_tests/ support`选项，请参考《Rockchip-Developer-Guide-DDR-CN》的"DDR 如何定频"和"如何 enable/disable kernel 中的 DDR 变频功能"章节，分别编译定频固件和变频固件。

3.3.2 RV1108 测试环境搭建

将"rv1108_ddr_test_files"目录下的 stressapptest 和 memtester_32bit 考到 SD 卡根目录，把卡插到测试板上。

3.3.3 RV1108 确认容量是否正确

通过`<rv1108:/ #>  cat /proc/meminfo`查看 MemTotal 项所示容量是否与测试机器 DDR 总容量相符。

log eg：

```
<rv1108:/ #> cat /proc/meminfo
MemTotal:         133376 kB
```

64MB 约等于 66688kB

128MB 约等于 133376kB

256MB 约等于 266752kB

512MB 约等于 533504kB

由于系统内存分配管理差异的原因，得到的容量有些偏差，属于正常。

3.3.4 RV1108 定频测试

1. DDR 定频到拷机频率

请根据测试机器要跑的 DDR 最高频率，进行设置。

Eg：

如果测试机器要跑的 DDR 最高频率为 800MHz。

```
<rv1108:/ #> echo set clk_ddr 800000000 > /sys/pm_tests/clk_rate
```

2. google stressapptest 拷机，拷机时间 12 小时以上

如果总容量是 128MB 则申请 16MB 进行 stressapptest，一般是总容量的八分之一

```
<rv1108:/ #> /mnt/sdcard/stressapptest -s 500 -i 1 -C 1 -W --stop_on_errors -M
16
```

3. 确认拷机结果

拷机结束，确认机器是否正常，stressapptest 结果是 PASS 还是 FAIL。stressapptest 每隔 10 秒会打印一条 log，log 显示测试剩余时间。测试完成后会打印测试结果，如果测试通过打印 Status: PASS，如果测试失败打印 Status: FAIL。

4. memtester 拷机，拷机时间 12 小时以上

如果总容量是 128MB 则申请 16MB 进行 memtester，一般是总容量的八分之一

```
<rv1108:/ #> /mnt/sdcard/memtester_32bit 16m
```

5. 确认拷机结果

拷机结束，确认机器是否正常，memtester 是否正常运行。"DDR 测试资源文件"目录里的 memtester 程序进行过修改，测试过程如果有发现错误会自动停止测试，如果持续测试 12H+后，memtester 仍在继续运行，说明测试过程没有发现错误。

- memtester 运行过程如果没有发现错误，会持续打印如下 log：

```
Loop 10:
  Stuck Address       : ok
  Random Value        : ok
  Compare XOR         : ok
  Compare SUB         : ok
  Compare MUL         : ok
  Compare DIV         : ok
  Compare OR          : ok
  Compare AND         : ok
  Sequential Increment: ok
  Solid Bits          : ok
  Block Sequential    : ok
  Checkerboard        : ok
  Bit Spread          : ok
  Bit Flip            : ok
  Walking Ones        : ok
  Walking Zeroes      : ok
```

- memtester 运行过程如果有发现错误，会自动停止测试并退出，退出时打印如下 log：

```
FAILURE: 0xffffffff != 0xffffbfff at offset 0x03b7d9e4.
EXIT_FAIL_OTHERTEST
```

3.3.5 RV1108 变频测试

如果机器前面做过定频测试，要重启机器，否则后续的变频命令会无法进行。

1. 后台执行 memtester

如果总容量是 128MB 则申请 16MB 进行 memtester，一般是总容量的十六分之一

```
<rv1108:/ #> /mnt/sdcard/memtester_32bit 16m > /data/memtester_log.txt &
```

2. 执行变频命令

变频测试频率范围为 400MHz 到测试机器要跑的 DDR 最高频率之间进行。

Eg：

如果测试机器 DDR 要跑的最高频率为 800MHz：

```
<rv1108:/ #> echo 'a:400M-800M-1000000T' > proc/driver/ddr_ts
```

3. 确认拷机结果，拷机时间 12 小时以上

- 确认机器是否正常
- 确认变频程序运行是否正常，变频 log 是否在正常打印
- 确认 memtester 是否正常运行

在串口输入`<rkxxxx:/ #> ps | grep memtester`，看 memtester 进程是否存在。

Eg：

```
<rkxxxx:/ #>  ps | grep memtester
root      14309 1730  74332  68156           0 5e980bf564 R /data/memtester
```

3.3.6 RV1108 reboot 拷机

可用 1108 自带 reboot 功能，menu -> debug -> reboot test。

3.4 RK3308 DDR 颗粒验证流程说明

3.4.1 RK3308 确定容量是否正确

通过`<rk3308:/ #>  cat /proc/meminfo`查看 MemTotal 项所示容量是否与测试机器 DDR 总容量相符。

log eg：

```
<rk3308:/ #> cat /proc/meminfo
MemTotal:         246832 kB
MemFree:          201800 kB
```

64MB 约等于 66688kB

128MB 约等于 133376kB

256MB 约等于 266752kB

512MB 约等于 533504kB

由于系统内存分配管理差异的原因，得到的容量有些偏差，属于正常。

3.4.2 RK3308 拷机测试

由于 3308 当前不支持变频功能，ddr 频率由 loader 初始化期间设置的频率，到后面是不会修改 ddr 频率的。DDR 颗粒测试请使用最高频率的设置，即 DDR3请使用800MHz 的 loader，DDR2和LPDDR2请使用533MHz 的 loader。

1. memtester 拷机，拷机时间 12 小时以上

首先要确认测试文件是否存在：

```
<rk3308:/ #> ls usr/bin/memtester
usr/bin/memtester
```

- 如果有测试文件，memtester 测试命令如下：（如果总容量是 128MB 则申请 16MB 进行 memtester，一般是总容量的八分之一。）

```
<rk3308:/ #> memtester 16m > /data/memtester_log.txt &
```

- 如果没有测试文件，则请将随此文件一同发布的 memtester_32bit.32bit（或 memtester_64bit.64bit）文件通过 adb push 到 data 分区：（memtester_32bit.32bit 适用于 32 位系统，memtester_64bit.64bit 适用于 64 位系统）

32 位系统命令：

```
adb push \*文件路径*\memtester_32bit.32bit data/memtester
```

64 位系统命令：

```
adb push \*文件路径*\memtester_64bit.64bit data/memtester
```

将 memtester_32bit.32bit 文件修改权限为可执行：

```
<rk3308:/ #> chmod 777 /data/memtester
```

memtester 测试命令如下（如果总容量是 128MB 则申请 16MB 进行 memtester，一般是总容量的八分之一）：

```
<rk3308:/ #> /data/memtester 16m > /data/memtester_log.txt &
```

2. 确认拷机结果

- 拷机结束，确认机器是否正常。
- 确认 memtester 打印是否正常：（注意，memtester 出错==不会==停止测试，需要查看所有打印是否正确）

正确打印：

```
Loop 1:
  Stuck Address       : ok
  Random Value        : ok
  Compare XOR         : ok
  Compare SUB         : ok
  Compare MUL         : ok
  Compare DIV         : ok
  Compare OR          : ok
  Compare AND         : ok
  Sequential Increment: ok
  Solid Bits          : ok
  Block Sequential    : ok
  Checkerboard        : ok
  Bit Spread          : ok
  Bit Flip            : ok
  Walking Ones        : ok
  Walking Zeroes      : ok
  8-bit Writes        : ok
  16-bit Writes       : ok
```

出错打印：

```
Loop 92:
  Stuck Address       : ok
```

```
  Random Value        : FAILURE: 0x37fe0f4190f6b999 != 0x37fe0f4196f6b999 at
offset 0x0027a958.
FAILURE: 0x2823d0d6f62a4b01 != 0x2823d0d6f02a4b01 at offset 0x0027a958.
  Compare XOR         : FAILURE: 0x4c4f418e764340e8 != 0x4c4f418e704340e8 at
offset 0x0027a958.
  Compare SUB         : FAILURE: 0x2856fb8ee22bd230 != 0xfe5ee503ee2bd230 at
offset 0x0027a958.
  Compare MUL         : FAILURE: 0x00000000 != 0x00000001 at offset 0x0027a958.
  Compare DIV         : FAILURE: 0xefb2eceaffbf9604 != 0xefb2eceaffbf9605 at
offset 0x0027a958.
  Compare OR          : FAILURE: 0xcbb2a0e8cfbb8200 != 0xcbb2a0e8cfbb8201 at
offset 0x0027a958.
  Compare AND         :   Sequential Increment: ok
  Solid Bits          : ok
  Block Sequential    : ok
  Checkerboard        : ok
  Bit Spread          : ok
  Bit Flip            : ok
  Walking Ones        : ok
  Walking Zeroes      : ok
  8-bit Writes        : ok
  16-bit Writes       : ok
```

