AM335x 关于 LCD 屏幕的配置

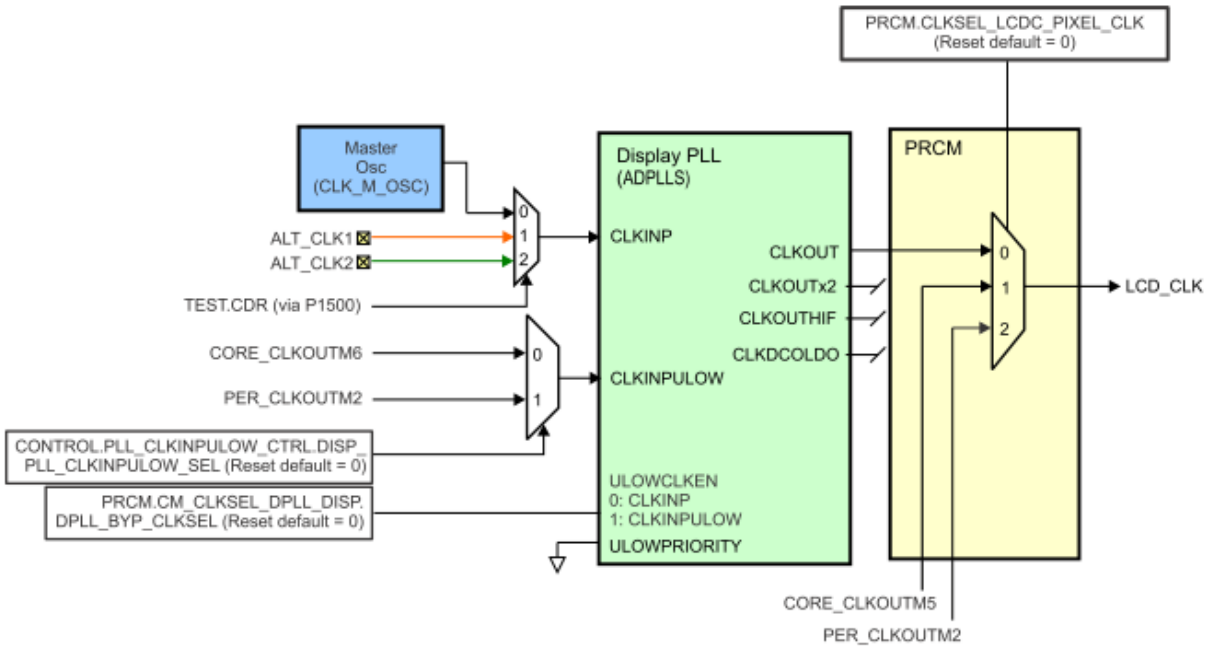
----Steven Liu

很多朋友在配置 AM335x 的 LCD 屏幕的时候会发现有频率配不出来，或者是不好配的情况，因为感觉他的时钟树过于繁琐，这里，正好抽空整理了一下这部分的内容（主要参考的是 AM335x 的 TRM 的第 8 章 PRCM 模块和 13 章 LCD Controller。

这里在 LCD Controller 里面的配置描述的比较详细了，分频和像素、消影值的设置等等。不在赘述，很多人都会抱怨说，LCD_PCLK 配置只能通过 LCD_CLK 经过一个分频而来，这样对于频率 70~90MHz 时配置很困难。但事实上，我们对 LCD_CLK 的设置，是比较灵活的，参考如下：

8.1.6.10 节中：Display PLL Description

Figure 8-13. Display PLL Structure



左边的部分我会慢慢解释，先看右边。右边的 LCD_CLK 就是最终给到 LCD 模块的 LCD_CLK，就是对接到下面的框图中。

但下面的框图中有个地方描述的容易让人混淆，就是被黄色部分框住的地方，这个时钟源的选取不是唯一的，如上图所示，LCD_CLK 并不一定要使用 Display PLL 的 CLKOUT，还可以使用 CORE_CLKOUTM5 或者 PER_CLKOUTM2 作为时钟。

