RK322X/RK3328 HDMI-PHY-PLL配置指南

文档标识：RK-KF-YF-108

发布版本：V1.1.0

日期：2020-04-09

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，福州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。

福州瑞芯微电子股份有限公司

Fuzhou 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

本文档主要介绍 RK322X/RK3328 芯片 HDMI PHY PLL 的计算以及相关寄存器配置。主要用于软件开发工程师或 FAE 对客户支持，并不对 PHY 进行深入介绍。

概述

产品版本

芯片名称	内核版本
RK322X/RK3328	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
2020-03-26	V1.0.0	操瑞杰	初始发布
2020-04-09	V1.1.0	操瑞杰	修改文档标题、措辞、对新增PLL配置的说明进行补充

RK322X/RK3328 HDMI-PHY-PLL配置指南

1. 概述

RK322X/RK3328 芯片 HDMI PHY PLL 除了供 HDMI PHY 使用外，还作为显示的时钟源，供 HDMI/CVBS/VOP 使用。在产品开发设计中，当需要增加对某些特殊分辨率的支持时，需要新增 PHY PLL 配置，以便 PHY 能输出该分辨率对应的时钟频率。

2. PHY配置计算说明

PLL 计算包括两部分，PRE-PLL 和 POST-PLL 计算。

2.1 PRE-PLL计算过程

PRE-PLL 计算过程如图所示，其中 F_{REF} 为 24Mhz：

PRE-PLL($F_{pre-VCO}$） 的频率 VCO 是由 `pre-pll-pre-divider[5:0]` 和 `pre-pll-feedback-divider[11:0]` 控制的。在 RK3328 芯片还可以由 `pre-pll-fractional-feedback-divider[23:0]` 控制，支持浮点运算。

$$F_{pre-VCO} = (F_{REF}/pre-pll-pre-divider[5:0]) \times (pre-pll-feedback-divider[11:0] + pre-pll-fractional-feedback-divider[23:0]/2^{24})$$

TMDS CLOCK 通道频率 F_{TX3} 由 `tmds-dividera[1:0]` 和 `tmds-dividerb[1:0]` 控制。

$$F_{TX3} = F_{pre-VCO}/(4 \times tmds-dividera[1:0] \times tmds-dividerb[1:0])$$

$$tmds-dividera[1:0] = 1, 2, 3, 5$$
$$tmds-dividerb[1:0] = 1, 2, 4, 8$$

`pin_hdmi20_tmdsclk` 频率 $F_{pin\_hdmi20\_tmdsclk}$ 由 `tmds-dividera[1:0]` 和 `tmds-dividerc[1:0]` 控制。

$$F_{pin\_hdmi20\_tmdsclk} = F_{pre-VCO}/(4 \times tmds-dividera[1:0] \times tmds-dividerc[1:0])$$

$$tmds-dividera[1:0] = 1, 2, 3, 5$$
$$tmds-dividerc[1:0] = 1, 2, 4, 8$$

`pin_hdmi20_prepclk` 频率 $F_{pin\_hdmi20\_prepclk}$ 计算方法如下：

$$F_{pin\_hdmi20\_prepclk} = (pclk-dividera[4:0] == 1) ? \{F_{pre-VCO}/(pclk-dividerb[1:0] \times pclk-dividerc[4:0])\} : \{F_{pre-VCO}/(pclk-dividera[4:0] \times pclk-dividerc[4:0])\}$$

$$pclk-dividera[4:0] = 1 \sim 31$$
$$pclk-dividerb[1:0] = 2, 3, 4, 5$$
$$pclk-dividerc[1:0] = 1, 2, 4, 8$$

当 `pin_hdmi20_pclk` 的频率 ($F{pin_hdmi20_pclk}$) 为 $F{pre-VCO}$ 的 1/5 时，可以使能 VCO/5 直接获取`pin_hdmi20_pclk`。其他频率计算如下：

$$
\begin{aligned}
&F_{pin\_hdmi20\_pclk} = (pclk - dividera[4:0] == 1)\ ? \\
&\{F_{pre-VCO}/(pclk - dividerb[1:0] \times pclk - dividerc[4:0]) \times 2\}\ : \\
&\{F_{pre-VCO}/(pclk - dividera[4:0] \times pclk - dividerc[4:0]) \times 2\}
\end{aligned}
$$

$$
\begin{aligned}
&pclk - dividera[4:0] = 1 \sim 31 \\
&pclk - dividerb[1:0] = 2, 3, 4, 5 \\
&pclk - dividerd[4:0] = 1 \sim 31
\end{aligned}
$$