3. google stressapptest 拷机，拷机时间 12 小时以上

首先要确认测试文件是否存在：

```
<rk3308:/ #> ls usr/bin/stressapptest
usr/bin/stressapptest
```

- 如果有测试文件，stressapptest 测试命令如下：（如果总容量是 256MB 则申请 32MB 进行 stressapptest，一般是总容量的八分之一。时间设置由-s 后面的参数控制，单位是秒。如下为拷机 24 小时的命令）

```
<rk3308:/ #> stressapptest -s 86400 -i 4 -C 4 -W --stop_on_errors -M 32
```

- 如果没有测试文件，则请将随此文件一同发布的 stressapptest_32bit（或 stressapptest_64bit） 文件通过 adb push 到 data 分区（stressapptest_32bit 适用于 32 位系统，stressapptest_64bit 适用于 64 位系统）：

32 位系统命令：

```
adb push \*文件路径*\stressapptest_32bit data/stressapptest
```

64 位系统命令：

```
adb push \*文件路径*\stressapptest_64bit data/stressapptest
```

将 stressapptest_32bit 文件修改权限为可执行：

```
<rk3308:/ #> chmod 777 /data/stressapptest
```

stressapptest 测试命令如下（如果总容量是 256MB 则申请 32MB 进行 stressapptest，一般是总容量的八分之一。时间设置由-s 后面的参数控制，单位是秒。如下为拷机 24 小时的命令）：

```
<rk3308:/ #> /data/stressapptest -s 86400 -i 4 -C 4 -W --stop_on_errors -M 32
```

4. 确认拷机结果

- 拷机结束，确认机器是否正常。
- stressapptest 结果是 PASS 还是 FAIL。stressapptest 每隔 10 秒会打印一条 log，log 显示测试剩余时间。测试完成后会打印测试结果，如果测试通过打印 Status: PASS，如果测试失败打印 Status: FAIL。

3.4.3 RK3308 休眠唤醒测试

休眠唤醒需要 kernel 使能自动唤醒功能。打开 rk3308.dtsi 文件，找到休眠唤醒节点 rockchip_suspend,位或上 RKPM_TIMEOUT_WAKEUP_EN，使能自动唤醒：

```
rockchip_suspend: rockchip-suspend {
                ...
                rockchip,wakeup-config = <
                        (0
                        | RKPM_GPIO0_WAKEUP_EN
                        | RKPM_TIMEOUT_WAKEUP_EN
                        )
                >;
        };
```

编译好固件后，建议使用 3308 测试脚本的休眠唤醒测试。首先要确认测试文件是否存在：

```
<rk3308:/ #> ls rockchip_test/rockchip_test.sh
rockchip_test/rockchip_test.sh
```

- 若有测试文件，则休眠自动唤醒测试命令如下：

```
<rk3308:/ #> /rockchip_test/rockchip_test.sh
...
please input your test moudle: //串口先输入8<enter>，再输入1<enter>
8
1
```

- 若没有测试文件，可以在串口命令行直接输入命令进行休眠唤醒测试，命令如下：

```
<rk3308:/ #> while true; do echo mem >  /sys/power/state; sleep 5; done
```

拷机时间 12h+，确认机器是否正常。

3.4.4 RK3308 reboot 拷机

建议使用 3308 测试脚本的 reboot 测试命令。首先要确认测试文件是否存在：

```
<rk3308:/ #> ls rockchip_test/rockchip_test.sh
rockchip_test/rockchip_test.sh
```

- 若有测试文件，reboot 测试命令如下：

```
<rk3308:/ #> /rockchip_test/rockchip_test.sh
...
please input your test moudle: //串口输入13<enter>
13
```

- 若无测试文件，则请将随此文件一同发布的 auto_reboot_test.sh 文件通过 adb push 到 data 分区：

```
adb push \*文件路径*\auto_reboot_test.sh data/.
```

将 auto_reboot_test.sh 文件修改权限为可执行：

```
<rk3308:/ #> chmod 777 /data/auto_reboot_test.sh
```

reboot 测试命令如下：

```
<rk3308:/ #> /data/auto_reboot_test.sh
```

拷机时间 12h+，确认机器是否正常。

4. Chapter-4 Rockchip DDR DQ 眼图工具指南

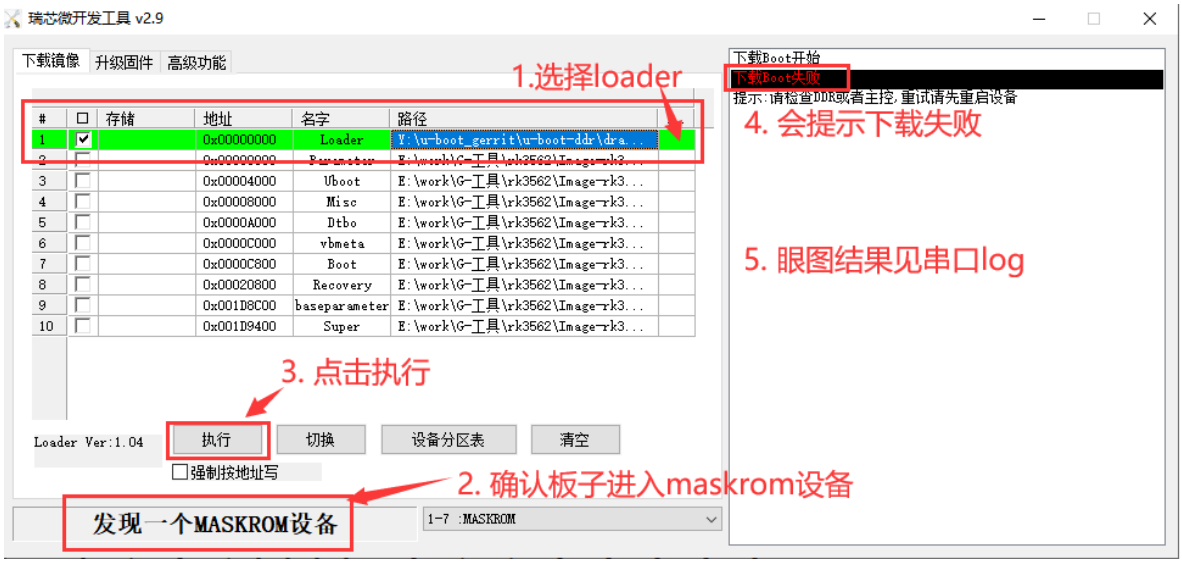
4.1 二维眼图的获取

4.1.1 支持的平台

RK3528、RK3562、RK3566、RK3568、RK3588、RK3576

4.1.2 使用方法

- Loader需要使用rkbin/bin/rkxx目录（与常规ddrbin同目录）下带eyescan后缀的ddrbin（如`rkbin/bin/rk35/rk3568_ddr_1560MHz_eyescan_v1.16.bin`）自行合成完整Loader使用。Loader的合成方法详见文档：Rockchip-Developer-Guide-DDR-CN.pdf。
- 如需要其他频率的ddrbin，可使用`rkbin/tools/ddrbin_tool`修改ddrbin频率，再合成Loader使用。ddrbin_tool的使用方法详见`rkbin/tools/ddrbin_tool_user_guide.txt`。
- 进入maskrom状态，通过下载工具下载loader到板子中。结果会通过串口输出。loader扫描完DDR 2D眼图后会自动停下，所以会提示下载失败。



- 如果是loader状态下烧写会烧写成功，每次开机都会扫描完2D眼图后停止，结果同样是通过串口输出。开机无法正常进入系统，需要进入maskrom设备烧写普通loader恢复。