13 章：LCD Controller 部分：

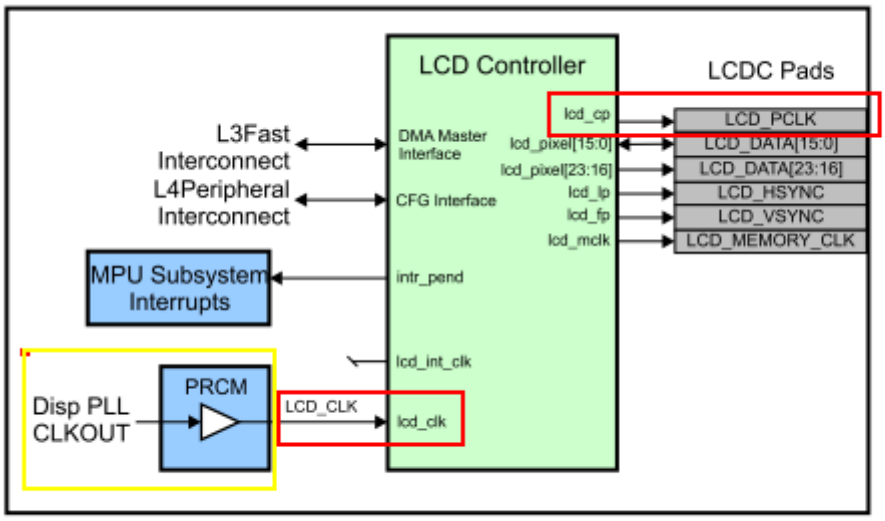


Figure 13-2. LCD Controller Integration

重点来了，在 StarterKit EVM 板上，基于 Starterware 的 LCD 显示例程，就是使用的 PER_CLKOUTM2 作为时钟源，该时钟为 192MHz（其实 PER_CLKOUTM2 时钟也是可以灵活配置的，但是不推荐，因为他是属于时钟数比较核心的时钟，如果这个一改变，其他的外设也要跟着变，影响比较大，所以对于该路时钟老老实实用 192MHz 就好，这方面 TRM 的描述可以参考 TRM 的 Table 8-24. Per PLL Typical Frequencies(MHz) ），所以这给很多人的直观印象就是我们的 LCD_CLK 只能从 192MHz 分频，事实上还是可以有更多选择的。

比如使用 CORE_CLKOUTM5，这路时钟是 250MHz 的（类似于 PER_CLKOUTM2，也可更改配置但不推荐，参考 TRM 的 Table 8-22. Core PLL Typical Frequencies (MHz)）。

再者还可以使用更为灵活的 Display PLL CLKOUT（后续再对这部分详解）。

所以 Clock 的配置选择还是很灵活的吧:). 那么如何设置选取不同的时钟源呢？很简单，在 CM_DPLL 寄存器（0x44E0_0500)偏移量为 34h 的 CLKSEL_LCDC_PIXEL_CLK 寄存器中，1-0 位就是对这个时钟的选择配置。

OK，到这里，是不是就觉得选择宽了？如果想要更加灵活的配置，剩下来的就是看你怎么去玩 Display PLL 的 clock 了，这部分参考 TRM 的 8.1.6.10 Display PLL Description 就可以了，里面也包括了怎么样配置。主要配置的地方就是 CM_WKUP 中(0x44E0_0400)的几个寄存器，在 TRM 的 8.1.6.10.1 Configuring the Display PLL 节中，有非常详细的描述，每一步怎么做都描述的很清楚了。有问题的朋友们可以留言，我再回复、更新到文章里。

