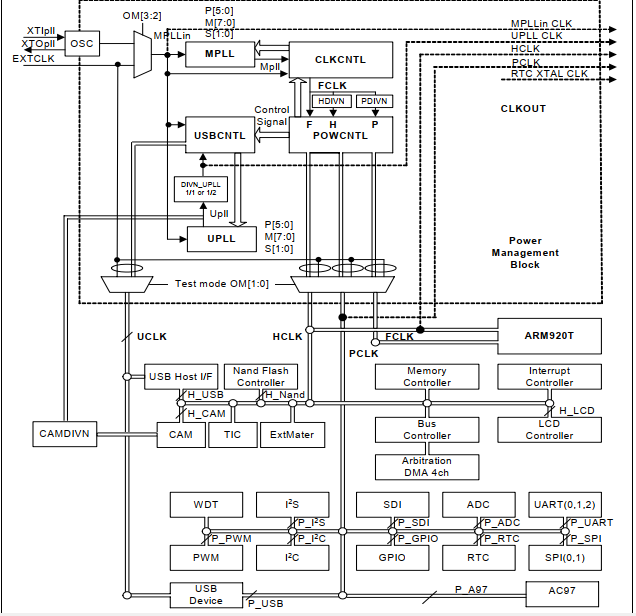
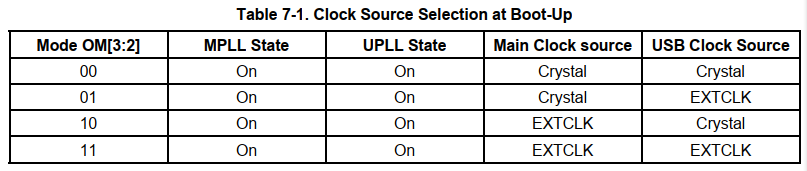
1. S3C2440时钟体系
2. BLOCK DIAGRAM



一种时钟源：12M外部晶振/外部引脚 由OM[3:2]决定



当这两个晶振引脚都接地时，时钟源为外部12M晶振（Crystal）

三种时钟：FCLK HCLK PCLK

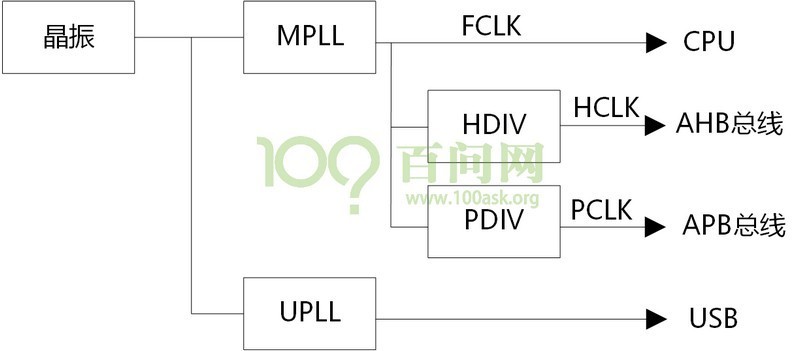
FCLK：400MHz 用于CPU核

HCLK：136MHz 用于AHB高速设备总线,比如CPU核、存储器控制器、中断控制器、DMA、USB主机模块

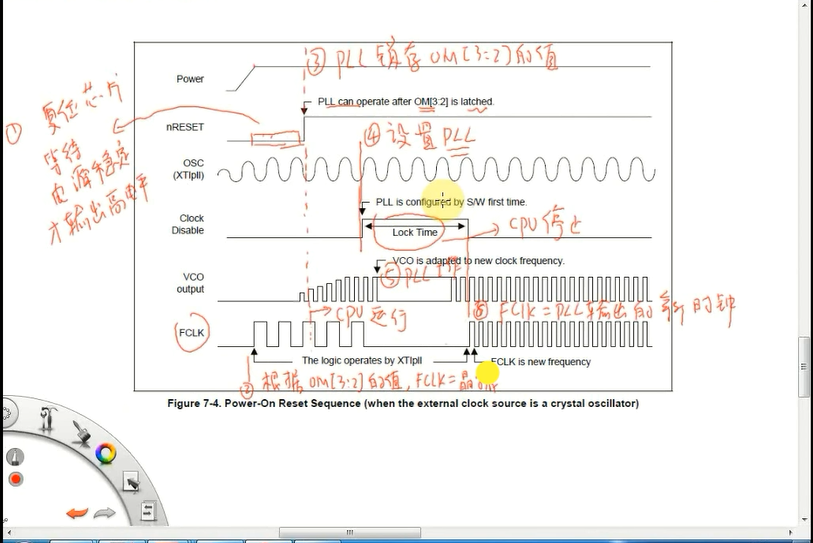
PCLK：68MHz 用于APB低带宽外部设备总线,比如USART,IIC,IIS,GPIO,RTC,TIM等

