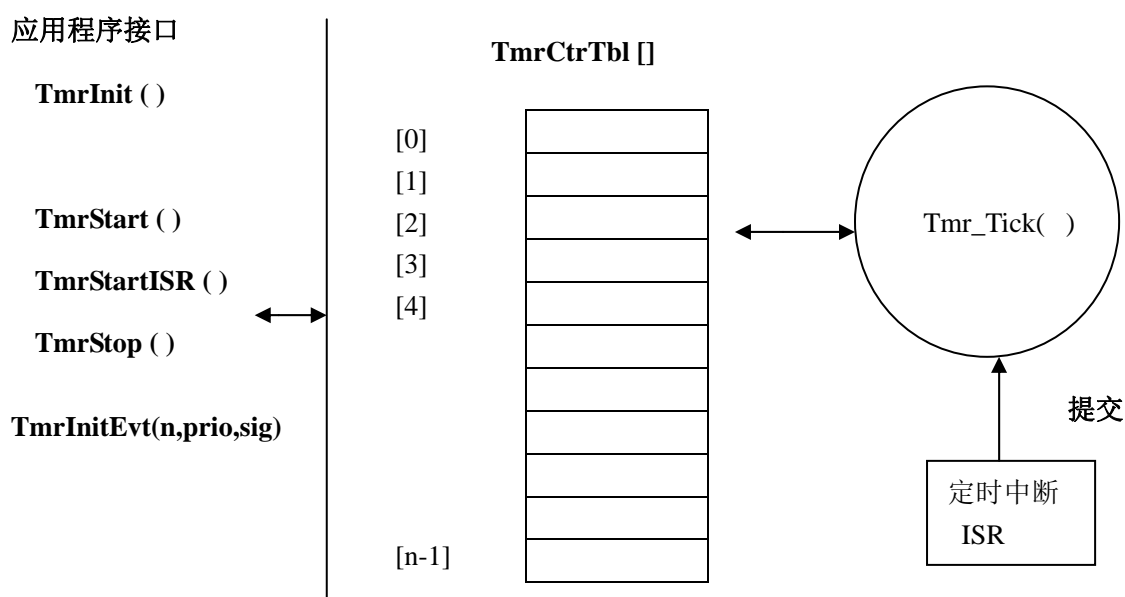


节约 RAM 的倒计时器处理方法

一、 计时器概述

程序设计时经常会用到计时，比如：30MS 做一次按键扫描，100MS 做一次 AD 转换，5 秒超时处理等等，但 MCU 的硬件定时器是有限数量的，本文介绍一种用一个定时器，最少的内存占用，做出多到 **255 个倒计时器** 出来。

二、 计时原理分析：



每一个时钟节拍（通过定时器设定一个时间间隔，比如 1MS），调用 Tmr_Tick()，扫描一次 TmrCtrTbl []，如果为 0 则为已停止的计时器，如不为 0 则减 1 操作，如减后为 0 则通知定时时间到，同时这个计时器也停止了（因为为 0 了）。这样每一个计时器只占数组的一个单元。

如果 TmrCtrTbl [] 是 8 位的数组（和定时长短有关），则一个计时器只占用一个字节。

```

void Tmr_Tick (void)
{
    INT8U i,prio,sig;
    TMRSIG *ptmr;
    for (i = 0; i < TMR_MAX_TMR; i++) {
        if (TmrCtrTbl[i] != 0) {
            if((-- TmrCtrTbl[i]) == 0){ //如不为 0 则减 1,减为 0 则计时结束
                ptmr = &TmrEvtList[i]; //
                prio = ptmr->prio;
                sig = ptmr->sig;
                EvtPostQue(prio,sig,0);//给设定的任务发设定的消息
            }
        }
    }
}
    
```

三、应用程序接口：

1、TmrInit ()：用初始化

```

void TmrInit (void)
{
    INT8U i;
    for (i = 0; i < TMR_MAX_TMR; i++) {
        TmrCtrTbl[i] = 0;
    }
}
    
```

2、TmrStart ()：用于启动计时器

启动一个计时器比较简单，只需把计时时间赋值给对应的计时器号就可以了。

```

void TmrStart (INT8U n, TMR tenths)
{
    if (n < TMR_MAX_TMR) {
        OS_ENTER_CRITICAL();
        TmrCtrTbl[n] = tenths;
        OS_EXIT_CRITICAL();
    }
}

void TmrStartISR (INT8U n, TMR tenths)
{
    if (n < TMR_MAX_TMR) {
        TmrCtrTbl[n] = tenths;
    }
}
    
```

```
}
```

3、TmrStop() : 用于停止计时器

停止一个计时器，就把对应值清零就可以了。

```
void TmrStop (INT8U n)
{
    if (n < TMR_MAX_TMR) {
        OS_ENTER_CRITICAL();
        TmrCtrTbl[n] = 0;
        OS_EXIT_CRITICAL();
    }
}
```

4、TmrInitEvt(n,prio,sig) : 设定计时为哪个任务发消息，发什么消息

```
void TmrInitEvt(INT8U n,INT8U prio,INT8U sig)
{
    TMRSIG *ptmr;
    if (n < TMR_MAX_TMR) {
        ptmr = &TmrEvtList[n];
        ptmr->prio = prio;
        ptmr->sig = sig;
    }
}
```

四、结束语：

如果时钟节拍为 1MS,用 16 位数组,则可以计时从 2MS 到 65 秒

如果定义 5 个倒计时器，则指定用 10 字节 RAM。本文中发消息部分读者可以改成自己的方式。