第010课 掌握ARM芯片时钟体系

来自百问网嵌入式Linux wiki

第001节_S3C2440时钟体系结构

S3C2440是System On Chip(SOC)，在芯片上不仅仅有CPU还有一堆外设。

至于有哪些外设，可以查看参考手册。在S3C2440参考手册的第一章PRODUCT OVERVIEW里面有个BLOCK DIAGRAM图：

BLOCK DIAGRAM

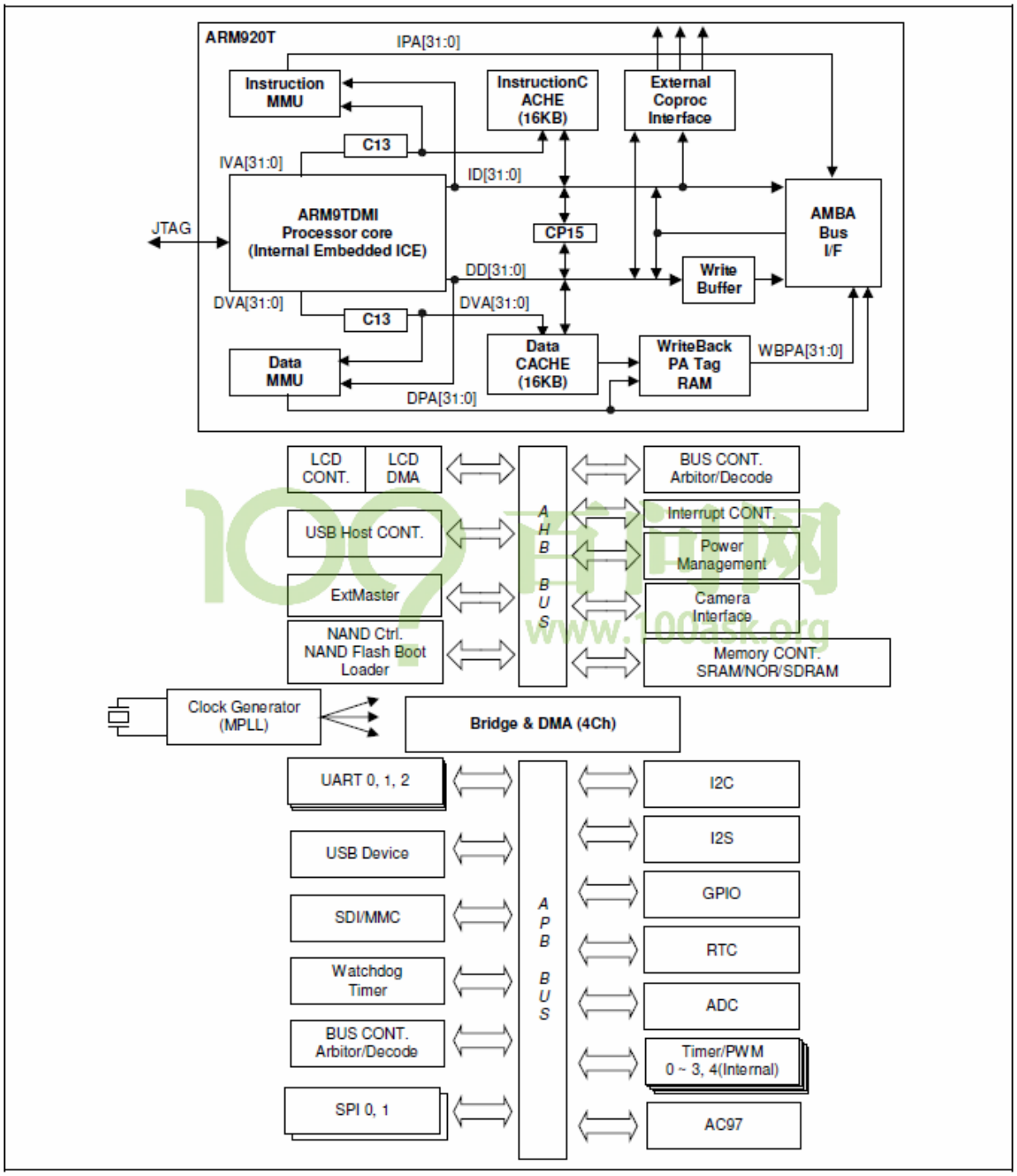


Figure 1-1. S3C2440A Block Diagram

可以把该图分为上中下三块，上面的是与CPU密切相关的，工作于FCLK；中间的一些对性能要求较高的设备，像LCD显示、相机等，在AHB BUS，H即为High，高速之意，工作于HCLK；下面的是一些对性能要求不那么高的低速设备，在APB BUS，P即为Peripheral之意，工作在PCLK。

在参考手册的特性里介绍了S3C2440的工作频率，Fclk最高400MHz，Hclk最高136MHz，Pclk最高68MHz。

如何得到以上的三种时钟?