测试实例：

平台：Beagle Bone

软件：StarterWare_02_00_01_01\build\armv7a\cgt_ccs\am335x\evmskAM335x\raster 工程

测试结果：

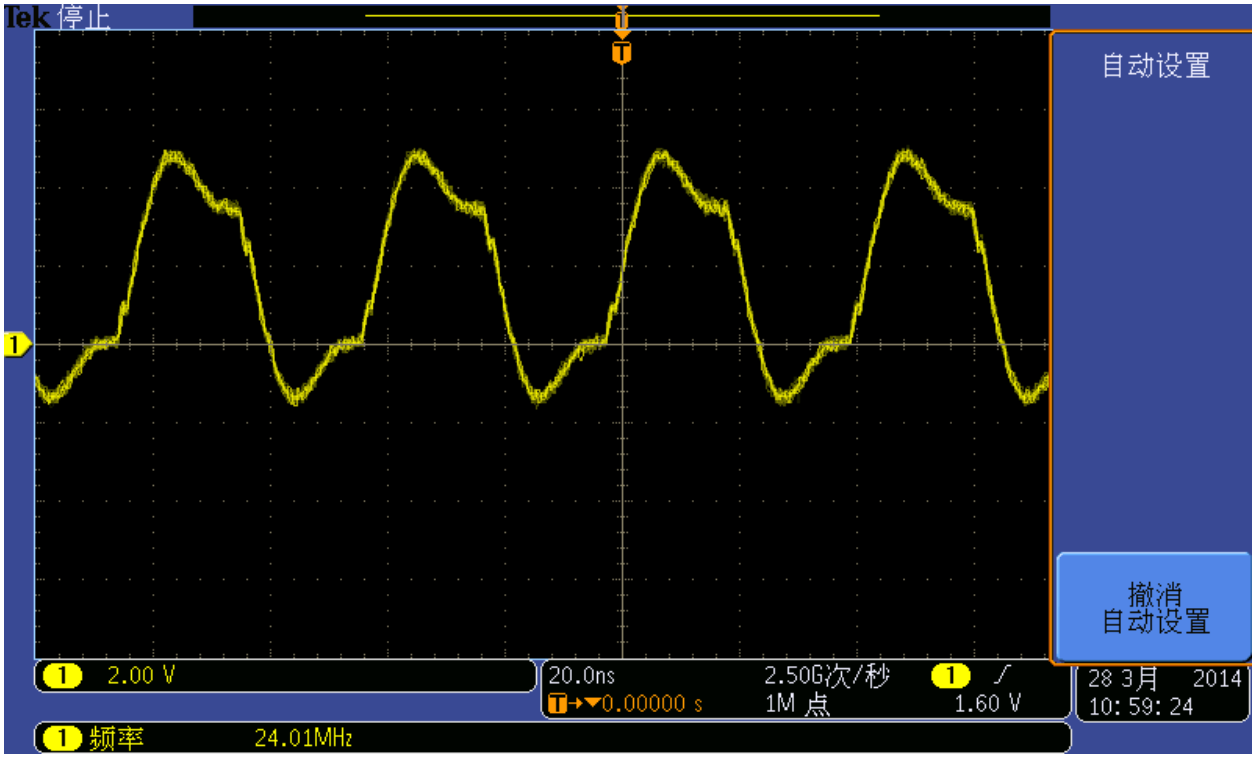
测试 1：

Raster 例程中使用 **PER_CLKOUTM2** 为时钟源（192MHz），并在 LCD 模块做了 8 分频，因此，可以得到的 LCD_PCLK=LCD_CLK/8=192/8=24MHz，测试结果如下：

Table 8-158. CLKSEL_LCDC_PIXEL_CLK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0h	
1-0	CLKSEL	R/W	0h	Controls the Mux Select of LCDC PIXEL clock 0x0 = SEL1 : Select DISP PLL CLKOUTM2 0x1 = SEL2 : Select CORE PLL CLKOUTM5 0x2 = SEL3 : Select PER PLL CLKOUTM2 0x3 = SEL4 : Reserved

```
0x44E00500   00000000 00000001 00000002 00000000 00000001
0x44E00514   00000004 00000001 00000000 00000000 00000000
0x44E00528   00000000 00000000 00000000 00000002 00000000
0x44E0053C   00000000 00000000 00000000 00000000 00000000
0x44E00550   00000000 00000000 00000000 00000000 00000000
```