4.1.2.1 输出结果分析

- 每个step单位在眼图log最开始处有打印出，单位为fs即0.001ps。例如下图第一行Unit：1913fs，则每个step为1.913ps。
- margin: rx:22-8,tx:26-32，代表read方向setup/hold time 至少需要22step/8step的margin。write方向setup/hold time 至少需要26step/32step的margin，该值已根据频率从ps换算为软件眼图上的step单位。如下图中使用'|'在rx dq0眼图上按照setup time 22step,hold time 8step圈出来的最小rx信号需要的建立保持时间需求。右侧的'[33 ~ -1 ~-35]'代表该vref下setup time有33step，hold time有35step。表明与rx最小需要22step/8step余量相比，setup time额外还有33-22=11step余量，hold time额外还有35-8=27step余量。

```
|Unit: 1913 fs
|margin: rx:22-8,tx:26-32
|scan ch[0]:
|cs0 result:
|rxvref:16.6%-28.5%-38.1%
|rx dq0, max_eye vref:28.2%[68]:
|   vref 37.0%:------------------********************----------------[18 ~ -1 ~ -21(39)]
|   vref 35.9%:----------------**********************----------------[22 ~ 0 ~ -23(45)]
|   vref 34.8%:---------------************************----------------[23 ~ 0 ~ -24(47)]
|   vref 33.7%:--------------**************************---------------[26 ~ 0 ~ -26(52)]
|   vref 32.6%:------------****************************--------------[29 ~ 0 ~ -28(57)]
|   vref 31.5%:------------****************************--------------[30 ~ 0 ~ -29(59)]
|   vref 30.4%:-----------******************************-------------[31 ~ 0 ~ -30(61)]
|  +vref 29.3%:-----------***|**********+**********|*****-------------[32 ~ 0 ~ -32(64)]
|* vref 28.2%:-----------****|**********+**********|******----------[33 ~ -1 ~ -35(68)]
|   vref 27.1%:------------***************************************----------[30 ~ -3 ~ -36(66)]
|   vref 26.0%:------------***************************************----------[28 ~ -3 ~ -35(63)]
|   vref 24.9%:--------------*************************************----------[25 ~ -4 ~ -34(59)]
|   vref 23.8%:--------------*************************************----------[24 ~ -4 ~ -33(57)]
|   vref 22.7%:----------------***********************************----------[22 ~ -5 ~ -32(54)]
|   vref 21.6%:------------------*********************************----------[19 ~ -5 ~ -30(49)]
|   vref 20.5%:--------------------*******************************----------[17 ~ -6 ~ -30(47)]
|   vref 19.4%:----------------------*****************************----------[15 ~ -7 ~ -29(44)]
|   vref 18.3%:------------------------***************************----------[10 ~ -7 ~ -24(34)]
|   vref 17.2%:--------------------------*************************----------[8 ~ -7 ~ -23(31)]
|
```


- 上图为单个DQ眼图结果。
- 左侧vref为对应该参考电压下的眼宽情况，vref前的'*'表明该参考电压下的眼宽最宽，'+'为当前固件所使用的参考电压。Vref取值并非准确取值，实际内部取值会是最接近该值的可配置值，例如上图的30.4%实际内部可配置值是30.2%的话，则实际生效值是30.2%。
- 中间眼宽部分'-'为无效相位点。'*'为有效相位点。'+'为默认采样点。'|'范围为实际setup/hold time需要的最小margin范围。'+'到最左侧'*'为信号的setup time margin，'+'到最右侧'*'为信号hold time margin。当实际有效眼图压到'|'范围内的点则会用'o'替代，有出现'o'则表明实际眼图存在风险。如下图：

```
tx dq12, max_eye vref:15.6%[71]:
  vref 33.8%:--------------------------*********----------------------[3 ~ -6 ~ -15(18)]
  vref 32.4%:-----------------------************--------------------[7 ~ -6 ~ -19(26)]
  vref 31.0%:----------------------**************------------------[9 ~ -6 ~ -21(30)]
  vref 29.6%:--------------------****************------------------[12 ~ -5 ~ -23(35)]
  vref 28.2%:-------------------******************-----------------[14 ~ -5 ~ -25(39)]
  vref 26.8%:-----------------********************----------------[18 ~ -4 ~ -27(45)]
  vref 25.4%:-----------------**********************--------------[19 ~ -5 ~ -30(49)]
  vref 24.0%:---------------************************--------------[23 ~ -4 ~ -31(54)]
  vref 22.6%:---------------************************--------------[23 ~ -4 ~ -32(55)]
  vref 21.2%:-------------**************************------------[26 ~ -4 ~ -34(60)]
  vref 19.8%:-------------**************************************----------[29 ~ -3 ~ -36(65)]
 +vref 18.4%:-------------**|**********+**********|******----------[30 ~ -4 ~ -39(69)]
  vref 17.0%:-------------**|**********+**********|******----------[31 ~ -4 ~ -39(70)]
* vref 15.6%:----------************************************----------[34 ~ -1 ~ -37(71)]
  vref 14.2%:----------************************************----------[32 ~ -1 ~ -35(67)]
  vref 12.8%:-------------**************************************----------[29 ~ -1 ~ -32(61)]
  vref 11.4%:-------------**************************************----------[26 ~ -2 ~ -30(56)]
  vref 10.0%:---------------************************************----------[23 ~ -3 ~ -29(52)]

tx dq13, max_eye vref:15.6%[67]:
  vref 33.8%:-----------------------------****----------------------[-8 ~ -11 ~ -14(6)]
  vref 32.4%:--------------------------*********--------------------[0 ~ -8 ~ -17(17)]
  vref 31.0%:-------------------------***********------------------[3 ~ -7 ~ -18(21)]
  vref 29.6%:-----------------------*************------------------[7 ~ -7 ~ -21(28)]
  vref 28.2%:----------------------***************-----------------[8 ~ -8 ~ -24(32)]
  vref 26.8%:--------------------*****************----------------[11 ~ -8 ~ -27(38)]
  vref 25.4%:-------------------*******************---------------[14 ~ -8 ~ -30(44)]
  vref 24.0%:-------------------*******************---------------[15 ~ -7 ~ -30(45)]
  vref 22.6%:-----------------*********************---------------[17 ~ -7 ~ -31(48)]
  vref 21.2%:-----------------***********************-------------[20 ~ -7 ~ -35(55)]
  vref 19.8%:---------------*************************-------------[22 ~ -7 ~ -36(58)]
 +vref 18.4%:-------------o-oo**********+**********|******----------[23 ~ -8 ~ -39(62)]
  vref 17.0%:-------------o-o**********+**********|*******---------[24 ~ -8 ~ -41(65)]
* vref 15.6%:-------------*************************************---------[27 ~ -6 ~ -40(67)]
  vref 14.2%:-------------*************************************---------[29 ~ -4 ~ -38(67)]
  vref 12.8%:-------------*************************************---------[29 ~ -3 ~ -36(65)]
  vref 11.4%:-------------**************************************---------[26 ~ -3 ~ -33(59)]
  vref 10.0%:---------------************************************---------[22 ~ -4 ~ -31(53)]
```

- 右侧'[]'中的数据从左到右分别为左侧眼宽边界（setup time余量)，中间值，右侧眼宽边界（hold time余量），总眼宽宽度。

```
rx all dq, max_eye vref:30.1%[76]:
  vref 42.7%:------------------------**********----------[8 ~ -8 ~ -24(32)]
  vref 41.3%:----------------------**********----------[13 ~ -6 ~ -25(38)]
  vref 39.9%:--------------------**********----------[20 ~ -4 ~ -28(48)]
  vref 38.5%:----------------**********----------[27 ~ -2 ~ -31(58)]
  vref 37.1%:--------------**********----------[30 ~ -1 ~ -32(62)]
  vref 35.7%:------------**********----------[32 ~ -1 ~ -35(67)]
  vref 34.3%:----------**********----------[35 ~ -1 ~ -37(72)]
  vref 32.9%:----------**********---------[37 ~ 0 ~ -38(75)]
  vref 31.5%:--------**********----------[40 ~ 2 ~ -35(75)]
*+vref 30.1%:-------**********|**********+***|**********----------[43 ~ 5 ~ -33(76)]
  vref 28.7%:-------**********|**********+***|**********----------[43 ~ 5 ~ -32(75)]
  vref 27.3%:--------**********----------[40 ~ 4 ~ -31(71)]
  vref 25.9%:---------**********----------[39 ~ 5 ~ -29(68)]
  vref 24.5%:----------**********----------[36 ~ 3 ~ -29(65)]
  vref 23.1%:------------**********----------[33 ~ 3 ~ -26(59)]
  vref 21.7%:-------------**********----------[31 ~ 2 ~ -26(57)]
  vref 20.3%:--------------**********----------[28 ~ 2 ~ -24(52)]
  vref 18.9%:----------------**********----------[24 ~ 1 ~ -22(46)]
  vref 17.5%:-----------------**********----------[22 ~ 1 ~ -19(41)]
```

- 上图为所有DQ累加的眼图结果。

```
cs0 RD:
max eye:
left : -42  -42  -44  -46  -39  -44  -44  -39 ,  -45  -44  -47  -44  -46  -47  -44  -49

midd : 0  0   -2   -4    1   -2   -3  0 ,    3    3  0    3    3    1    2   -1

right:  41   43   39   38   42   40   38   40 ,   51   51   48   51   52   50   49   47

range:  83   85   83   84   81   84   82   79 ,   96   95   95   95   98   97   93   96

current eye:
left : -39  -44  -44  -43  -39  -44  -41  -39 ,  -44  -43  -45  -41  -45  -46  -40  -47

midd : 0   -1   -2   -1    1   -2  0  0 ,    3    4    1    5    3    2    6  0

right:  38   41   39   40   41   40   40   40 ,   51   52   48   52   52   51   52   48

range:  77   85   83   83   80   84   81   79 ,   95   95   93   93   97   97   92   95
```

- 为了方便查看所有单个DQ眼图结果，上图为所有单个DQ眼图结果的总结。
- 其中max eye结果列出所有DQ最大眼宽也就是'+' 所指向vref行最右侧'[]'中的左侧眼宽边界，中间值，右侧眼宽边界，总眼宽宽度信息总结。从左到右分别为DQ0，DQ1，至最大DQ。
- 其中current eye结果列出所有DQ当前所使用的vref处的眼宽结果，也就是'*'所指向vref行的结果总结。打印格式类似max eye的打印。
- 最终结果：在眼图log最后会打印“all result: pass”则表明所有的dq余量满足要求，否则会打印“all result: err”。