硬件电路上有个12M的晶振，作为时钟源产生12MHz的频率，经过SOC的PLL(锁相环)倍频产生Fclk、Hclk、Pclk。

再具体看看第7章的时钟，在Clock Generator Block Diagram展示了时钟的产生。

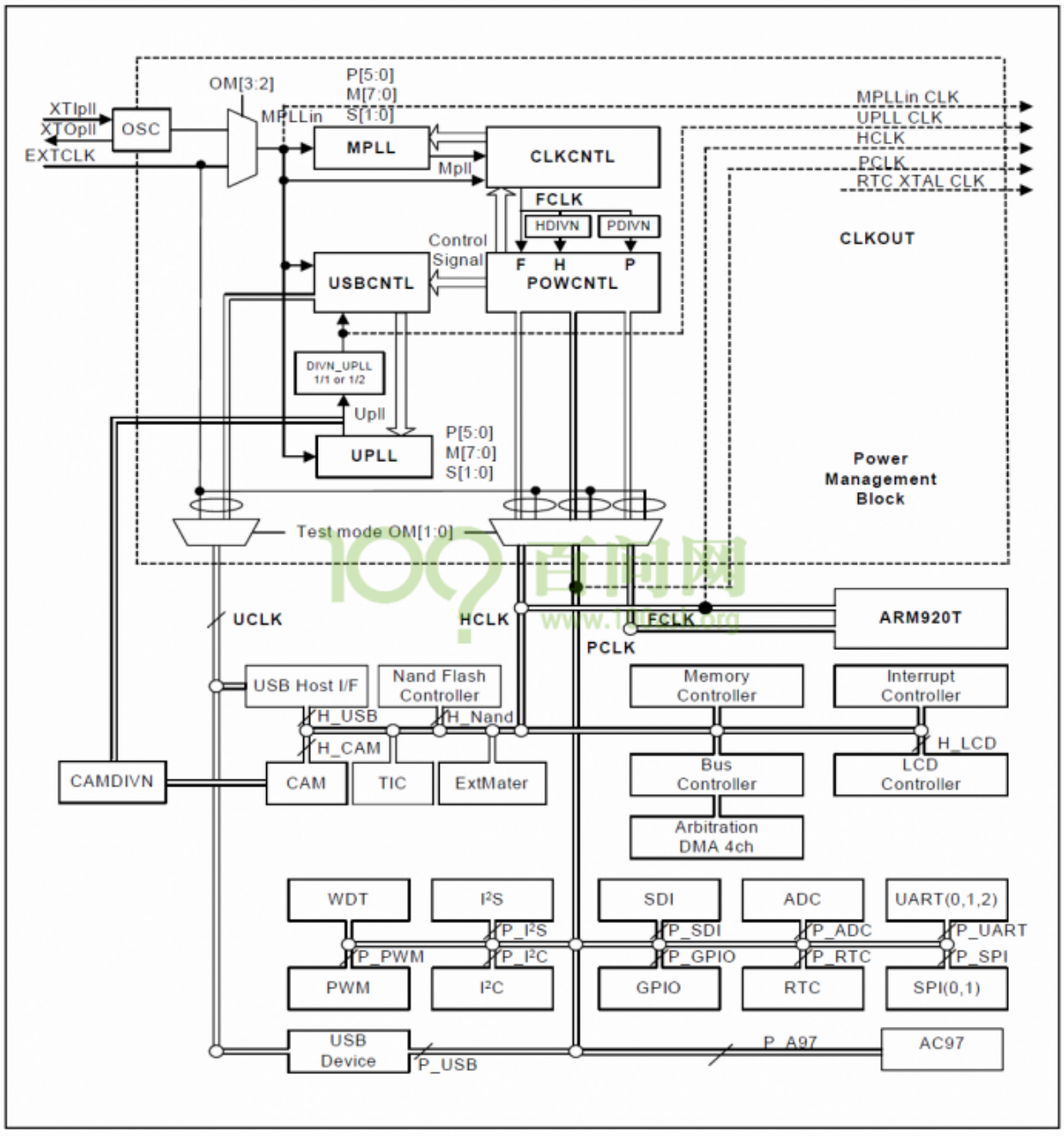
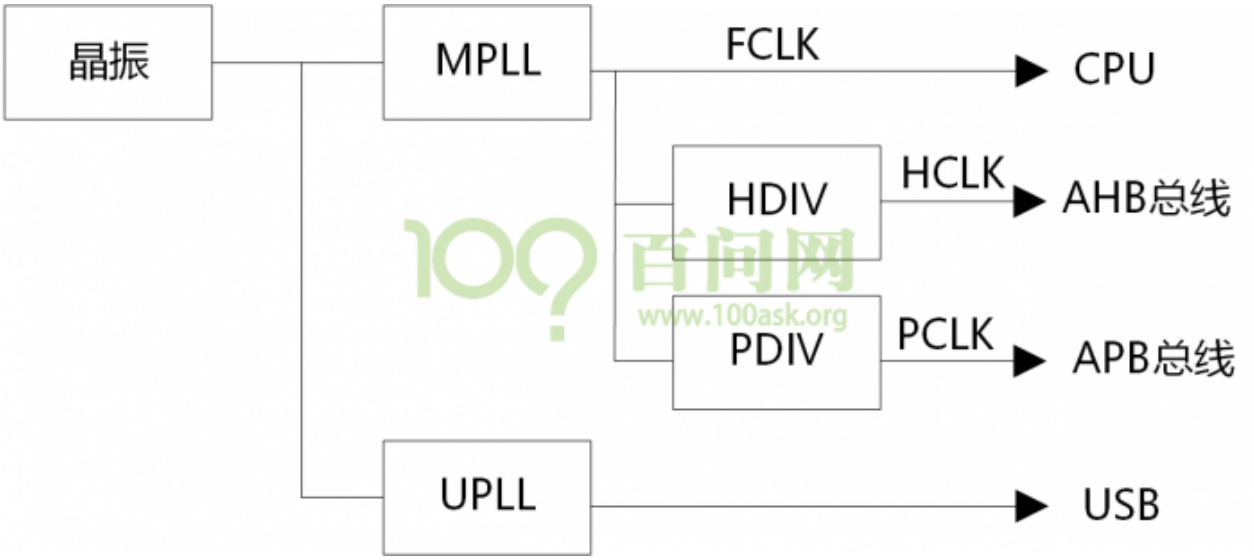