其中需要注意两个 CLOCK， `pin_hd20_tmdsclk` 为 HDMI 输出的 TMDS CLOCK， `pin_hd20_pclk` 为 HDMI 输出的 PIXEL CLOCK。此外还有需要注意的地方：

1. `pin_hd20_pclk` 频率是由 `pin_hd20_prepclk` 以 1~10 的倍数分频得到。在 `no-repeating` 模式下，两种频率相等。
2. `pin_hd20_prepclk` 和 `pin_hd20_tmdsclk` 的频率在 8-bit 色深下相等。在 10-bit 色深下，`pin_hd20_tmdsclk` 等于 1.25 倍 `pin_hd20_prepclk` 频率。目前 RK 平台仅支持 8-bit 和 10-bit 两种色深。

如其中第一点所述，实际使用中，在非 `4K-YUV420` 的场景下， `pin_hd20_pclk` 与 `pin_hd20_prepclk` 始终相等。`4K-YUV420` 的场景下 `pin_hd20_pclk` 是 `pin_hd20_prepclk` 的两倍。

2.2 POST-PLL计算过程

POST-PLL 计算过程如下：

POST-PLL 用来产生差分串行时钟，其频率 $F_{diff-sclk}$ 总是 `pin_hdmi20_tmdsclk` 的五倍。

$$
\begin{aligned}
F_{post-VCO} = &(F_{pin\_hdmi20\_tmdsclk}/post - pll - pre - divider[4:0]) \times \\
&post - pll - feedback - divider[8:0]
\end{aligned}
$$

当 `post-pll-post-divider` 未使能时（0xaa[3:2] = 0），$F_{diff-sclk}$ 与 $F_{post-VCO}$ 相等。否则计算方法为：

$$
F_{diff-sclk} = F_{post-VCO}/post - pll - post - divider[1:0]
$$

$$
post - pll - post - divider = 2, 4, 8(2'b0, 2'b01, 2'b11)
$$

POST-PLL 计算较为简单，且 LINUX 4.4/4.19 内核驱动中的 `post_pll_cfg_table` 已经包含所有场景，无需另行添加。

LINUX 3.10 内核需要根据当前需要支持分辨率对应的的 TMDS CLOCK 所处的区间以及芯片的版本，直接使用`post_pll_cfg_table`当中相应的配置。具体说明请见 3.2 节。

需要注意的是，无论是 PRE-PLL 还是 POST-PLL，VCO 的频率都必须在 1.4~3.0Ghz 之间（实际使用中最大可以放宽到 3.2Ghz）。

2.3 计算工具的使用

实际工作中可以使用计算工具 cal_innophy 进行计算，该计算工具可通过 RK 技术支持窗口获取。

使用方法为：

```
cal_innophy 148500000 185625000 1
```

其中三个参数的含义分别为

参数	说明
148500000	PIXLE CLOCK
185625000	TMDS CLOCK
1	是否使用浮点计算，RK322X系列芯片不支持浮点计算只能为0

其中第三项参数，推荐优先为 0，不使用浮点计算。当不使用浮点计算不出时，再设置为 1 进行计算。

TMDS CLOCK在不同色深的情况下，与PIXLE CLOCK的比率不同，详见2.1节。

计算结果为：

```
148500000, 185625000, 4, 495, 0, 2, 2, 1, 3, 2, 2, 0, 0x816817
```

各项参数含义为：

参数	说明
148500000	pixel clock
185625000	tmds clock
4	pre-pll-pre-divider
495	pre-pll-feedback-divider
0	tmds-dividera
2	tmds-dividerb
2	tmds-dividerc
1	tmds-dividerd
3	pclk-dividera
2	pclk-dividerb
2	pclk-dividerc
0	pclk-dividerd
0x816817	pre-pll-fractional-feedback-divider