4.2 一维眼图的获取

能用二维眼图的平台尽量用二维眼图，因为多了vref的维度。除非有特殊需求，才用一维眼图。

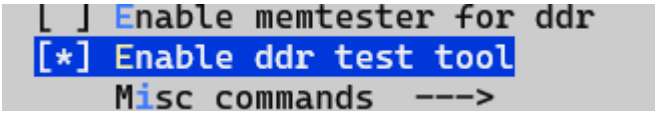
4.2.1 支持的平台

RV1109、RV1126、RK3566、RK3568、RK3528、RK3562

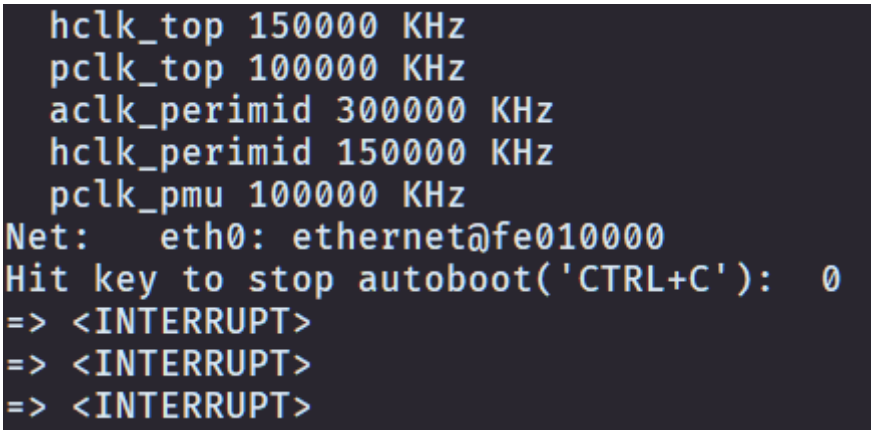
4.2.2 使用方法

4.2.2.1 前期准备

1. 确认 DDR bin 版本（RV1126 平台需要 V1.09 或以上，RK3568/RK3566 平台需要 V1.07 或以上），如果版本太旧请更新 rkbin 工程
2. 编译 U-Boot 工程前，在工程根目录下打开 menuconfig，进入 Command line interface，配置 Enable ddr test tool 并保存编译配置（Rockchip DDR DQ 眼图工具集成在 DDR Test Tool 中）。



3. 编译 U-Boot 工程并烧写编好的 Loader 和 uboot（具体请参考 UBOOT 文档中“编译烧写”相关章节）。
4. 将待测单板的串口连接至上位机，确保单板与上位机可以通过串口正常通信。单板开机时，上位机长按 Ctrl + C 让单板停留在 U-Boot（出现“<INTERRUPT>”说明单板已停留在 U-Boot）。



4.2.2.2 U-Boot 下查看 DDR DQ 读写眼图

U-Boot 下输入命令

```
ddr_dq_eye <DDR frequency in MHz>
```

参数 <DDR frequency in MHz> 指定需要查看 DQ 眼图的 DDR 时钟频率，单位为 MHz，留空时默认为最高频率。

- 例：查看 DDR 时钟频率为 1056MHz 时的 DQ 眼图，U-Boot 下输入命令

```
ddr_dq_eye 1056
```

- 例：查看 DDR 最高时钟频率时的 DQ 眼图，U-Boot 下输入命令

```
ddr_dq_eye
```

4.2.2.3 输出结果分析

```
=> <INTERRUPT>
=> ddr_dq_eye
Rockchip DDR DQ Eye Tool v0.0.4
DDR type: DDR3
CS0 1056MHz read DQ eye:
     0   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60     Margin_L Sample Margin_R Width    DQS
DQ0  -------******|******----------------------------------------------      6      13       6    [ 13]     17
DQ1  --**********|**********----------------------------------------------     10      12      11       22     17
DQ2  --**********|**********----------------------------------------------     10      12      11       22     17
DQ3  --**********|**********----------------------------------------------     10      12      11       22     17
DQ4  -**********|**********-----------------------------------------------     10      11      11       22     17
DQ5  -**********|**********----------------------------------------------     11      12      11       23     17
DQ6  ---*********|**********----------------------------------------------     10      13      10       21     17
DQ7  -**********|**********----------------------------------------------     11      12      11       23     17
DQ8  ---*********|**********----------------------------------------------     11      13      11       23     17
DQ9  --**********|**********----------------------------------------------     10      12      11       22     17
DQ10 ---*********|**********----------------------------------------------     11      13      11       23     17
DQ11 --**********|**********----------------------------------------------     11      13      11       23     17
DQ12 ---*********|**********----------------------------------------------     11      13      11       23     17
DQ13 --**********|**********----------------------------------------------     11      13      11       23     17
DQ14 ---*********|**********----------------------------------------------     11      13      11       23     17
DQ15 --**********|**********----------------------------------------------     11      13      11       23     17
DQ16 ---*********|**********----------------------------------------------     11      14      11       23     18
DQ17 ---*********|**********----------------------------------------------     11      14      11       23     18
DQ18 ---*********|**********----------------------------------------------     10      13      11       22     18
DQ19 ---*********|**********----------------------------------------------     11      13      11       23     18
DQ20 ---*********|**********----------------------------------------------     11      14      11       23     18
DQ21 --**********|**********----------------------------------------------     10      12      11       22     18
DQ22 ---*********|**********----------------------------------------------     11      14      11       23     18
DQ23 ----********|**********----------------------------------------------     10      14      11       22     18
DQ24 -**********|**********-----------------------------------------------     11      12      11       23     17
DQ25 ---*********|**********----------------------------------------------     11      13      11       23     17
DQ26 --**********|**********----------------------------------------------     10      12      11       22     17
DQ27 --**********|**********----------------------------------------------     10      12      11       22     17
DQ28 --**********|**********----------------------------------------------     10      12      11       22     17
DQ29 -**********|**********-----------------------------------------------     11      12      11       23     17
DQ30 --**********|**********----------------------------------------------     10      12      11       22     17
DQ31 ---*********|**********----------------------------------------------     11      13      11       23     17

CS0 1056MHz write DQ eye:
     0   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60     Margin_L Sample Margin_R Width    DQS
DQ0  ----------------------**********|**********----------------------     11      31      12       24     36
DQ1  ----------------------**********|**********----------------------     11      33      12       24     36
DQ2  ----------------------**********|**********----------------------     11      31      12       24     36
DQ3  ----------------------**********|**********----------------------     11      33      11       23     36
DQ4  ----------------------**********|**********----------------------     11      32      12       24     36
DQ5  ----------------------**********|**********----------------------     11      32      12       24     36
DQ6  ----------------------**********|**********----------------------     11      32      12       24     36
DQ7  ----------------------**********|**********----------------------     11      31      12       24     36
DQ8  --------------**********|**********----------------------------     11      26      12       24     29
DQ9  --------------**********|**********----------------------------     11      25      12       24     29
DQ10 --------------**********|**********----------------------------     11      25      12       24     29
DQ11 --------------**********|**********----------------------------     11      25      12       24     29
DQ12 --------------**********|**********----------------------------     11      25      12       24     29
DQ13 --------------**********|**********----------------------------     11      25      11       23     29
DQ14 --------------**********|**********----------------------------     11      26      11       23     29
DQ15 --------------**********|**********----------------------------     11      25      12       24     29
DQ16 ----------------------**********|**********----------------------     11      35      11       23     38
DQ17 ----------------------**********|**********----------------------     11      34      12       24     38
DQ18 ----------------------**********|**********----------------------     11      33      12       24     38
DQ19 ----------------------**********|**********----------------------     11      33      12       24     38
DQ20 ----------------------**********|**********----------------------     11      34      12       24     38
DQ21 ----------------------**********|**********----------------------     11      34      11       23     38
DQ22 ----------------------**********|**********----------------------     11      35      11       23     38
DQ23 ----------------------**********|**********----------------------     11      34      12       24     38
DQ24 --------------**********|**********----------------------------     11      25      12       24     28
DQ25 --------------**********|**********----------------------------     11      25      12       24     28
DQ26 --------------**********|**********----------------------------     11      25      11       23     28
DQ27 -------------**********|**********-----------------------------     11      24      11       23     28
DQ28 -------------**********|**********-----------------------------     11      24      11       23     28
DQ29 --------------**********|**********----------------------------     11      25      11       23     28
DQ30 --------------**********|**********----------------------------     11      25      12       24     28
DQ31 --------------**********|**********----------------------------     11      25      11       23     28

DQ eye width min: 13(read), 23(write)
DQ eye width limit: 14(read), 14(write) in 1056MHz
DQ eye width may be unreliable, please check!
```