Figure 7-1. Clock Generator Block Diagram

在该图的左上角，晶振和一个外部时钟接在一个选择器上，这个选择器通过OM[3:2]的值来决定选择哪个时钟源。然后生成的MPLL(Main PLL)和UPLL(USB PLL)，MPLL直接提供给FCLK，通过HDIVN分频给HCLK，通过PDIVN分频给PCLK，再传给下面的各个设备。



第002节_编程提高运行时钟

怎么编程控制MPLL、HDIV、PDIV，使FCLK=400MHz，HCLK=100MHz，PLCK=50MHz?

需要设置MPLLCON的FCLK=400MHz，设置CLKDIVN的HCLK=FCLK/4，PCLK=FCLK/8。

- 1. 首先看CLKDIVN寄存器：

Register	Address	R/W	Description	Reset Value
CLKDIVN	0x4C000014	R/W	Clock divider control register	0x00000000

CLKDIVN	Bit	Description	Initial State
DIVN_UPLL	[3]	UCLK select register(UCLK must be 48MHz for USB) 0: UCLK = UPLL clock 1: UCLK = UPLL clock / 2 Set to 0, when UPLL clock is set as 48MHz Set to 1. when UPLL clock is set as 96MHz.	0
HDIVN	[2:1]	00 : HCLK = FCLK/1. 01 : HCLK = FCLK/2. 10 : HCLK = FCLK/4 when CAMDIVN[9] = 0. HCLK= FCLK/8 when CAMDIVN[9] = 1. 11 : HCLK = FCLK/3 when CAMDIVN[8] = 0. HCLK = FCLK/6 when CAMDIVN[8] = 1.	00
PDIVN	[0]	0: PCLK has the clock same as the HCLK/1. 1: PCLK has the clock same as the HCLK/2.	0

想设置HCLK=FCLK/4需要将HDIVN[2:1]设置为10，同时将CAMDIVN[9]设置为0。

查看CAMDIVN[9]的初始值默认就是0，因此只需要设置HDIVN[2:1]为10。

CAMDIVN	Bit	Description	Initial State
DVS_EN	[12]	0:DVS OFF ARM core will run normally with FCLK (MPLLout). 1:DVS ON ARM core will run at the same clock as system clock (HCLK).	0
Reserved	[11]	–	0
Reserved	[10]	–	0
HCLK4_HALF	[9]	HDIVN division rate change bit, when CLKDIVN[2:1]=10b. 0: HCLK = FCLK/4 1: HCLK= FCLK/8 Refer the CLKDIV register.	0

想设置PCLK=FCLK/8需要将PDIVN[0]设置为1，因此整个CLKDIVN寄存器设置如下：

```
/* CLKDIVN(0x4C000014) = 0X5, tFCLK:tHCLK:tPCLK = 1:4:8  */
  ldr r0, =0x4C000014
  ldr r1, =0x5
  str r1, [r0]
```

- 2. 现在看如何使FCLK=400MHz.

