第一百二十三节： 一种能省去一个 lock 自锁变量的按键驱动程序。

【123.1 一种能省去一个 lock 自锁变量的按键驱动程序。】

一位群友给我提到了一个按键的改进建议，能巧妙的省去一个 lock 自锁变量。这个建议引起了我对“变量的分工要专一，一个变量尽量只用在一类事物上，尽量不取巧兼容”的思考。

第一种：带 lock 自锁变量，也是我一直在用的代码。

```
if(0!=KEY_INPUT1)//IO 是高电平，说明按键没有被按下，这时要及时清零一些标志位
{
    Su8KeyLock1=0; //按键解锁
    Su16KeyCnt1=0; //按键去抖动延时计数器清零，此行非常巧妙，是全场的亮点。
}
else if(0==Su8KeyLock1)//有按键按下，且是第一次被按下。这行如果有疑问，请看 92 节的专题分析。
{
    Su16KeyCnt1++; //累加定时中断次数
    if(Su16KeyCnt1>=KEY_FILTER_TIME) //滤波的“稳定时间”KEY_FILTER_TIME，长度是 25ms。
    {
         Su8KeyLock1=1;  //按键的自锁,避免一直触发
         vGu8KeySec=1;    //触发 1 号键
    }
}
```

、

第二种：省略掉一个 lock 自锁变量，群友提出的改进建议。

```
if(0!=KEY_INPUT1)
{
    Su16KeyCnt1=0;
}
else if(Su16KeyCnt1<KEY_FILTER_TIME) //巧妙的利用了 Su16KeyCnt1 等于滤波时间时，只执行一次
{
    Su16KeyCnt1++;
    if(KEY_FILTER_TIME==Su16KeyCnt1) //巧妙的利用了 Su16KeyCnt1 等于滤波时间时，只执行一次
    {
         vGu8KeySec=1;
    }
}
```

分析：

不得不佩服群友的智慧，第二种改进后看起来非常巧妙，犹如蜻蜓点水般轻盈洒脱。但是，为此代码狂欢片刻后，我又有了新的思考和看法。“计时器 Su16KeyCnt1”和“自锁变量 Su8KeyLock1”是两个不同的事物，是两个不同的范畴，就应该用两个不同的变量进行区分。如果逞一时之巧，把两种不同范畴的事物巧妙合并成一个变量,势必会导致程序的“易读性”和“后续维护的可扩展性”大打折扣。“自锁变量 Su8KeyLock1”真的是可有可无吗？假设，如果“计时器 Su16KeyCnt1”的消抖时间 KEY_FILTER_TIME 要求等于 0，那么第二种改进后的代码立刻暴露出了问题，行不通。而第一种代码，因为有“自锁变量 Su8KeyLock1”的存在，即使消抖时间 KEY_FILTER_TIME 等于 0，也不影响代码功能的完整性，因为第一种代码的理念是“自锁与计

时器是两种不同的功能范畴，用两个不同的变量进行分开隔离，各自管理两种不同的事物，计时器即使为 0 也不影响代码本该有的自锁功能”。通过此例子，给初学者一个建议，在代码的“队形感，易读性，扩展性，分类清晰”和“巧妙，节省代码”两者之间，建议大家优先考虑“队形感，易读性，扩展性，分类清晰”，追求一种原则上的“工整，不出奇兵，扎硬寨，打呆仗，步步为营”，这样阵脚不易乱，能走得更远，驾驭更多千军万马的代码。