DDR开发指南

发布版本:1.0

作者邮箱：hcy@rock-chips.com

日期:2017.12.21

文件密级：公开资料

前言

适用于所有平台的开发指南

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	何灿阳	

怎么判断**DDR**已经进入自刷新（**self-refresh**省电模式）
怎么判断**DDR**已经进入**auto power-down**省电模式
如何调整**DQ**、**DQS**、**CA**、**CLK**的**skew**

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。有得芯片没加这个功能，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，DDR不一定没问题。
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:开头的打印结束，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uboot工程的tools\rk_tools\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uboot（见uboot根目录下的UserManual），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
3036	tools\rk_tools\bin\rk30\rk3036_ddr3_XXXMHz_vX.XX.bin	1
3126、3126B、3126C	tools\rk_tools\bin\rk31\rk3126_ddr3_300MHz_vX.XX.bin	
3128	tools\rk_tools\bin\rk31\rk3128_ddr_300MHz_vX.XX.bin	
3288	tools\rk_tools\bin\rk32\rk3288_ddr_400MHz_vX.XX.bin	
322x	tools\rk_tools\bin\rk32\rk322x_ddr_300MHz_vX.XX.bin	
3368	tools\rk_tools\bin\rk33\rk3368_ddr_600MHz_vX.XX.bin	
322xh	tools\rk_tools\bin\rk33\rk322xh_ddr_333MHz_vX.XX.bin	
3328	tools\rk_tools\bin\rk33\rk3328_ddr_333MHz_vX.XX.bin	
3399	tools\rk_tools\bin\rk33\rk3399_ddr_XXXMHz_vX.XX.bin	2

Note 1：具体用哪个频率由tools\rk_tools\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tools\rk_tools\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uboot中生成loader。

如何修改uboot中的DDR频率

目前这个功能只有322x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。

uboot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。举例如下，3399 EVB板的最终dmc节点在arch/arm64/boot/dts/rockchip/rk3399-evb.dtsi中，

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

```
&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，3288 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，就不会做DDR变频了。举例如下，3066 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变频都不做

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，会做一次变频，用于更新DDR timing，让性能更高。如果想让kernel中一次DDR变频都不做，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变频都不会做了

- 对于kernel 3.10

  芯片：322x

  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：3188

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

  芯片：3288

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

  芯片：3126B、3126C非trust流程

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

  芯片：3126、3128

  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：1108

  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，剩下芯片都是走trust流程，包括3126B/3126c走trust流程的，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变频都不会做了。

- 对于kernel 3.0

芯片	代码位置
3066	arch/arm/mach-rk30/ddr.c的ddr_init()函数
3026、3028A	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**
3026	有详细信息	有详细信息
3028A	有详细信息	有详细信息
3036	有详细信息	没有
3066	有详细信息	有详细信息
3126B/3126C走trust流程	有详细信息	没有
3126B/3126C非trust流程	有详细信息	有详细信息
3126	有详细信息	有详细信息
3128	有详细信息	有详细信息
3188	有详细信息	有详细信息
3288	有详细信息	有详细信息
322x	有详细信息	没有
322xh	有详细信息	没有
3328	有详细信息	没有
3368	有详细信息	没有
3399	有详细信息	没有
1108	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，一种是负载变频。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，3399 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";
};
```

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，3288 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频率在这里找不到对应电压，也会无法切换到对应频率。可以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。

- 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，3066 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。

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：

3399： `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，3399 EVB板的在arch/arm64/boot/dts/rockchip/rk3399-opp.dtsi、3368在arch/arm64/boot/dts/rockchip/rk3368.dtsi，以3399为例。

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

    opp-200000000 {
        opp-hz = /bits/ 64 <200000000>; //当DDR频率小于等于200MHz时，使用这个电压
        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>;
    };
};
```

3368为例：

```
dmc_opp_table: opp_table2 {
    compatible = "operating-points-v2";

    opp-192000000 {
        opp-hz = /bits/ 64 <192000000>; //当DDR频率小于等于200MHz时，使用这个电压
        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，3288 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，3066 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，3399 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，3288 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 3.10，需要找到dts中的clk_ddr_dvfs_table节点。举例如下，3288 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. 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

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，3066 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. 注释掉其他场景，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，跑google stressapptest，碰到报错，就降低频率，再跑。没报错，可以多跑一段时间，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，这里再跑一个memtester。方法一样，碰到报错，就降低频率，再跑。没报错，可以多跑一段时间，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skew

要调整kernel中的skew，目前也只有3228h、3328支持。需要改对应的dts文件

芯片：322xh、3328

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“3228h_ddr_scan_eye.sh”工具自动扫出来的结果，选择mid值，修改到对应dts定义中。