- 工具首先输出工具版本、DDR 类型、频率等信息
- 工具分别输出各 CS 的读眼图与写眼图
- 输出眼图图形中，“-”标记的位置位于眼图外，“*”标记的位置位于眼图内，“|”标记的位置为采样点
- 眼图图形右边显示了采样点距离眼图左右边界的裕量（Margin_L、Margin_R）、采样点位置（Sample）、眼宽（Width）等信息，方括号标记的为不满足最小眼宽限制的眼宽（如图中读眼图 DQ0）
- 工具最后输出了读眼图和写眼图的最小眼宽，以及最小眼宽限制值（选取相近的频率）

4.2.3 DDR DQ 最小眼宽限制

根据 DEMO 测试和相关项目经验，本文档对 DDR DQ 的最小读写眼宽做出了相应的限制。如果最小读写眼宽不满足此限制值，DDR 的运行可能不稳定。

> 满足 DDR DQ 最小眼宽限制只能说明当前设计下 DDR DQ 眼宽大小较为可靠，不代表 DDR 的相关设计一定不存在其它问题，请根据实际使用需求做进一步的可靠性测试。

4.2.3.1 RV1126 DDR DQ 最小眼宽限制值

DDR 类型	DDR 时钟频率	最小读眼宽限制值	最小写眼宽限制值
LPDDR4	1056MHz	12	13
LPDDR4	924MHz	15	15
DDR4	1056MHz	13	9
DDR4	924MHz	15	11
LPDDR3	1056MHz	15	13
LPDDR3	924MHz	16	15
DDR3	1056MHz	14	14
DDR3	924MHz	17	17

4.2.3.2 RK3568/RK3566 DDR DQ 最小眼宽限制值

DDR 类型	DDR 时钟频率	最小读眼宽限制值	最小写眼宽限制值
LPDDR4	1560MHz	25	24
LPDDR4	1184MHz	30	29
DDR4	1560MHz	30	22
DDR4	1184MHz	32	26
LPDDR3	1184MHz	34	25
LPDDR3	1056MHz	39	28
DDR3	1184MHz	32	31
DDR3	1056MHz	39	34

5. Chapter-5 Rockchip DDR 带宽工具使用说明

5.1 名词解释

本文档涉及的主要名词解释：

- monitor： 是监视和统计各DDR命令的功能。

5.2 工具获取

https://redmine.rock-chips.com/documents/49 -> DDR相关资料_VerX.XX.7z -> 工具 -> DDR带宽工具

5.3 平台支持

通过如下命令获取支持的平台。

```
# rk-msch-probe_vx.xx -h
```

将打印如下信息（具体支持平台情况查看工具版本的打印）：

```
Usage: [-c chip_name -d msecs -f freq -h help]
-c chip_name:
'rk312x' include rk3126,rk3128 and px3se
'rk322x' include rk3128h,rk3228a,rk3228b and rk3229
...
[option] -d msecs for sampling interval
[option] -f freq mean current ddr frequency unit MHz
-h for help
```

5.4 参数说明

DDR带宽测试工具（rk_msch_probe_vx.xx）支持如下参数传入。

- -c：芯片名称，确认当前平台支持后，按照此平台名称输入即可，如 rk3326。
- -d：监视间隔时间，单位ms，可选。若不传参，则默认1000ms。
- -f： 当前 DDR 的频率，单位MHz，可选。每个监视时间段，工具都会尝试获取DDR频率。若获取失败，则需要传入当前DDR的频率。

  注意：需要先把DDR定频，否则DDR频率可能发生变化，测出结果不准确。
- -t test_loop：指定测试轮次，到达后退出。默认无限次数。
- -h： 帮助。

5.5 使用条件

DDR devfreq的策略不能是'dmc_ondemand'，建议切换到'userspace'，命令:

```
# echo userspace > /sys/class/devfreq/dmc/governor
```

然后将DDR定频，如DDR定频到780MHz：

```
# echo 780000000 > /sys/class/devfreq/dmc/userspace/set_freq
```

工具测试期间，若出现如下打印，则说明上述使用条件不满足：

```
Error: The DDR monitor time gets error!!!
Please check the devfreq governor is not 'dmc_ondemand'!!!
For example:
# cat sys/class/devfreq/dmc/governor
# dmc_ondemand
Please switch the devfreq governor to others, such as 'userspace'.
For example:
# echo userspace > /sys/class/devfreq/dmc/governor
Delete the result of this time!
```

5.6 打印说明

执行`rk_msch_probe_vx.xx`，将监视并打印DDR带宽及利用率。

一般平台打印如下：

```
V1.06_20200629
ddr freq: 928Mhz
CH0:
ddr monitor statistics:
ddr load = 3251.23MB/s(43.76%) [RD:1859.93MB/s(25.03%), WR:1391.30MB/s(18.72%),
ACT(access : active): 3.34, srex:0.54%, pdex:1.27%, clkstp:0.00%, lp:1.81%]
```

RK3588打印改动较大，如下：

```
ddr freq: 2133Mhz
==========ALL=============CH0===============CH1==============CH2===============
CH3========
LOAD:  4.51MB/s(0.03%),  1.23MB/s(0.04%),  1.13MB/s(0.03%),  1.16MB/s(0.04%),
 1.00MB/s(0.03%)
RD:    2.57MB/s(0.02%),  0.74MB/s(0.02%),  0.65MB/s(0.02%),  0.67MB/s(0.02%),
 0.51MB/s(0.02%)
WR:    1.94MB/s(0.01%),  0.49MB/s(0.02%),  0.48MB/s(0.01%),  0.49MB/s(0.02%),
 0.48MB/s(0.01%)
================================================================================
===========
```

统计的结果说明：

打印	解释
ALL	所有channel总的带宽统计信息
CHx	DDR channel x的带宽统计信息
recorded LOAD（max，min，avg）	对于所有DDR bank，运行此工具已记录到的最大、最小、平均带宽及负载
load/LOAD	对于所有DDR bank，此channel的带宽及负载
RD	对于所有DDR bank，DDR read 数据的带宽及占比
WR	对于所有DDR bank，DDR write 数据的带宽及占比
PRE	对于所有DDR bank，命令precharge占总命令数量的百分比。precharge与active成对出现，实际没有硬件统计，而是直接取用active命令统计结果
ACT(access : active)	对于所有DDR bank，平均每个DDR active命令后有几个read/write，值越大说明访问DDR地址越连续，越大越好
srex	DDR 处于 self-refresh 状态的时间占比
pdex	DDR 处于 power down 状态的时间占比
clkstp	DDR 处于 clock stop 状态的时间占比
srpdex	DDR 处于 self-refresh power down 状态的时间占比
dsmex	DDR 处于 Deep Sleep Mode 状态的时间占比
lp/ LOW POWER	DDR 在 low power 状态的时间占比（low power状态包括self-refresh，power down 和 clock stop等）

5.7 FAQ

1）使用时异常打印 /dev/mem 不存在的情况。打印如下

```
rk3568:/ # rk-msch-probe_vx.xx -c rk356x
open /dev/mem error: No such file or directory
```

是因为kernel 没有打开 CONFIG_DEVMEM 宏，工具无法访问到寄存器。

方式1：kernel里对应config，如android-11.config，增加 CONFIG_DEVMEM=y，再次编译固件；

方式2：kernel 里敲make menuconfig，搜索DEVMEM，选择打开，重新编译kernel即可（注意不要覆盖config）。

```
Symbol: DEVMEM [=y]
Type  : bool
Prompt: /dev/mem virtual device support
Location:
    -> Device Drivers
(1)   -> Character devices
  Defined at drivers/char/Kconfig:10

...
[*] /dev/mem virtual device support
...
```

6. Chapter-6 Rockchip DDR_UserTool（DDR 焊接检测工具）指南

6.1 概述

DDR_UserTool 是用于 **DDR 焊接检测**的工具。

本工具发布于 https://redmine.rock-chips.com/documents/49，DDR相关资料_Verx.xx.7z->工具->DDR_UserTool_vx.xx。

DDR_UserTool 的版本号可能长期不变，但是内部配置文件会因为支持新平台等原因进行更新，所以每个新项目都需要上 Redmine 获取 DDR_UserTool 的最新版。