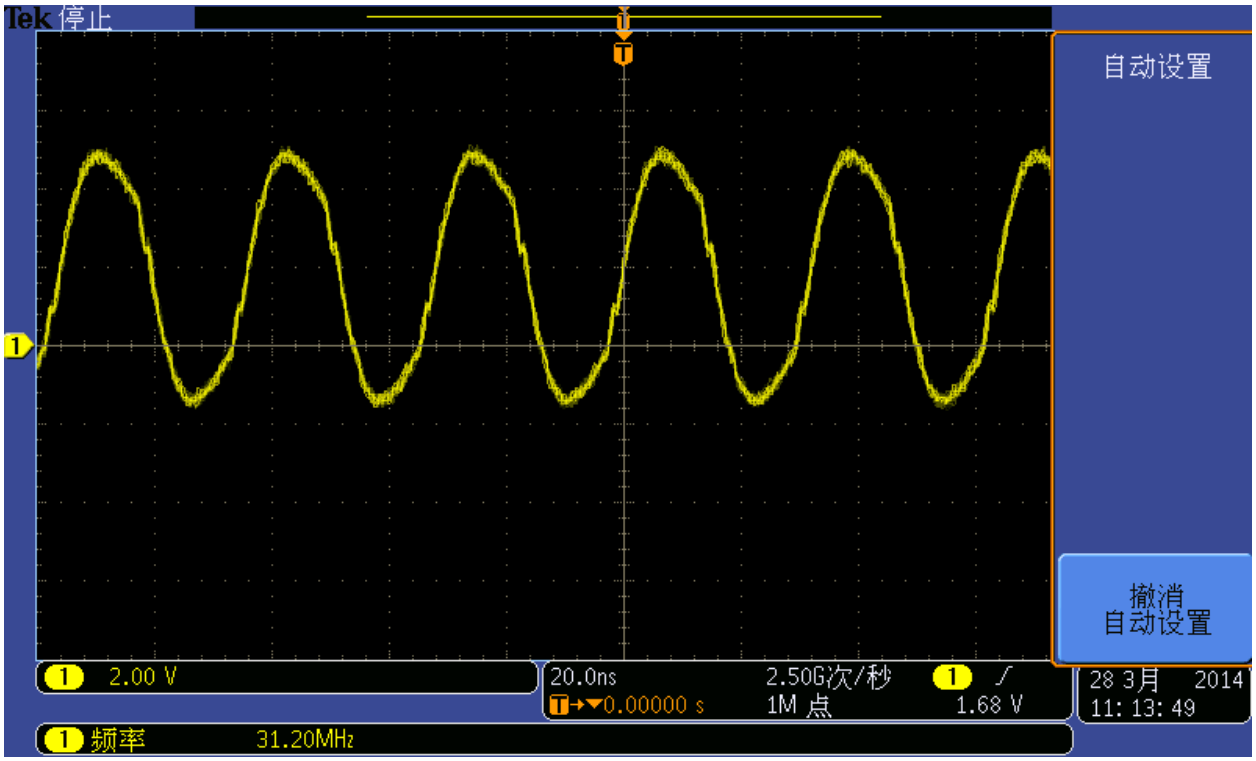
测试 2：

修改时钟源，变更为 **CORE_CLKOUTM5** 为时钟源（250MHz），其他条件不变，因此获得的 LCD_PCLK=LCD_CLK/8=250/8=31.25MHz，测试结果如下：