计算结果分别对应 2.1 节中说明的各项寄存器配置。如何在驱动中新增计算出的配置请见本文第 3 节。本工具只计算了 PRE-PLL 的配置。

针对 POST-PLL，当 TMDS CLOCK <= 74.25MHz 时，RK322X 和 RK3328 早期样片配置一致，但与 RK3328 量产芯片的配置存在差异，需要在芯片版本上做区分。

在 LINUX 4.4/4.19 内核中，POST-PLL 的配置较为简单，已经包含在了 `post_pll_cfg_table` 当中，驱动会查找合适的配置，无需在驱动中另行添加新的配置。

LINUX 3.10 内核的 POST-PLL 配置分为 `RK322XH_V1_PLL_TABLE` 和 `EXT_PLL_TABLE` 两个 TABLE。可以根据所需 TMDS CLOCK 以及当前使用芯片的版本，直接选用 `post_pll_cfg_table` 当中对应值，在相应的 TABLE 中添加配置，详见 3.2 节。

3. 代码中新增PHY配置说明

LINUX 3.10 内核以及 LINUX 4.4/4.19 内核将 PHY 的配置保存在对应的 TABLE 中，根据当前输出分辨率的 PIXEL CLOCK 和 TMDS CLOCK 选择对应的配置并设置到对应的寄存器中。相同分辨率的不同色深模式的配置不同（TMDS CLOCK 不同），所以如果需要同时支持 8-bit 和 10-bit 的色深，需要在 TABLE 中添加两项配置，详见2.1节。

3.1 LINUX 3.10内核

LINUX 3.10 内核驱动 HDMI 需要在特定的 TABLE 中新增对应的配置项，路径为：

```
kernel/drivers/video/rockchip/hdmi/rockchip-hdmiv2/rockchip_hdmiv2_hw.c
```

LINUX 3.10 内核驱动中包含两个 TABLE， `RK322XH_V1_PLL_TABLE` 适用于 TMDS CLOCK <= 74.25MHz 且所用芯片为 RK3328 量产芯片的场景。 `EXT_PLL_TABLE` 适用于 TMDS CLOCK > 74.25MHz 且所用芯片为 RK3328 早期样片和 RK322X 的场景。

```
static const struct ext_pll_config_tab RK322XH_V1_PLL_TABLE[] = {
        {27000000,      27000000,       8,      1,      90,     3,      2,
                2,      10,     3,      3,      4,      0,      1,      80,
                8,      0xE8FBA7},
        {27000000,      33750000,       10,     1,      90,     1,      3,
                3,      10,     3,      3,      4,      0,      1,      80,
                8,      0xE8FBA7},
        {59400000,      59400000,       8,      1,      99,     3,      1,
                1,      1,      3,      3,      4,      0,      18,     80,
                8,      0xE6AE6B},
        {59400000,      74250000,       10,     1,      99,     0,      3,
                3,      1,      3,      3,      4,      0,      18,     80,
                8,      0xE6AE6B},
        {74250000,      74250000,       8,      1,      99,     1,      2,
                2,      1,      2,      3,      4,      0,      18,     80,
                8,      0xE6AE6B},
};

static const struct ext_pll_config_tab EXT_PLL_TABLE[] = {
        {27000000,      27000000,       8,      1,      90,     3,      2,
                2,      10,     3,      3,      4,      0,      1,      40,
                8,      0xE8FBA7},
        {27000000,      33750000,       10,     1,      90,     1,      3,
                3,      10,     3,      3,      4,      0,      1,      40,
                8,      0xE8FBA7},
        {59400000,      59400000,       8,      1,      99,     3,      1,
                1,      1,      3,      3,      4,      0,      1,      40,
                8,      0xE6AE6B},
        {59400000,      74250000,       10,     1,      99,     0,      3,
                3,      1,      3,      3,      4,      0,      1,      40,
                8,      0xE6AE6B},
        {74250000,      74250000,       8,      1,      99,     1,      2,
                2,      1,      2,      3,      4,      0,      1,      40,
                8,      0xE6AE6B},
        {74250000,      92812500,       10,     4,      495,    1,      2,
```

```
                2,      1,      3,      3,      4,      0,      2,      40,
                4,      0x816817},
        {148500000,     148500000,      8,      1,      99,     1,      1,
                1,      1,      2,      2,      2,      0,      2,      40,
                4,      0xE6AE6B},
        {148500000,     185625000,      10,     4,      495,    0,      2,
                2,      1,      3,      2,      2,      0,      4,      40,
                2,      0x816817},
        {297000000,     297000000,      8,      1,      99,     0,      1,
                1,      1,      0,      2,      2,      0,      4,      40,
                2,      0xE6AE6B},
        {297000000,     371250000,      10,     4,      495,    1,      2,
                0,      1,      3,      1,      1,      0,      8,      40,
                1,      0x816817},
        {594000000,     297000000,      8,      1,      99,     0,      1,
                1,      1,      0,      2,      1,      0,      4,      40,
                2,      0xE6AE6B},
        {594000000,     371250000,      10,     4,      495,    1,      2,
                0,      1,      3,      1,      1,      1,      8,      40,
                1,      0x816817},
        {594000000,     594000000,      8,      1,      99,     0,      2,
                0,      1,      0,      1,      1,      0,      8,      40,
                1,      0xE6AE6B},
};
```