6.2 测试步骤

6.2.1 打开 DDR_UserTool.exe

> Note：Windows 7 系统需要右键“以管理员身份运行”。

6.2.2 选择测试文件

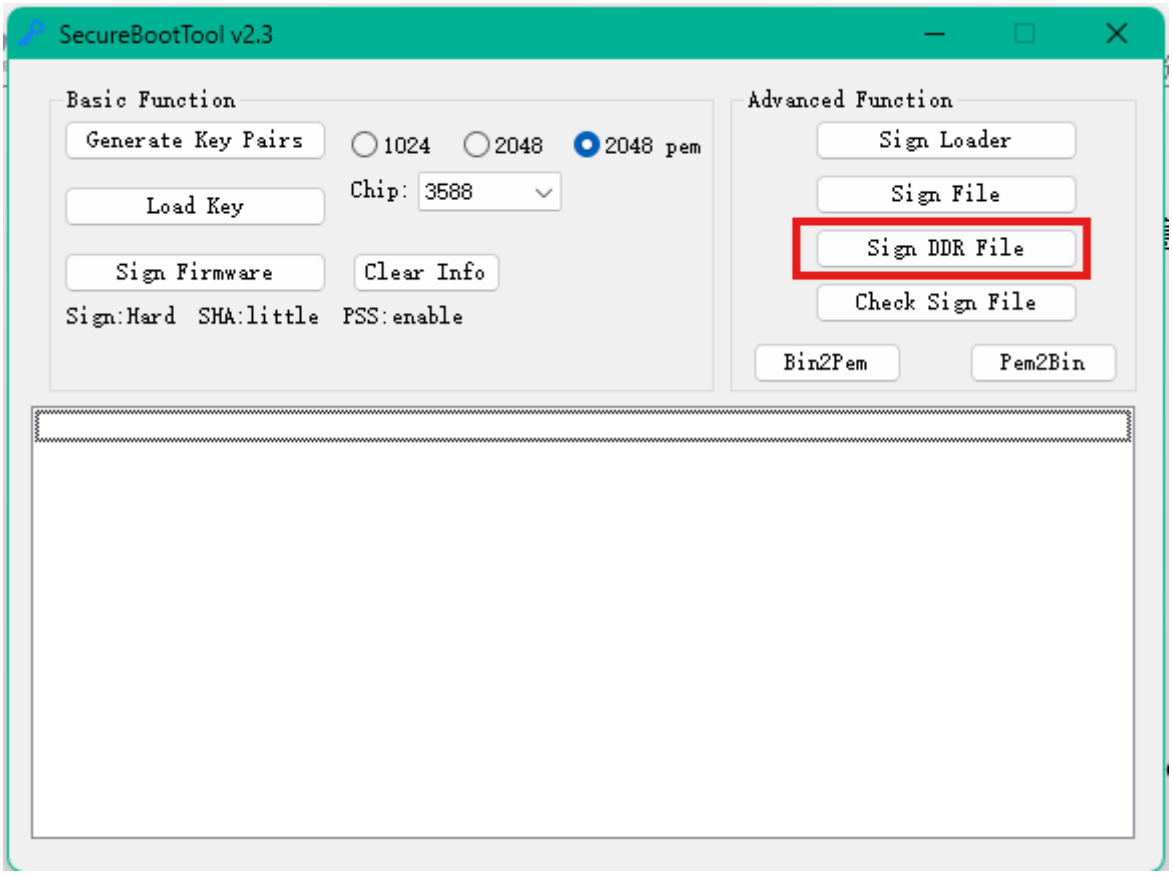
测试文件位于工具目录下的“测试文件”文件夹中，后缀名为 .cfg，请根据待测平台和所用的 DDR 容量信息选择对应的测试文件。

请注意：

- DDR4 x16 DDP (Dual Die Package) 的颗粒需要看作是两片 x8 的颗粒。例如：机器本身是“4GB DDR4(用1个CS且由2片1Gb×16bit组成)”，如果所用的 DDR4 颗粒是 x16 DDP 的，就要选择“4GB DDR4(用1个CS且由4片1Gb×8bit组成)”对应的测试文件。判断 DDR4 x16 DDP 的方法请参考“如何判断 DDR4 x16 DDP 颗粒”章节
- LPDDR4 和 LPDDR4X 的测试文件不通用，请注意区分。
- DDR3L 用 DDR3 对应的测试文件。

6.2.3 签名测试文件

该步骤仅适用于已刷 OTP 的机器。未刷 OTP 的机器请跳过该步骤。



测试已刷 OTP 的机器需要先用 SecureBootTool 签名测试文件，SecureBootTool 也放在 https://redmine.rock-chips.com/documents/49。配置好 Chip、Key 等选项后，点击“Sign DDR File”签名所需的测试文件，SecureBootTool 的具体使用方法请参考 SecureBootTool_UserManual（SecureBoot签名工具用户手册）。

6.2.4 待测机器进入测试状态

连接待测机器的 OTG，若有条件建议也连接串口。

待测机器进入测试状态有如下几种方法：

- 机器未烧写过任何固件

  这种机器本身就处于 Maskrom 状态，可以直接测试。

- 让机器进入 Maskrom 状态

机器上电时短路存储设备的 CLK 和 GND，让机器进入 Maskrom 状态。具体短路哪些 pin 可以参考 RK 发布的原理图，一般都有预留进入 Maskrom 模式的按键或 TP。

- 让机器进入 Loader 模式

待测机器正常进入测试状态后，工具将显示“发现一台设备”。

6.2.5 开始测试

点击“开始测试”。进度条开始滚动，工具界面也会同步显示测试 log。测试一般在 2 分钟内完成。

如果勾选了“自动测试”，那么当有设备插入时，会自动进行测试。

6.2.6 获取测试结果

6.2.6.1 测试 PASS

6.2.6.2 测试 FAIL

请根据工具的报错打印信息，补焊或重焊出错的 DDR 颗粒和 **SoC**。可参考“报错打印信息与实际硬件信号线的对应关系”章节找到具体报错的信号线。

对于测试失败的机器，维修后，还必须再进行测试，直到测试成功为止。

6.2.6.3 测试中断

请根据"测试中断的原因"章节进行排查。

6.3 FAQ

6.3.1 DDR_UserTool 能否测试 DDR 稳定性？

不能！DDR_UserTool 现在只提供 DDR 焊接检测功能。DDR 稳定性的相关测试请进入系统进行，请参考本文档《DDR 颗粒验证流程说明》章节。

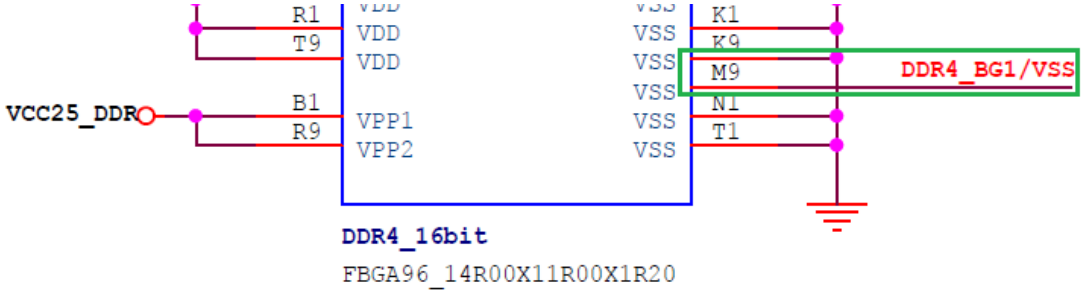
6.3.2 DDR_UserTool 能检测到所有 pin 的焊接问题吗？

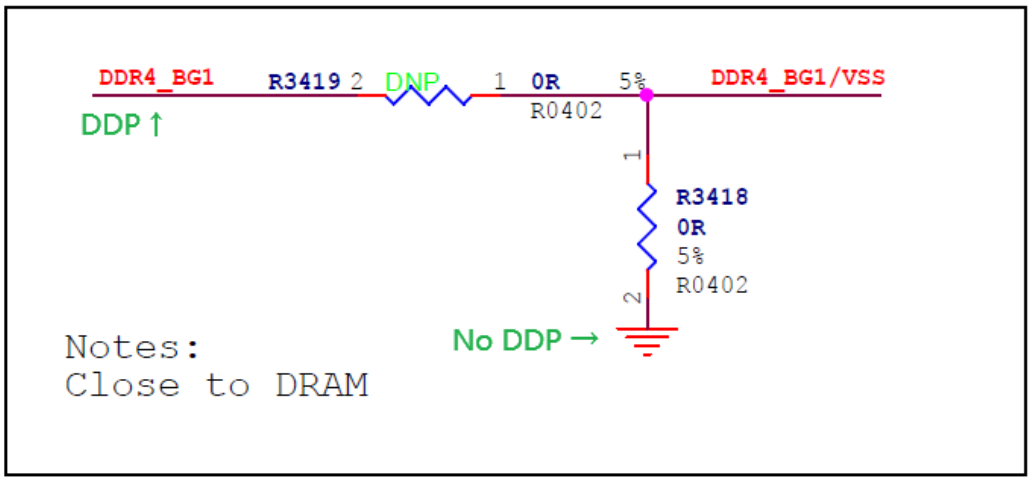
不能！DDR_UserTool 是通过软件方式提供一个快速、便捷的 DDR 焊接检测方案，但是软件方式是有局限性的，没法检测所有 DDR 相关的 pin，更没法检测其它模块的 pin。