Table 8-158. CLKSEL_LCDC_PIXEL_CLK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0h	
1-0	CLKSEL	R/W	0h	Controls the Mux Select of LCDC PIXEL clock 0x0 = SEL1 : Select DISP PLL CLKOUTM2 0x1 = SEL2 : Select CORE PLL CLKOUTM5 0x2 = SEL3 : Select PER PLL CLKOUTM2 0x3 = SEL4 : Reserved

```
0x44E00500   00000000 00000001 00000002 00000000 00000001
0x44E00514   00000004 00000001 00000000 00000000 00000000
0x44E00528   00000000 00000000 00000000 00000001 00000000
0x44E0053C   00000000 00000000 00000000 00000000 00000000
0x44E00550   00000000 00000000 00000000 00000000 00000000
```



由此可以推测，我们是可以使用 CORE_CLKOUTM5 作为时钟源，最高可以达到 250MHz，与 TRM 中描述的略有出入。

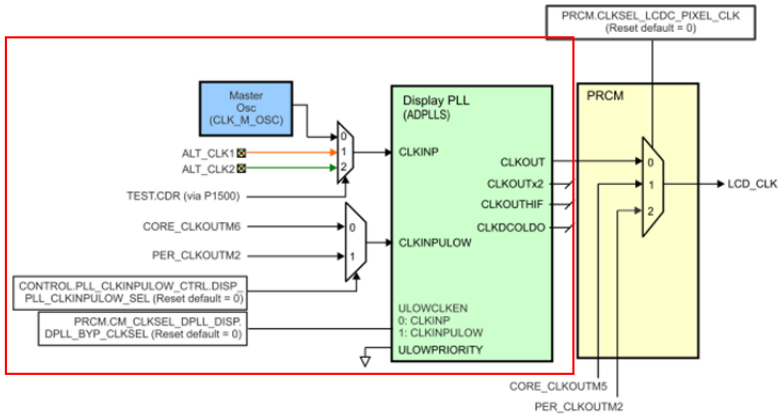
测试 3：

使用 Display PLL 作为时钟源：测试中得到的 LCD_PCLK=LCD_CLK/8;