`struct ext_pll_config_tab` 的定义为：

```
struct ext_pll_config_tab {
        u32     pix_clock;
        u32     tmdsclock;
        u8      color_depth;
        u8      pll_nd;
        u16     pll_nf;
        u8      tmsd_divider_a;
        u8      tmsd_divider_b;
        u8      tmsd_divider_c;
        u8      pclk_divider_a;
        u8      pclk_divider_b;
        u8      pclk_divider_c;
        u8      pclk_divider_d;
        u8      vco_div_5;
        u8      ppll_nd;
        u16     ppll_nf;
        u8      ppll_no;
        u32     frac;
};
```

各项参数说明见下表：

参数	说明
pix_clock	HDMI输出分辨率的pixel clock
tmdsclock	HDMI输出分辨率的tmds clock
color_depth	HDMI输出色深8/10 bits
pll_nd	pre-pll-pre-divider
pll_nf	pre-pll-feedback-divider
tmsd_divider_a	tmds-dividera
tmsd_divider_b	tmds-dividerb
tmsd_divider_c	tmds-dividerc
pclk_divider_a	pclk-dividera
pclk_divider_b	pclk-dividerb
pclk_divider_c	pclk-dividerc
pclk_divider_d	pclk-dividerd
vco_div_5	pin_hd20_pclk是否直接由VCO / 5所得，特定clock情况下使用
ppll_nd	post-pll-pre-divider
ppll_nf	post-pll-feedback-divider
ppll_no	post-pll-post-divider
frac	pre-pll-fractional-feedback-divider

ppll_nd，ppll_nf，ppll_no 对应 LINUX 4.19/4.4 内核 `post_pll_config` 中的 prediv，fbdiv，postdiv。其值的选择需要根据 TMDS CLOCK 以及芯片版本，选择方法详见 3.2 节。

