

第八十三节： 累计主循环的“非阻塞”延时控制 LED 闪烁。

【83.1 累计主循环的“非阻塞”。】

上一节提到，当 Delay 的“阻塞”时间超过 1ms 并且被频繁调用的时候，由于 Delay 做“独占式无用功”而消耗的延时太长，会影响其它任务的并行处理，整个系统给人的感觉非常卡顿不流畅。为了解决此问题，本节引入累计主循环的“非阻塞”，同时，希望通过此例子，让大家第一次感受到 switch 语句在多任务并行处理时候的优越性。switch 的精髓在于“根据某个判断条件实现步骤之间的灵活跳转”，这个思路是以后做所有大项目的框架性思路。

为什么“累计主循环”可以兼顾到其它任务的并行处理？因为单片机进入 main 函数以后，在一个主循环里要扫描 N 个任务，从头到尾，把 N 个任务扫描一遍，每扫描一遍算“一次主循环”，每一次“主循环”都是要消耗一点时间，累计的“主循环”次数越多，所要消耗的时间就越长，但是跟 Delay 唯一的差别是，Delay 做延时的时候没有办法扫描其它任务，而“累计主循环”内部本身就是在不断扫描其它任务，产生时间越长扫描其它任务的次数就越多，两者是完全相互促进而没有矛盾的。具体内容，请看下面的例子。

【83.2 累计主循环“非阻塞”的一个例子。】

现在利用“累计主循环非阻塞”编写一个练习程序，让一个 LED 灯闪烁。例子如下：

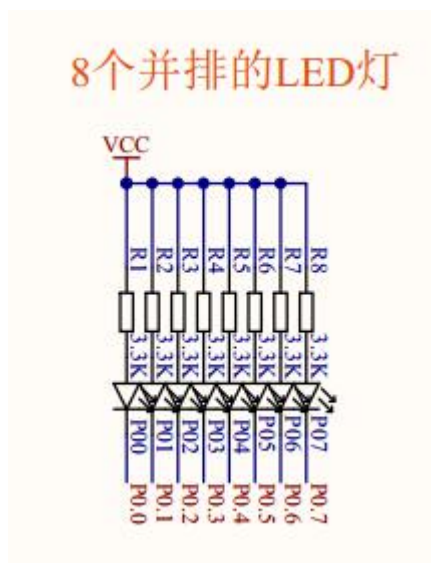


图 83.2.1 灌入式驱动 8 个 LED

```
#include "REG52.H"

#define CYCLE_SUM 5000 //累计主循环次数的设定阈值，该值决定了 LED 闪烁频率

sbit P0_0=P0^0; //利用 sbit 和符号“^”的组合，把变量名字 P0_0 与 P0.0 引脚关联起来

unsigned char Gu8CycleStep=0; //switch 的跳转步骤
unsigned long Gu32CycleCnt=0; //累计主循环的计数器
```

```

void main()
{
    while(1)
    {
        switch(Gu8CycleStep)
        {
            case 0:
                Gu32CycleCnt++;    //这里就是累计 main 函数内部的主循环 while(1)的次数
                if(Gu32CycleCnt>=CYCLE_SUM) //累计的次数达到设定值 CYCLE_SUM 就跳到下一步骤
                {
                    Gu32CycleCnt=0;    //及时清零计数器，为下一步骤的新一轮计数准备
                    P0_0=0; //LED 灯亮。
                    Gu8CycleStep=1; //跳到下一步骤
                }
                break;

            case 1:
                Gu32CycleCnt++;    //这里就是累计 main 函数内部的主循环 while(1)的次数
                if(Gu32CycleCnt>=CYCLE_SUM) //累计的次数达到设定值 CYCLE_SUM 就返回上一步骤
                {
                    Gu32CycleCnt=0;    //及时清零计数器，为返回上一步骤的新一轮计数准备
                    P0_0=1; //LED 灯灭。
                    Gu8CycleStep=0; //返回到上一个步骤
                }
                break;
        }
    }
}

```

【83.3 累计主循环的不足。】

上述 83.2 例子中，“累计主循环次数”实现时间延时是一个不错的选择。这种方法能胜任多任务处理的程序框架，但是本身也有一个小小的不足，比如“阈值 CYCLE_SUM 到底应该取多少才能产生多长的时间”是没有标准的，只能依靠不断上机实验来拿到一个你所需要的数值，这种“不规范”，当程序要移植到其它单片机平台上的时候就特别麻烦，需要重新修正阈值 CYCLE_SUM。除此之外，哪怕在同样的一个单片机里，随着主函数里任务量的增加，累计一次主循环所消耗的时间长度也会发生变化，意味着靠“累计主循环次数”所获得的时间也会发生变化而导致不准确，此时，为了保证延时时间的准确性，必须要做的就是再一次修正“设定累计主循环次数”的阈值 CYCLE_SUM，这样显然给我们带来了一丝不便，怎么办？假设单片机没有“定时中断”这个资源，那么这种“累计主循环次数”在多任务处理中确实是不二之选，但是，因为现在几乎所有的单片机内部都有“定时中断”这个资源，所以，大家不用为这个“不足”而烦恼，我们只要用上本节的 switch 思路，再外加一个“定时中断”，就可以轻松解决此问题，下一节就跟大家讲“定时中断”的内容。