1. PLL（锁相环）

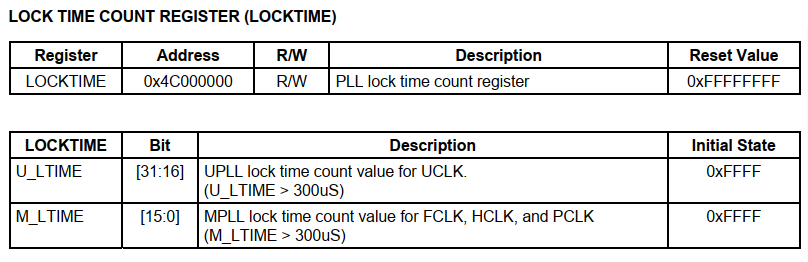
12M晶振通过MPLL得到FCLK，用于CPU核，通过UPLL得到专用于USB的时钟。FCLK再经过HDIV和PDIV产生HCLK和PCLK，HCLK和PCLK再作用于AHB和APB



工作过程如下：



1. 复位芯片，等待电源稳定才输出高电平
2. 一上电，FCLK便开始起振，根据OM[3:2]的值，FCLK=晶振
3. 复位芯片输出高电平，CPU开始工作，PLL锁存OM[3:2]的值
4. 软件配置PLL相关寄存器，改变FCLK的频率，此时CPU停止工作，FCLK停振，其锁定时间由下图所示寄存器决定，默认使用默认值

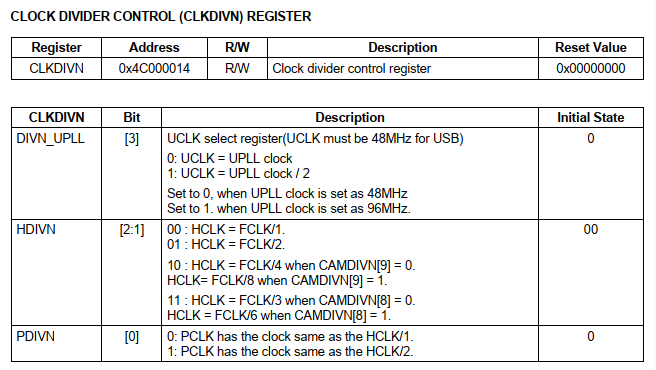


1. PLL开始工作
2. 设置完成后，FCLK=PLL输出的新时钟，CPU继续运行

主要设置MPLLCON CLKDIVN来控制FCLK、HCLK和PCLK，设置CLKCON控制外设设备时钟的开关，例如Camera、AC97、SPI、IIS、IIC等，默认值为0xfffff0，开启全部外设时钟，当设计项目时，应当只开启必要外设时钟，减小CPU能耗

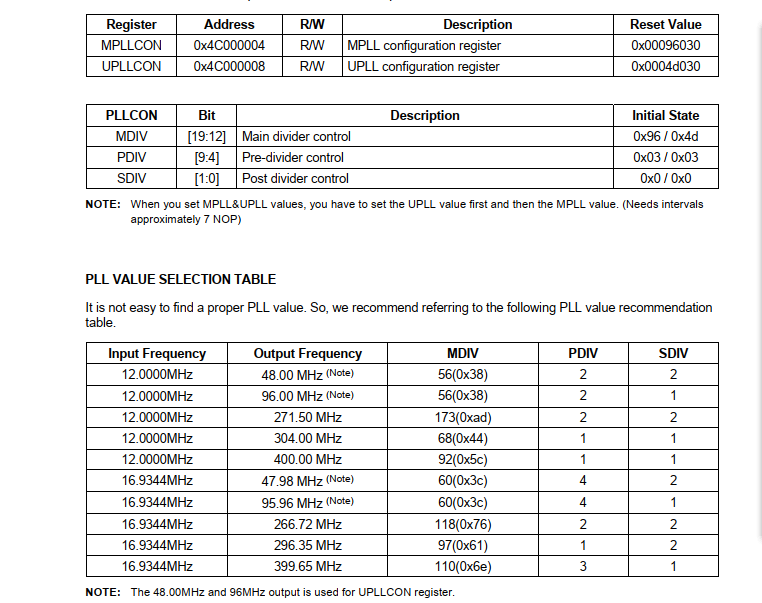
1. 以FCLK=400MHz HCLK=100MHz PCLK=80MHz为例

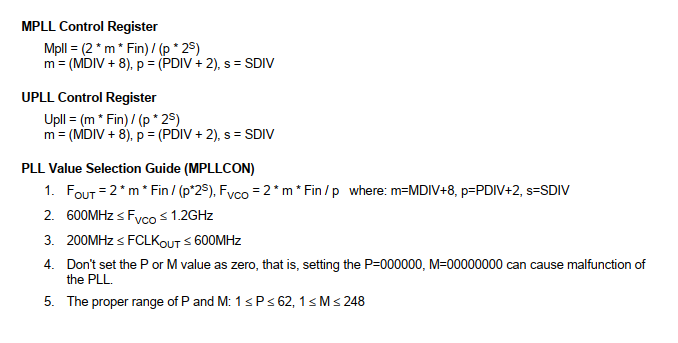
（1） CLKDIVN=0x05 (DIVN\_UPLL=0 HDIVN=10 PDIVN=1 🡪 0101)



