DDR 开发指南

文件标识：RK-KF-YF-39

发布版本：V1.3.0

日期：2021.1.21

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

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

日期	版本	作者	修改说明
2017.12.21	V1.0.0	何灿阳	
2018.3.30	V1.1.0	何灿阳	增加对 kernel 4.4 DDR 频率相关的描述
2019.1.29	V1.2.0	何智欢	增加调整 loader 的 de-skew 说明
2021.1.21	V1.3.0	汤云平	增加RK356x/RV1109/RV1126说明

目录

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
```

如何将我们给的 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
RK1808	rk\rkbin\bin\rk1x\rk1808_ddr_XXXMHz_vX.XX.bin	
RK3036	rk\rkbin\bin\rk30\rk3036_ddr3_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_300MHz_vX.XX.bin	
RK3308	rk\rkbin\bin\rk33\rk3308_ddr_XXXMHz_uartX_mX_vX.XX.bin	
PX30	rk\rkbin\bin\rk33\px30_ddr_333MHz_vX.XX.bin	
RK3326	rk\rkbin\bin\rk33\rk3326_ddr_333MHz_vX.XX.bin	
RK3368	rk\rkbin\bin\rk33\rk3368_ddr_600MHz_vX.XX.bin	
RK322xh	rk\rkbin\bin\rk33\rk322xh_ddr_333MHz_vX.XX.bin	
RK3328	rk\rkbin\bin\rk33\rk3328_ddr_333MHz_vX.XX.bin	
RK3399	rk\rkbin\bin\rk33\rk3399_ddr_XXXMHz_vX.XX.bin	2

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：不在这个表格中的芯片，就是不支持从 U-Boot 中生成 loader。

如何修改 U-Boot 中的 DDR 频率

目前这个功能只有 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。

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，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: 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 {
    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},
};
```

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

  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);
```

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 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
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
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
RK3126	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
RK3288	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
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

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/Total Capability。芯片有 2 个通道的，整机容量等于 2 个通道的 size/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
```

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，一种是变频。对于这 2 种变频策略，kernel 4.4 和 kernel 3.10 的实现有些不同。

kernel 4.4：

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，还是得按分支来分开处理

- 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==：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。

- 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

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：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： `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。

如何关闭 DDR 的负载变频功能，只留场景变频

- 对于 kernel 4.4，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

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：

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`

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，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

如何查看 DDR 带宽利用率

kernel 4.4 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:/sys/class/devfreq/dmc # cat load
11@396000000Hz
```

11 表示 DDR 的当前带宽利用率是 11%

如何测试 DDR 可靠性

请查看文档“DDR 颗粒验证流程说明”

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. 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“DDR 颗粒验证流程说明”包中可以找到，如何使用，也在里面，不在此文档范围。

3. 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“DDR 颗粒验证流程说明”包中可以找到，如何使用，也在里面，不在此文档范围。

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 信号是否在最高频能否达标，就不一定了，需要信号测试和拷机。

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。

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。

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。

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

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。

RV1109/RV1126/RK356x DDR频率选择

对于RV1109/RV1126/RK356x平台loader有变4次频率，保存着相应频率的training结果。kernel中dmc所配的4个频率点需要与loader的频率点保持一致。RV1126/RV1109 loader中默认的频率点其中三个是328M，528M，784M，最高的那个频率点在ddr bin名称中体现，如rv1126_ddr_924MHz_v1.05.bin，最后一个频点则是924M。RK356x loader中默认的频率点其中三个是324M，528M，780M，最高那个频率点一样在ddr bin名称中体现，如rk3568_ddr_1560MHz_v1.04.bin，最后一个频率点则是1560M。另外loader中的频率点也可通过串口log查看，串口log中有重复4次change to：xxxMHz字样代表着包含的4个频率点信息。loader的频率点也可通过rkbin目录下tools/ddrbin_tool来修改，具体使用规则可参考tools/ddrbin_tool_user_guide.txt。

例如RK3568，如果最高频率需要改成1332M，需要在rkbin工程目录下将RKBOOT/RK3568MINIALL.ini中的Path1和FlashData指向的ddr bin改为1332M，并且dts中的频率点也对应的改成324M,528M,780M,1332M。实际运行的频率点只能是这4个中的一个。

RK3568 ECC的使能

RK3568支持ECC，如果DDR ECC DQ0-7有接颗粒的话loader会自动enable ECC功能。 需要注意的是ECC byte上所贴的颗粒需要与DQ0-31上所贴的颗粒具有一样的row/bank/col。