Volatile是区分C程序员和嵌入式系统程序员最基本的问题。 嵌入式系统程序员经常同硬件、中断、RTOS等打交道，所有这些都要求volatile变量。不懂volatile将会带来灾难。  
  
volatile的作用

volatile的作用: 作为指令关键字,确保本条指令不会因编译器的优化而省略，要求每次直接读值.非常典型的就是寄存器的值可能被人为的从硬件上改变，所以必须要求每次直接读值。  
简单地说就是防止编译器对代码进行优化.比如程序:

XBYTE[2]=0x55;  
XBYTE[2]=0x56;  
XBYTE[2]=0x57;  
XBYTE[2]=0x58;

如果对外部硬件的上述四条语句分别表示不同的操作, 会产生四种不同的动作,那么编译器就不能像对待纯粹的程序那样对上述四条语句进行优化只认为XBYTE[2]=0x58，而忽略前三条语句，并且只产生1条机器码。如果XBYTE用volatile修饰过，此时编译器会逐一的进行编译并产生相应的4条机器码。  
再详细解释一下：  
　　定义为volatile的变量是说这变量可能会被意想不到地改变，这样，编译器就不会去假设这个变量的值了。精确地说就是，优化器在用到这个变量时必须每次都小心地重新读取这个变量的值，而不是使用保存在寄存器里的备份。

下面是volatile变量的几个例子：  
　　1). 并行设备的硬件寄存器（如：状态寄存器）  
　　2). 一个中断服务子程序中会访问到的非自动变量(Non-automatic variables)  
　　3). 多线程应用中被几个任务共享的变量  
  
面试中可能会提问到volatile的相关知识：  
　　1). 一个参数既可以是const还可以是volatile吗？为什么。  
　　2). 一个指针可以是volatile 吗？为什么。  
　　3). 下面的函数有什么错误：  
　　int square(volatile int \*ptr)  
　　{  
　　return \*ptr \* \*ptr;  
　　}  
　　下面是答案：  
　　1). 是的。一个例子是只读的状态寄存器。它是volatile因为它可能被意想不到地改变。它是const因为程序不应该试图去修改它。  
　　2). 是的。尽管这并不很常见。一个例子是当一个中断服务子程序修改一个指向一个buffer的指针时。  
　　3). 这段代码是个恶作剧。这段代码的目的是用来返回指针\*ptr指向值的平方，但是，由于\*ptr指向一个volatile型参数，编译器将产生类似下面的代码：  
　　int square(volatile int \*ptr)  
　　{  
　　int a,b;  
　　a = \*ptr;  
　　b = \*ptr;  
　　return a \* b;  
　　}  
　　由于\*ptr的值可能被意想不到地改变，因此a和b可能是不同的。结果，这段代码可能返回不是你所期望的平方值！正确的代码如下：  
　　long square(volatile int \*ptr)  
　　{  
　　int a;  
　　a = \*ptr;  
　　return a \* a;  
　　}