在手册的PLL VALUE SELECTION TABLE里列出了常见情况PLL的设置，我们输入的是晶振的12MHz，输出需要400MHz，因此根据表格需要设置 MDIV=92(0x5C)，PDIV=1，SDIV=1；

Input Frequency	Output Frequency	MDIV	PDIV	SDIV
12.0000MHz	48.00 MHz (Note)	56(0x38)	2	2
12.0000MHz	96.00 MHz (Note)	56(0x38)	2	1
12.0000MHz	271.50 MHz	173(0xad)	2	2
12.0000MHz	304.00 MHz	68(0x44)	1	1
12.0000MHz	400.00 MHz	92(0x5c)	1	1

在手册介绍了MPLL的m、p、s与MDIV、PDIV、SDIV之间的关系：

```
Mpll = (2 * m * Fin) / (p * 2^S)
m = (MDIV + 8), p = (PDIV + 2), s = SDIV

m=MDIV+8=92+8=100
p=PDIV+2=3
s=SDIV=1
MPLL=2x100x12/(3x2^1)=400MHz
```

PLL控制寄存器如下：

PLL CONTROL REGISTER (MPLLCON & UPLLCON)

Register	Address	R/W	Description	Reset Value
MPLLCON	0x4C000004	R/W	MPLL configuration register	0x00096030
UPLLCON	0x4C000008	R/W	UPLL configuration register	0x0004d030

PLLCON	Bit	Description	Initial State
MDIV	[19:12]	Main divider control	0x96 / 0x4d
PDIV	[9:4]	Pre-divider control	0x03 / 0x03
SDIV	[1:0]	Post divider control	0x0 / 0x0

因此需要配置（92<<12）|(1<<4)|(1<<0),

```
/* 设置MPLLCON(0x4C000004) = (92<<12)|(1<<4)|(1<<0)
   * m = MDIV+8 = 92+8=100
   * p = PDIV+2 = 1+2 = 3
   * s = SDIV = 1
   * FCLK = 2*m*Fin/(p*2^s) = 2*100*12/(3*2^1)=400M
   */
  ldr r0, =0x4C000004
  ldr r1, =(92<<12)|(1<<4)|(1<<0)
  str r1, [r0]
```

- 3.此外，手册还提到，需要SetAsyncBusMode。

```
MMU_SetAsyncBusMode
mrc p15,0,r0,c1,c0,0
orr r0,r0,#R1_nF:OR:R1_iA
mcr p15,0,r0,c1,c0,0
```

完整的start.S:

```
.text
.global _start

_start:

  /* 关闭看门狗 */
  ldr r0, =0x53000000
  ldr r1, =0
  str r1, [r0]

  /* 设置MPLL, FCLK : HCLK : PCLK = 400m : 100m : 50m */
  /* LOCKTIME(0x4C000000) = 0xFFFFFFFF */
  ldr r0, =0x4C000000
  ldr r1, =0xFFFFFFFF
  str r1, [r0]

  /* CLKDIVN(0x4C000014) = 0X5, tFCLK:tHCLK:tPCLK = 1:4:8  */
  ldr r0, =0x4C000014
  ldr r1, =0x5
  str r1, [r0]

  /* 设置CPU工作于异步模式 */
  mrc p15,0,r0,c1,c0,0
  orr r0,r0,#0xc0000000   //R1_nF:OR:R1_iA
  mcr p15,0,r0,c1,c0,0
```

```
    /* 设置MPLLCON(0x4C000004) = (92<<12)|(1<<4)|(1<<0)
     *  m = MDIV+8 = 92+8=100
     *  p = PDIV+2 = 1+2 = 3
     *  s = SDIV = 1
     *  FCLK = 2*m*Fin/(p*2^s) = 2*100*12/(3*2^1)=400M
     */
    ldr r0, =0x4C000004
    ldr r1, =(92<<12)|(1<<4)|(1<<0)
    str r1, [r0]

    /* 一旦设置PLL, 就会锁定lock time直到PLL输出稳定
     * 然后CPU工作于新的频率FCLK
     */



    /* 设置内存: sp 栈 */
    /* 分辨是nor/nand启动
     * 写0到0地址, 再读出来
     * 如果得到0, 表示0地址上的内容被修改了, 它对应ram, 这就是nand启动
     * 否则就是nor启动
     */
    mov r1, #0
    ldr r0, [r1] /* 读出原来的值备份 */
    str r1, [r1] /* 0->[0] */
    ldr r2, [r1] /* r2=[0] */
    cmp r1, r2   /* r1==r2? 如果相等表示是NAND启动 */
    ldr sp, =0x40000000+4096 /* 先假设是nor启动 */
    moveq sp, #4096  /* nand启动 */
    streq r0, [r1]   /* 恢复原来的值 */


    bl main

halt:
    b halt
```

《《所有章节目录》》

分类： ARM裸机加强版

- 本页面最后修改于2018年1月25日 (星期四) 07:38。