一般来说，DDR 的 DQ 和大部分 CA 的焊接问题是可以检测出来的。电源和 CKE、ODT 的焊接问题无法检测。

6.3.3 如何判断 DDR4 x16 DDP 颗粒

1. 看原理图。DDR4 颗粒的 M9 pin 如果连到了 SoC 端 BG1 对应的 pin 就是 x16 DDP 颗粒，如果连到了 VSS 就不是。





2. 看正常机器开机的 Loader 打印。如果 DDR 容量信息打印中 Die BW=8 就是 x16 DDP 颗粒。
3. 看颗粒的 datasheet。是否有说明是 DDP 颗粒；或者如果 M9 pin 是 BG1 就是 x16 DDP 颗粒，如果是 VSS 就不是。

6.3.4 找不到机器对应的测试文件

检查是否有更新。

如果较新的平台暂时没有对应的测试文件，请等待工具更新。

对于已经发布测试文件的平台，常见的 DDR 容量信息一般都有对应的测试文件，测试文件数量较多，请再确认下。如果所用 DDR 容量组合比较特殊，确实找不到对应的测试文件，请上 Redmine 提问。

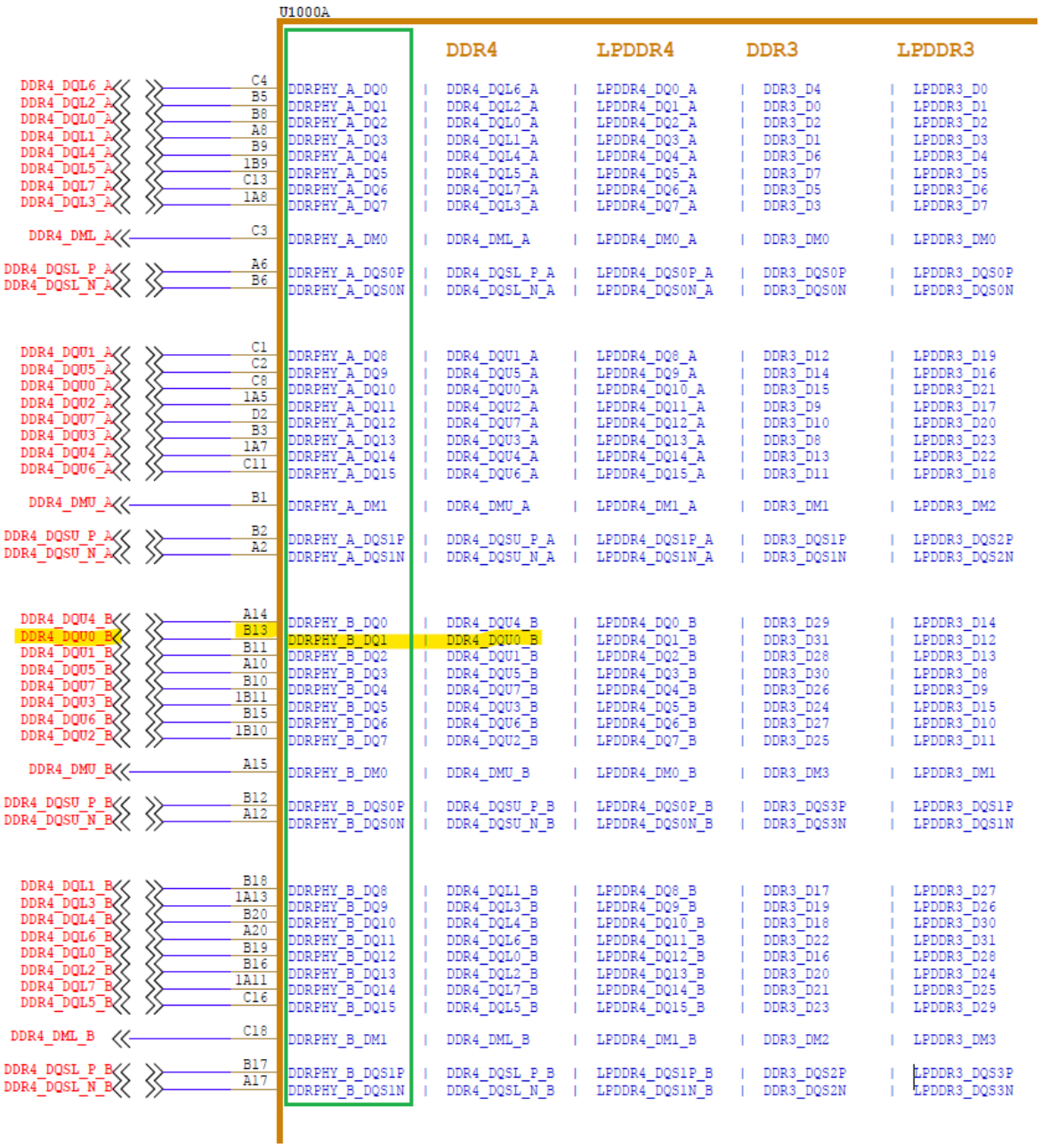
6.3.5 测试中断的原因

1. Windows 7 系统没有“以管理员身份运行”
2. 测试文件选择错误（如平台不匹配、DDR 类型不匹配、DDR 容量信息不匹配）
3. 机器没有重新进入测试状态，被重复测试
4. 机器已经刷了 OTP，却没有签名测试文件
5. 测试工具、测试文件未更新
6. 机器电源问题（例如 USB 供电不足）
7. USB 连接问题，请尝试更换线材或更换电脑测试
8. 如果机器是 Loader 模式，尝试换成 Maskrom 模式进行测试

6.3.6 报错打印信息与实际硬件信号线的对应关系

CA 总线不允许调换，对应关系一般没有歧义。

报错打印信息中如果有出错的 DQ，都是对应 SoC 端 pin 的命名，且需要看 pin 名字中通用的部分，不能看 DDR 类型对应的部分。例如，下图是 RK3528 DDR4 原理图的一部分，SoC pin 名字中通用的部分是绿色框起来的一列；若工具报 DQ1_B 错误，对应到原理图是 SoC 的 DDRPHY_B_DQ1 这个 pin 对应的信号线有问题，即 DDR4_DQU0_B 这条信号线。



6.3.7 DDR_UserTool 测试 FAIL，但是 X-ray 没看到有焊接问题

X-ray 不一定能看出所有的焊接问题，还是请尝试补焊或重焊 DDR 颗粒和 **SoC**。

6.3.8 已经补焊或重焊了，还是测试 FAIL

1. 测试文件选择错误或测试工具、测试文件未更新
2. DDR 颗粒端和 SoC 端都可能虚焊，两端都需要补焊或重焊
3. PCB 工艺有问题，例如走线没导通
4. 其它非 DDR 焊接问题（DDR 颗粒问题、硬件设计问题、电源问题等）

6.3.9 大量机器测试 FAIL

1. 测试文件选择错误或测试工具、测试文件未更新
2. 贴片厂、PCB 工艺问题
3. 其它非 DDR 焊接问题（DDR 颗粒问题、硬件设计问题、电源问题等）

6.3.10 还有疑问需要上 Redmine 提问

为了提高沟通效率，在清晰地描述问题的同时，请提供以下信息：

1. 实际使用的 DDR 颗粒型号、datasheet
2. 板子上贴了几片颗粒，最好能提供原理图
3. 所用的测试文件
4. 异常机器的比例
5. 工具目录下的 DDR_Test_Log.txt、resource\Log 中对应的 Log 文件、测试时的串口 Log（如有正常机器，也请提供正常机器的这些 log 供分析）
6. 开机上电的串口 log（如有正常机器，也请提供正常机器上电的 log 供分析）

7. Chapter-7 Rockchip Developer Guide HAL DDR ECC

7.1 适用性

部分文档，仅适用于带有HAL的系统，不适用Linux系统。

7.2 名词解释

本文档的简写	本文档内释义
ECC	Error Correcting Code
SEC ECC	Single Bit Single Error Correction Code
DED ECC	Double Error Detection Error Correction Code
DDR	Double Data Rate SDRAM
CE	Correctable Error，指单bit可检测可纠正性错误
UE	Uncorrectable Error，指双bit可检测不可纠正性错误
cs	chip select
Row	特指 DDR row 地址
Chip ID	特指 DDR chip id，未激活功能，请忽略
BankGroup	特指 DDR4 Bank Group 地址，其他DDR类型请忽略
Bank	特指 DDR bank 地址
Col	特指 DDR column 地址
Bit position	特指 CE 纠正的bit位

7.3 简介

ECC 指 Error Correcting Code ，而DDR ECC 是对DDR数据进行错误检查和纠正的。

RK3568支持的是SideBand ECC，即在DDR数据通道旁增加一个专门存放ECC数据的DDR通道。

其ECC具有纠错1bit，发现2bit错误的能力，即SEC/DED ECC（Single Error Correction/Double Error Detection）。