在对 Display PLL 采用了 24MHz 的 clock 的时钟源，进行相关的配置：

LCD_CLK= Resource Clock * Mult / (N2 +1) /M2

Figure 8-13. Display PLL Structure

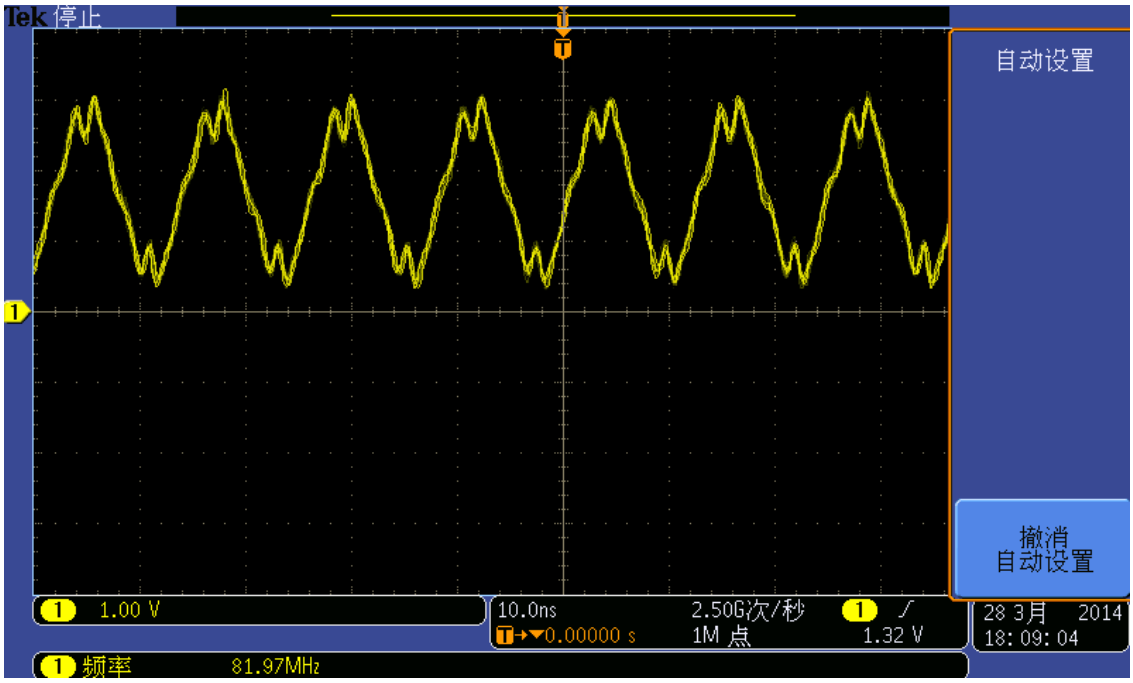


Test1:

Resource clock=24 ; Multi= 82 * 8; N2 =23; M2=1

LCD_CLK=24 * (82*8) / (23+1) /1=82*8MHz

LCD_PCLK=LCD_CLK/8=82MHz

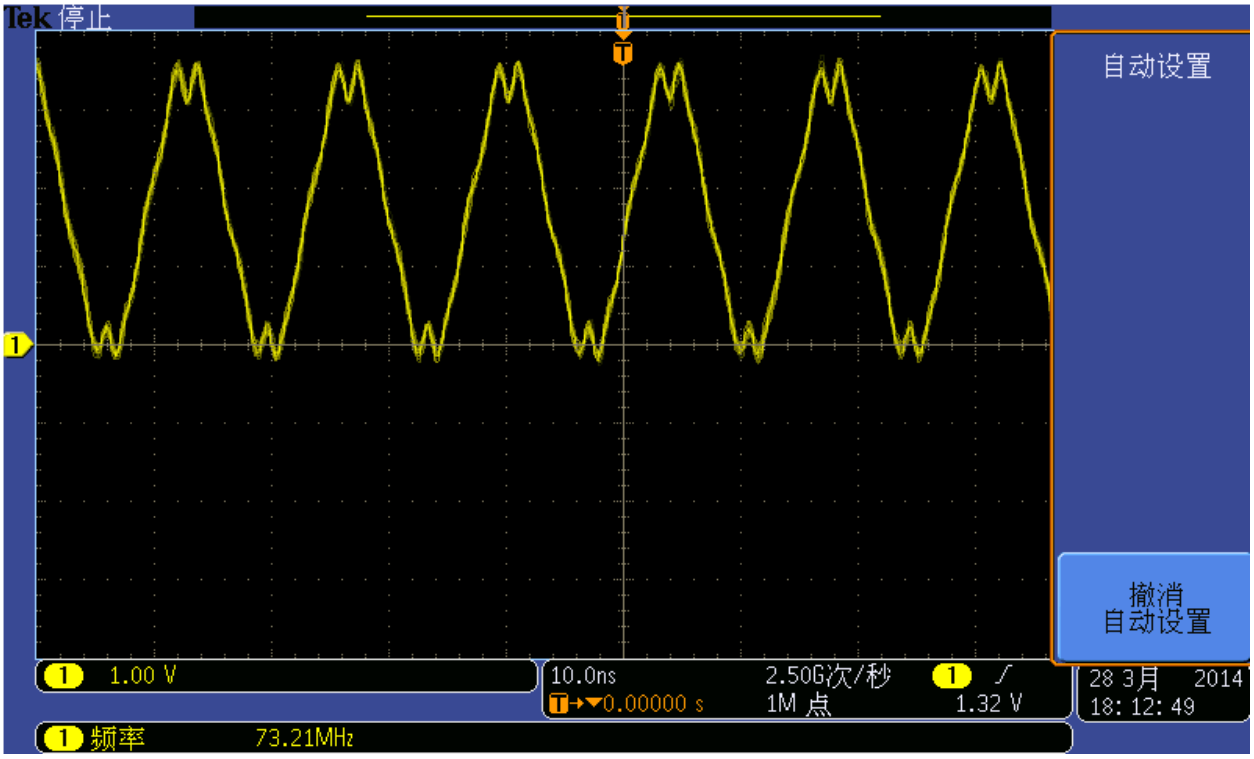


Test1:

Resource clock=24 ; Multi= 73 * 8; N2 =23; M2=1

LCD_CLK=24 * (73*8) / (23+1) /1=73*8MHz

LCD_PCLK=LCD_CLK/8=73MHz



参考代码：

```
  //choose the clock source:
    //PRCM.CM_CLKSEL_DPLL_DISP. DPLL_BYP_CLKSEL
    HWREG(0x44E00500 + 0x34) &= 0xfffffffc;         //

    HWREG(0x44E00400 + 0x54) &= 0xffff7fff ;        // 0->CLKINP; 1->CLKINPUTLOW

    // if CLKINPUTLOW, it works : CONTROL.PLL_CLKINPULOW_CTRL.DISP_PLL_CLKINPULOW_SEL
    //HWREG(0x44E10000 + 0x458)  |= (0x0 << 1) ;           //0-
>CORE_CLKOUT_M6(500MHz); 1->PER_CLK_OUT_M2(192MHz);


    //1. Switch PLL to bypass mode by setting CM_CLKMODE_DPLL_DISP.DPLL_EN to 0x4.
    HWREG(0x44E00400 + 0x98) = HWREG(0x44E00400 + 0x98) & 0xfffffffc | 0x4 ;

    //2. Wait for CM_IDLEST_DPLL_DISP.ST_MN_BYPASS = 1 to ensure PLL is in bypass
    //(CM_IDLEST_DPLL_DISP.ST_DPLL_CLK should also change to 0 to denote the PLL is
unlocked).
    while ((HWREG(0x44E00400 + 0x48) & (0x1 << 8)) != (0x1 << 8));
      //ST_MN_BYPASS
    while ((HWREG(0x44E00400 + 0x48) & 0x1) != 0x0);             //ST_DPLL_CLK

    //3. Configure Multiply and Divide values by setting
CM_CLKSEL_DPLL_DISP.DPLL_MULT and DPLL_DIV to the desired values.
    HWREG(0x44E00400 + 0x54) = HWREG(0x44E00400 + 0x54) & 0 | 0x17;            //DIV
[6-0]        /24
    //HWREG(0x44E00400 + 0x54) |= (0x290 << 8);     //MULT  [18-8]    *82*8
    HWREG(0x44E00400 + 0x54) |= (0x248 << 8);       //MULT  [18-8]    *73*8


    //4. Configure M2 divider by setting CM_DIV_M2_DPLL_DISP.DPLL_CLKOUT_DIV to the
desired value.
    HWREG(0x44E00400 + 0xA4) = HWREG(0x44E00400 + 0xA4) & 0xfffffffe0 | 0x1;   //M2
DIV [4-0];                /1
    HWREG(0x44E00400 + 0xA4);

    //lcd_clk= 24M *82 *8 / 24/1= 82M*8

    //5. Switch over to lock mode by setting CM_CLKMODE_DPLL_DISP.DPLL_EN to 0x7.
    HWREG(0x44E00400 + 0x98) = HWREG(0x44E00400 + 0x98) & 0xfffffffc | 0x7;

    //6. Wait for CM_IDLEST_DPLL_DISP.ST_DPLL_CLK = 1 to ensure PLL is locked
    //(CM_IDLEST_DPLL_DISP.ST_MN_BYPASS should also change to 0 to denote the PLL is
out of bypass mode).
    while ((HWREG(0x44E00400 + 0x48) & 0x1) != 0x1);             //ST_DPLL_CLK
    while ((HWREG(0x44E00400 + 0x48) & (0x1 << 8)) != (0x0 << 8));
      //ST_MN_BYPASS
```