**主函数与中断函数共享变量问题**

当时开发的一个产品，一项功能是在通电后播放40秒的语音.  
测试时发现，大约通电70-80次就有一次播放时间不够40秒就提前停止。  
当时以为复位有问题，换了复位片，没好。又先后换了CPU，语音芯片，还有电源，都没有好转。排除了硬件芯片原因导致的此现象.  
后来又从软件中查找原因。反复查找软件逻辑，也没发现问题。后来偶然发现在主while里增加大量延时后，稳定性提高。  
几乎不再出现问题。但是我还是觉得不对劲,用了两天时间终于找到了原因。因为这是公司的程序，所以不能贴源码。  
我把其他程序都略去，只把出错的程序大概写一下。大家看看能找到问题吗？  
unsigned int ms\_counter;                        
void T0()  
{         
    //定时器程序每100毫秒中断一次，程序略       
    if (ms\_counter<1000) ms\_counter++;  
}  
void main(void)  
{  
//初始化定时器程序每100毫秒中断一次，程序略                             
unsigned char tt;  
    ms\_counter=0;  
    tt=0;//用tt控制只响一次  
    while(1)  
    {  
        if (ms\_counter<400)  
        {  
            if (tt==0)  
            {  
            tt=1;  
            Sound\_on();  
            }  
        }  
        else  
        {  
            Sound\_off();  
        }     
    //其他程序  
    //。。。。。。      
    }  
}   
  
  
高手们不要笑，菜鸟们坐好  
问题出在ms\_counter不到400时，程序提前执行了Sound\_off();  
原因分析：if (ms\_counter<400)中的ms\_counter是两字节的整型，而且在中断里有增一操作。  
这就有一种错误的可能  
if (ms\_counter<400)  //被编译器翻译成以下语句  
+0000007C:   E9E0        LDI     R30,0x90         Load immediate  
+0000007D:   E0F1        LDI     R31,0x01         Load immediate  
+0000007E:   164E        CP      R4,R30           Compare  
+0000007F:   065F        CPC     R5,R31           Compare with carry  
+00000080:   F428        BRCC    +0x05            Branch if carry cleared  
在ms\_counter==255时  R4是255  R5是0  
  
CP      R4,R30  ;这时R4是255     
注意！如果在这两条语句中间产生了中断  ms\_counter增一 以后  R4是0    R5是1  
CPC     R5,R31  ;这时R5是1  
  
简单的说是由于在整型数增一进位的时候，又受到中断的影响。  
本来正确值  0x00ff或0x0100（ms\_counter），  
实际错误值  0x01ff（ms\_counter） 先判断低位时低位是FF，中断后判断高位时高位是01  
ms\_counter在255时被误认为511（0x01ff）导致提示音提前关闭。

当主函数与中断函数共用变量时，可能发生：   
1.主函数对变量的 读-写，可能造成中断函数对变量的 读-写无效。   
如：当主函数刚刚把变量读入到内部寄存器时，还未再回写到变量中 时，发生中断，中断中改写了变量。当中断返回时，主函数将值再回写到变量中。造成中断函数对变量的改写无效。   
2.多字节变量读取错误。   
如：当变量的其中一个字节读入到寄存器中时，发生中断，中断中改写了变量值。当中断返回时，变量的其他字节继续被读入到寄存器中，造成新旧字节组合错误。

主函数与中断函数共享变量问题类似两个线程共享资源的问题，如何解决共享资源冲突是系统结构设计的关键

解决方法  
1.volatile正确使用.   
2.注意临界段(或原子操作). （写变量时,关中断,写完后再开）

操作系统中对这种问题有另一种解决办法，即引入一个与ms\_counter相同类型的临时变量：  
unsigned int tmp\_counter;  
  
在使用ms\_counter做判断前作如下操作：  
do {  
    tmp\_counter = ms\_counter;  
while (tmp\_counter != ms\_counter);  
  
然后使用tmp\_counter代替ms\_counter进行判断，这样可以保证回避楼主所述问题。

一般说来，volatile用在如下的几个地方：

1)、中断服务程序中修改的供其它程序检测的变量需要加volatile；

2)、多任务环境下各任务间共享的标志应该加volatile；

3)、存储器映射的硬件寄存器通常也要加volatile说明，因为每次对它的读写都可能由不同意义；