7.4 开启DDR ECC

1. 使能ECC：只要DDR_ECC_DQ0-7有贴颗粒，ECC会自动enable。

2. DDR_ECC_DQ0-7的作用：ECC是32bit的DQ数据+7bit的ECC数据。DDR_ECC_DQ0-7用于存储DQ0-DQ31计算得到的ECC数据。所以需要多贴一颗用于存放ECC数据的颗粒。
3. 多贴的一颗ECC颗粒的要求：颗粒类型，Row/Col/Bank需要与DQ0-31上所贴的颗粒一致。
4. 支持的颗粒类型：所有颗粒类型均支持ECC。

7.5 HAL里获取DDR ECC信息

DDR ECC 具体的错误检查与纠正行为是硬件算法完成，软件可以获取相关信息。

7.5.1 配置

在相应工程的 hal_conf.h 下使能DDR ECC模块。如rk3568，在project/rk3568/src/hal_conf.h添加如下代码：

```
#define HAL_DDR_ECC_MODULE_ENABLED
```

7.5.2 代码和API

- lib/hal/src/hal_ddr_ecc.c
- lib/hal/inc/hal_ddr_ecc.h

```
/* 初始化DDR ECC相关信息 */
HAL_Status HAL_DDR_ECC_Init(struct DDR_ECC_CNT *p);

/* 获取DDR ECC累计的统计信息，包括单bit可纠正错误的数量和双bit可检测不可纠正错误的数量 */
HAL_Status HAL_DDR_ECC_GetInfo(struct DDR_ECC_CNT *p);
```

7.5.3 使用范例

上层软件可以用两种方式获取DDR ECC信息：软件轮询和硬件中断。

- 软件轮询方式

  示例如下：

```
struct DDR_ECC_CNT eccInfo;

void HAL_DDR_ECC_TEST_POLL(void)
{
    uint32_t cpuID;

    cpuID = HAL_CPU_TOPOLOGY_GetCurrentCpuId();
    if (cpuID == 0) {                       /* 使用一个cpu、线程或其他方式，初始化并轮询DDR ECC状态 */
        HAL_DDR_ECC_Init(&eccInfo);
        while (1) {                         /* 初始化DDR ECC信息之后，轮询获取DDR ECC信息 */
```

```
            HAL_DDR_ECC_GetInfo(&eccInfo); /* 累计的CE和UE数量存放在结构体
eccInfo中 */
            HAL_DelayMs(50);               /* 轮询间隔时间，可用其他让cpu空闲的
API */
        }
    }
}
```

- 硬件中断方式

  示例如下：

```
struct DDR_ECC_CNT eccInfo;

void HAL_DDR_ECC_IRQHandler(uint32_t irq)
{
    HAL_DDR_ECC_GetInfo(&eccInfo);
}

void HAL_DDR_ECC_TEST_INT(void)
{
    uint32_t cpuID;

    cpuID = HAL_CPU_TOPOLOGY_GetCurrentCpuId();
    if (cpuID == 0) {                                                    /* 使用
一个cpu、线程或其他方式，初始化DDR ECC相关 */
        HAL_DDR_ECC_Init(&eccInfo);
        HAL_GIC_SetHandler(DDR_ECC_CE_IRQn, HAL_DDR_ECC_IRQHandler); /* 挂载
CE中断服务子程序 */
        HAL_GIC_SetHandler(DDR_ECC_UE_IRQn, HAL_DDR_ECC_IRQHandler); /* 挂载
UE中断服务子程序 */
        HAL_GIC_Enable(DDR_ECC_CE_IRQn);                                 /* 使能
CE中断服务 */
        HAL_GIC_Enable(DDR_ECC_UE_IRQn);                                 /* 使能
UE中断服务 */
    }
}
```

- 若检测到ECC出错，则打印获取的ECC信息。

```
# 检测到CE（可纠正错误）2个
[HAL WARNING] DDR ECC error: CE, 2 errors, the last is in DDR cs 0, Row
0xa0, ChipID 0x0, BankGroup 0x0, Bank 0x5, Col 0x318, Bit position
0x10000000

# 检测到UE（不可纠正错误）1个
[HAL ERROR] DDR ECC error: UE, 1 errors, the last is in DDR cs 0, Row 0xa0,
ChipID 0x0, bankGroup 0x0, Bank 0x5, Col 0x354
```

7.6 DDR ECC错误注入

提供一种DDR ECC错误注入的方式，用于验证DDR ECC的功能。开启DDR ECC错误注入功能后，对特定物理地址的写操作将会触发DDR ECC CE/UE。

7.6.1 代码和API

- lib/hal/src/hal_ddr_ecc.c
- lib/hal/inc/hal_ddr_ecc.h

```
/* 开启DDR ECC错误注入功能 */
HAL_Status HAL_DDR_ECC_PoisonEnable(struct DDR_ECC_CNT *p);
/* 关闭DDR ECC错误注入 */
HAL_Status HAL_DDR_ECC_PoisonDisable(struct DDR_ECC_CNT *p);
```

DDR ECC 默认注入的错误类型是 单bit可检测可纠正性错误（CE），若想修改为 双bit可检测不可纠正性错误（UE），修改如下：

```
static HAL_Status DDR_ECC_SMCPoison(uint32_t eccPoisonEn,
                                    struct DDR_ECC_CNT *priv)
{
    ...
    pShareMemCfg->eccPoisonMode = DDR_ECC_UE_DATA_POISON;
    ...
}
```

7.6.2 使用范例

```
struct DDR_ECC_CNT eccInfo;

void DDR_ECC_PoisonTriger(struct DDR_ECC_CNT *priv)
{
    uint32_t volatile *p = NULL;

    printf("DDR_ECC debug: eccPoisonAddr = 0x%llx\n", priv->eccPoisonAddr);
    if (priv->eccPoisonAddr) {
        p = (uint32_t volatile *)priv->eccPoisonAddr;
        *p = 0x5aa5f00f; /* 对错误注入的物理地址进行写操作 */
        HAL_DCACHE_CleanInvalidate(); /* 若操作的地址属性是cache的，需flush Dcache确保写操作成功 */
        printf("DDR_ECC debug: %p = 0x%x, reread = 0x%x\n", p, *p, *p);
    }

}

void DDR_ECC_GetInfo(void)
{
    HAL_DDR_ECC_GetInfo(&eccInfo);
}

void HAL_DDR_ECC_TEST_POLL(void)
{
    uint32_t cpuID;
    uint32_t i = 10;

    cpuID = HAL_CPU_TOPOLOGY_GetCurrentCpuId();
    if (cpuID == 0) {
        HAL_DDR_ECC_Init(&eccInfo);
```

```
        HAL_DDR_ECC_PoisonEnable(&eccInfo);/* 开启DDR ECC错误注入功能，错误注入的物理
地址将保存在eccInfo.eccPoisonAddr */
        while (i) {
            DDR_ECC_GetInfo();
            HAL_DelayMs(50);
            i--;
            if (i == 5) {
                DDR_ECC_PoisonTriger(&eccInfo); /* 触发DDR ECC错误注入 */
            }
        }
        HAL_DDR_ECC_PoisonDisable(&eccInfo);
    }
}
```

7.7 Note

1. DDR ECC 会使用 [0x100000, 0x1F000] 的DDR空间，若此空间未映射MMU，则需要添加映射。

```
--- a/lib/CMSIS/Device/RK3568/Source/Templates/mmu_rk3568.c
+++ b/lib/CMSIS/Device/RK3568/Source/Templates/mmu_rk3568.c
@@ -41,6 +41,8 @@ void MMU_CreateTranslationTable(void)
     // Define dram address space
     ...
+    MMU_TTSection(MMUTable, 0x100000, 0x100000 >> 20, Sect_Normal);
     ...
```

2. 错误注入的物理地址空间在HAL可能并没有映射，需增加映射。如地址是0x13576c20，则增加如下修改

```
--- a/lib/CMSIS/Device/RK3568/Source/Templates/mmu_rk3568.c
+++ b/lib/CMSIS/Device/RK3568/Source/Templates/mmu_rk3568.c
@@ -41,6 +41,8 @@ void MMU_CreateTranslationTable(void)
     // Define dram address space
     ...
+    MMU_TTSection(MMUTable, 0x13500000, 0x100000 >> 20, Sect_Normal);
     ...
```

3. DDR ECC 双bit可检测不可纠正性错误（UE）会触发CPU data abort异常。HAL默认无处理，若想看到DDR ECC UE的出错打印，可增加如下修改：

```
index 599e14a1..9da84468
--- a/lib/CMSIS/Device/RK3568/Source/Templates/GCC/startup_rk3568.c
+++ b/lib/CMSIS/Device/RK3568/Source/Templates/GCC/startup_rk3568.c
@@ -237,6 +237,7 @@ void Reset_Handler(void)
  *-----------------------------------------------------------------------
*/
 void Default_Handler(void)
 {
+    DDR_ECC_GetInfo();
     while(1);
 }
```