需要注意的是，部分版本的sdk代码较旧不包括 frac。RK322X 系列的芯片 PHY 的版本不支持浮点，所以 frac 只能为 0。

3.2 LINUX 4.4/4.19内核

LINUX 4.4/4.19 内核需要新增 PHY 配置时，需在 PRE-PLL 的配置 TABLE：`pre_pll_cfg_table` 中新增对应的配置，而 POST-PLL 的配置 TABLE：`post_pll_cfg_table` 目前配置已经涵盖了 PHY 所支持的所有分辨率，无需再新增配置。路径为：

```
kernel/drivers/phy/rockchip/phy-rockchip-inno-hdmi-phy.c
```

```
static const struct pre_pll_config pre_pll_cfg_table[] = {
    { 27000000,  27000000, 1,  90, 3, 2, 2, 10, 3, 3,  4, 0, 0},
    { 27000000,  33750000, 1,  90, 1, 3, 3, 10, 3, 3,  4, 0, 0},
    { 40000000,  40000000, 1,  80, 2, 2, 2, 12, 2, 2,  2, 0, 0},
    { 40000000,  50000000, 1, 100, 2, 2, 2,  1, 0, 0, 15, 0, 0},
    { 59341000,  59341000, 1,  98, 3, 1, 2,  1, 3, 3,  4, 0, 0xE6AE6B},
```

```
    { 59400000,  59400000, 1,  99, 3, 1, 1,  1, 3, 3,  4, 0, 0},
    { 59341000,  74176250, 1,  98, 0, 3, 3,  1, 3, 3,  4, 0, 0xE6AE6B},
    { 59400000,  74250000, 1,  99, 1, 2, 2,  1, 3, 3,  4, 0, 0},
    { 65000000,  65000000, 1, 130, 2, 2, 2,  1, 0, 0, 12, 0, 0},
    { 65000000,  81250000, 3, 325, 0, 3, 3,  1, 0, 0, 10, 0, 0},
    { 71000000,  71000000, 3, 284, 0, 3, 3,  1, 0, 0,  8, 0, 0},
    { 71000000,  88750000, 3, 355, 0, 3, 3,  1, 0, 0, 10, 0, 0},
    { 74176000,  74176000, 1,  98, 1, 2, 2,  1, 2, 3,  4, 0, 0xE6AE6B},
    { 74250000,  74250000, 1,  99, 1, 2, 2,  1, 2, 3,  4, 0, 0},
    { 74176000,  92720000, 4, 494, 1, 2, 2,  1, 3, 3,  4, 0, 0x816817},
    { 74250000,  92812500, 4, 495, 1, 2, 2,  1, 3, 3,  4, 0, 0},
    { 83500000,  83500000, 2, 167, 2, 1, 1,  1, 0, 0,  6, 0, 0},
    { 83500000, 104375000, 1, 104, 2, 1, 1,  1, 1, 0,  5, 0, 0x600000},
    { 85750000,  85750000, 3, 343, 0, 3, 3,  1, 0, 0,  8, 0, 0},
    { 88750000,  88750000, 3, 355, 0, 3, 3,  1, 0, 0,  8, 0, 0},
    { 88750000, 110937500, 1, 110, 2, 1, 1,  1, 1, 0,  5, 0, 0xF00000},
    {108000000, 108000000, 1,  90, 3, 0, 0,  1, 0, 0,  5, 0, 0},
    {108000000, 135000000, 1,  90, 0, 2, 2,  1, 0, 0,  5, 0, 0},
    {119000000, 119000000, 1, 119, 2, 1, 1,  1, 0, 0,  6, 0, 0},
    {119000000, 148750000, 1,  99, 0, 2, 2,  1, 0, 0,  5, 0, 0x2AAAAA},
    {148352000, 148352000, 1,  98, 1, 1, 1,  1, 2, 2,  2, 0, 0xE6AE6B},
    {148500000, 148500000, 1,  99, 1, 1, 1,  1, 2, 2,  2, 0, 0},
    {148352000, 185440000, 4, 494, 0, 2, 2,  1, 3, 2,  2, 0, 0x816817},
    {148500000, 185625000, 4, 495, 0, 2, 2,  1, 3, 2,  2, 0, 0},
    {162000000, 162000000, 1, 108, 0, 2, 2,  1, 0, 0,  4, 0, 0},
    {162000000, 202500000, 1, 135, 0, 2, 2,  1, 0, 0,  5, 0, 0},
    {296703000, 296703000, 1,  98, 0, 1, 1,  1, 0, 2,  2, 0, 0xE6AE6B},
    {297000000, 297000000, 1,  99, 0, 1, 1,  1, 0, 2,  2, 0, 0},
    {296703000, 370878750, 4, 494, 1, 2, 0,  1, 3, 1,  1, 0, 0x816817},
    {297000000, 371250000, 4, 495, 1, 2, 0,  1, 3, 1,  1, 0, 0},
    {593407000, 296703500, 1,  98, 0, 1, 1,  1, 0, 2,  1, 0, 0xE6AE6B},
    {594000000, 297000000, 1,  99, 0, 1, 1,  1, 0, 2,  1, 0, 0},
    {593407000, 370879375, 4, 494, 1, 2, 0,  1, 3, 1,  1, 1, 0x816817},
    {594000000, 371250000, 4, 495, 1, 2, 0,  1, 3, 1,  1, 1, 0},
    {593407000, 593407000, 1,  98, 0, 2, 0,  1, 0, 1,  1, 0, 0xE6AE6B},
    {594000000, 594000000, 1,  99, 0, 2, 0,  1, 0, 1,  1, 0, 0},
    {     ~0UL,     0, 0,   0, 0, 0, 0,  0, 0, 0,  0, 0, 0}
};

static const struct post_pll_config post_pll_cfg_table[] = {
    {33750000,  1, 40, 8, 1},
    {33750000,  1, 80, 8, 2},
    {33750000,  1, 10, 2, 4},
    {74250000,  1, 40, 8, 1},
    {74250000, 18, 80, 8, 2},
    {148500000, 2, 40, 4, 3},
    {297000000, 4, 40, 2, 3},
    {594000000, 8, 40, 1, 3},
    {     ~0UL, 0,  0, 0, 0}
};
```

