 7．2．3 内核对RTC的操作 如前所述，Linux内核与RTC进行互操作的时机只有两个：（1）内核在启动时从RTC中读取启动时的时间与日期；（2）内核在需要时将时间与日期回写到RTC中。为此，Linux内核在arch/i386/kernel/time.c文件中实现了函数get_cmos_time()来进行对RTC的第一种操作。显然，get_cmos_time()函数仅仅在内核启动时被调用一次。而对于第二种操作，Linux则同样在arch/i386/kernel/time.c文件中实现了函数set_rtc_mmss()，以支持向RTC中回写当前时间与日期。下面我们将来分析这二个函数的实现。 在分析get_cmos_time()函数之前，我们先来看看RTC芯片对其时间与日期寄存器组的更新原理。（1）Update In Progress 当控制寄存器B中的SET标志位为0时，MC146818芯片每秒都会在芯片内部执行一个“ 更新周期 ”（Update Cycle），其作用是增加秒寄存器的值，并检查秒寄存器是否溢出。如果溢出，则增加分钟寄存器的值，如此一致下去直到年寄存器。在“ 更新周期 ”期间，时间与日期寄存器组（0x00~0x09）是不可用的，此时如果读取它们的值将得到未定义的值，因为MC146818在整个更新周期期间会把时间与日期寄存器组从CPU总线上脱离，从而防止软件程序读到一个渐变的数据。 在MC146818的输入时钟频率（也即晶体增荡器的频率）为4.194304MHZ或1.048576MHZ的情况下，“ 更新周期 ”需要花费248us，而对于输入时钟频率为32.768KHZ的情况，“ 更新周期 ”需要花费1984us＝1.984ms

。控制寄存器A中的UIP标志位用来表示MC146818是否正处于更新周期中，当UIP从0变为1的那个时刻，就表示MC146818将在稍后马上就开更新周期。在UIP从0变到1的那个时刻与MC146818真正开始Update Cycle的那个时刻之间时有一段时间间隔的，通常是244us。也就是说，在UIP从0变到1的244us之后，时间与日期寄存器组中的值才会真正开始改变，而在这之间的244us间隔内，它们的值并不会真正改变。如下图所示：（2）get_cmos_time()函数 该函数只被内核的初始化例程time_init()和内核的APM模块所调用。其源码如下：/* not static: needed by APM */ unsigned long get_cmos_time(void) { unsigned int year, mon, day, hour, min, sec. int i. /* The Linux interpretation of the CMOS clock register contents: * When the Update-In-Progress (UIP) flag goes from 1 to 0, the * RTC registers show the second which has precisely just started. * Lets hope other operating systems interpret the RTC the same way. */ /* read RTC exactly on falling edge of 0update flag */ for (i = 0 . i 对该函数的注释如下：（1）在从RTC中读取时间时，由于RTC存在Update Cycle，因此软件发出读操作的时机是很重要的。对此，get_cmos_time()函数通过UIP标志位来解决这个问题：第一个for循环不停地读取RTC频率选择寄存器中的UIP标志位，并且只要读到UIP的值为1就马上退出这个for循环。第二个for循环同样不停地读取UIP标志位，但他只要一读到UIP的值为0就马上退出这个for循环。这两个for循环的目的就是要在软件逻辑上同步RTC的Update Cycle，显然第二个for循环最大可能需要2.228ms(TBUC max(TUC)=244us 1984us=2.228ms) 100Test 下载频道开通，各类考试题目直接下