（2） MPLLCON=(92<<12)|(1<<4)|(1<<0) FCLK=(2\*m\*Fin)/(p\*2^S)

m=MDIV+8 Fin=12 p=PDIV S=SDIV





（3） CPU开启异步模式（当HDIV!=0时，如不开启异步模式,则CPU时钟为HCLK）

**NOTES**1. CLKDIVN should be set carefully not to exceed the limit of HCLK and PCLK.  
2. If HDIVN is not 0, the CPU bus mode has to be changed from the fast bus mode to the asynchronous  
bus mode using following instructions(S3C2440 does not support synchronous bus mode).  
**MMU\_SetAsyncBusMode  
mrc p15,0,r0,c1,c0,0 //c:co-process 协处理器 r:register寄存器  
orr r0,r0,#R1\_nF:OR:R1\_iA //orr r0,r0,0xc0000000  
mcr p15,0,r0,c1,c0,0**If HDIVN is not 0 and the CPU bus mode is the fast bus mode, the CPU will operate by the HCLK.

1. UART 参考博客：<https://blog.csdn.net/mr_raptor/article/details/6556133>

A．Printf的编写

1. 结构体对齐

可以通过使用gcc中的\_attribute\_选项来设置指定的对齐大小

1）：

\_\_attribute\_\_((packed))，让所作用的结构体取消在编译过程中的优化对齐，按照实际占用字节数进行对齐

2）：

\_\_attribute((aligned (n)))，让所作用的结构体成员对齐在n字节边界上，如果结构体中有成员变量的字节长度大于n，则按照最大成员变量的字节长度来进行对齐

1. printf (const char\* format, ...)

指定参数format存放在一个堆栈中，可变参数存放在另一个堆栈中。可通过获得指向format的指针p，改变p的指向，可以获得可变参数的首地址，进而打印出来，对于printf函数，所有的浮点数默认为double型，不是float型

1. printf中使用到的四个函数（stdarg.h）
2. va\_list p; //定义一个变参变量p,等价于char \*p
3. va\_start(p,format); //format是固定参数，经过此函数后，移动指针p到第一个变参变量
4. var = va\_arg(p,变量类型); //在已知变量类型的情况下，返回当前p所指向的变参变量的值给var，然后移动指针p到下一个变参变量
5. va\_end(p) ; //结束变参变量的使用，等价于p=NULL，避免野指针。
6. 整体结构

int printf(const char \*fmt , ...)

{

va\_list ap; //先创建一个指向堆栈的指针

va\_start(ap,fmt); //移动到第一个固定参数变量所在位置

my\_vprintf(fmt,ap); //遍历堆栈变量 并且打印出来

va\_end(ap); //清空指针ap 避免野指针

return 0; //打印结束 返回0

}

B．汇编中的除法和取余运算

当在C语言程序中用到除法和取余运算时，直接放到汇编中编译会出现以下类似错误：

undefined reference to `\_\_umodsi3'

undefined reference to `\_\_udivdi3'

以上是因为ARM是精简指令集，对求余和除法操作基本上不支持，所以应该尽量避免上述操作。libgcc库包含这些操作，这种错误是没有包含GCC的支持库libgcc.a，缺少库文件。

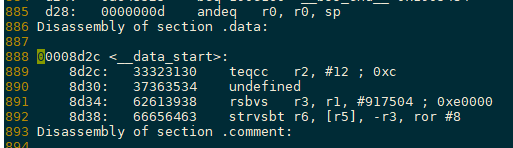
所以在Makefile中，应该也要编译库文件🡪 lib1funcs.S，但是编译时注意连接顺序

arm-linux-ld -Ttext 0 -Tdata 0xd30 start.o lib1funcs.o uart.o my\_printf.o main.o -o uart.elf

lib1funcs应在start.o之后，否则烧写程序后，程序无法运行

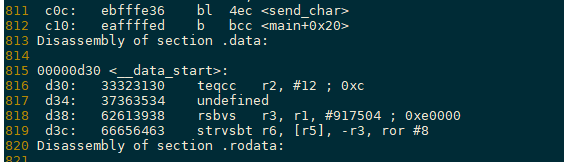
-Tdata 0xd30是根据默认的数据段起始地址来修改的，之前默认地址是8d24，这样造成的结果是.bin接近于4K,占用空间过大，应将其改为d20或者比d20要大

修改前：





修改后：





大小缩小了十倍左右

注意：串口打印字符串乱码的问题还未解决