`struct pre_pll_config` 和 `struct post_pll_config` 定义如下，LINUX 4.4/4.19 内核相当于拆分了 3.10 内核中的 `struct ext_pll_config_tab`。

```
struct pre_pll_config {
    unsigned long pixclock;
    unsigned long tmdsclock;
```

```
    u8 prediv;
    u16 fbdiv;
    u8 tmds_div_a;
    u8 tmds_div_b;
    u8 tmds_div_c;
    u8 pclk_div_a;
    u8 pclk_div_b;
    u8 pclk_div_c;
    u8 pclk_div_d;
    u8 vco_div_5_en;
    u32 fracdiv;
};

struct post_pll_config {
    unsigned long tmdsclock;
    u8 prediv;
    u16 fbdiv;
    u8 postdiv;
    u8 version;
};
```

`pre_pll_config` 各项参数说明见下表：

参数	说明
pixclock	HDMI输出分辨率的pixel clock
tmdsclock	HDMI输出分辨率的tmds clock
prediv	pre-pll-pre-divider
fbdiv	pre-pll-feedback-divider
tmds_div_a	tmds-dividera
tmds_div_b	tmds-dividerb
tmds_div_c	tmds-dividerc
pclk_div_a	pclk-dividera
pclk_div_b	pclk-dividerb
pclk_div_c	pclk-dividerc
pclk_div_d	pclk-dividerd
vco_div_5_en	pin_hd20_pclk是否直接由VCO / 5所得，特定clock情况下使用
fracdiv	pre-pll-fractional-feedback-divider

`post_pll_config` 各项参数说明见下表：

参数	说明
tmdsclock	HDMI输出分辨率的tmds clock
prediv	post-pll-pre-divider
fbdiv	post-pll-feedback-divider
postdiv	post-pll-post-divider
version	芯片版本，POST-PLL配置需根据时钟和芯片版本确定，其值含义： 1--RK322X与RK322XH早期样片，tmds clock为74.25Mhz及以下的配置 2--RK322XH量产芯片，tmds clock为74.25Mhz及以下的配置 3--RK322X与RK322XH芯片，tmds clock为74.25Mhz以上的配置，两者配置相同 4--RK322X部分芯片POST VCO为1080Mhz时不稳定，为270Mhz时工作稳定，需要特别区分出来

以 TMDS CLOCK 为 74.25Mhz，RK3328 量产芯片为例，POST-PLL 配置选择方法如下：

1. 首先在 `post_pll_cfg_table` 中根据 TMDS CLOCK 找到对应的区间。如 TMDS CLOCK 为 74.25Mhz 时，33.75Mhz < TMDS CLOCK <= 74.25Mhz,找到对应的二项：

```
{74250000,  1, 40, 8, 1},
{74250000, 18, 80, 8, 2},
```

2. 根据芯片版本进一步选择，此时是 RK3328 量产芯片，TMDS CLOCK <= 74.25Mhz，所以 version 的值应选择 2，所以最终选择：

```
{74250000, 18, 80, 8, 2},
```

3. 最终配置值为：prediv = 18，fbdiv = 80， postdiv = 8。在 LINUX 3.10 内核的驱动中对应 `struct ext_pll_config_tab` 中的 ppll_nd, ppll_nf, ppll_no 三项。由于是 RK3328 量产芯片切 TMDS CLOCK <= 74.25Mhz，所以需要添加在`RK322XH_V1_PLL_TABLE` 当中。