

最小的多线程框架

华荣汇电子科技（北京）有限公司

摘要：本文介绍一种方法，在 8 位 MCU 上进行任务切换，代码编译后大约 100 字节，可以代替原来的前后台系统。

关键词：多任务，线程，就绪，调度

引言

因为资源和成本的原因，前后台系统是 8 位 MCU 上的主流，本文介绍的方法可以在 8 位 MCU 上进行任务切换，代码编译后大约 **100 字节**，这 100 字节也会从原来纯前后台系统改到这种框架下节约的代码来补偿，也就是说，提高了性能，而没有增加代码长度，同时也不需要改变原来的编程方式，只是对原有的函数进行调度。可以在 1K ROM,64BYTE 的 RAM 上运行。

一、调度原理：

- 1、 用一个字节变量的每一位代表一个任务是否就绪，1 为就绪，0 为休眠。
- 2、 这个字节从高位到低位代表的任务，优先级也从高到低。
- 3、 通过查表从就绪的任务中找出最高优先级的任务并执行，同时清就绪标志。

就绪表 ThreadReadyList

1	0	1	0	0	0	0	0
位： 7	6	5	4	3	2	1	0
任务号： 8	7	6	5	4	3	2	1

上表表示有两任务：任务 8 和任务 6 就绪。

因为 8 位优先级高，我们来查表：

```
PRIORITY_TABLE[] = {0, 1, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4};
```

```
ready = ThreadReadyList; // 10100000
```

```
if (ready != 0) {
    if ((ready & 0xF0) != 0U) {
        prionum = PRIORITY_TABLE[ready >> 4] + 4;
    } else {
        prionum = PRIORITY_TABLE[ready];
    }
}
```

查表结果为 4，4+4=8

计算结果为 8，所以优先级为 8 的任务先执行，并清就绪位，完成后再次计算结果为 6，优先级为 6 的任务再执行。

二、任务就绪方法

任务就绪是一个宏，写成宏是因为在某一些 MCU 的编译器中规定不能在中断中调用函数。

```
#define ThreadSet (prio) (ThreadReadyList |= (1<<(prio-1))) //置就绪标志
```

比如在定时器中让优先级为 5 的任务就绪：

```
ThreadSet (5);
```

实际操作为：ThreadReadyList |= 0x10; (编译成汇编代码只一条指令)

把就绪表的第 4 位置 1。

三、任务运行方法

任务运行方法有两种，一种是 switch 一种是函数指针。

因为有些 8 位机的 C 编译器不支持函数指针，所以本文只介绍 switch 方式。(注：作者在 ARM 的多线程框中用的是函数指针)。

在调度原理中我们计算出了优先级号码 prionum

```
switch(prionum){
    case 8://最高优先级
        //任务 8 的函数放在这里
        break;
    case 7:
        //任务 7 的函数放在这里
        break;
    ... ..
```

四、任务就绪表上电初始化：

ThreadReadyList = 0; 在调度前把就绪表清 0 就可以了。

五、完整的任务调度函数：

```
void ThreadScheduler (void)
{
    INT8U prionum,ready;
    prionum = 0;
    ready = ThreadReadyList;
    if (ready != 0) {
        if ((ready & 0xF0) != 0U) { //找出就绪表的最高优先级的任务
            prionum = PRIORITY_TABLE[ready >> 4] + 4;
        }else{
            prionum = PRIORITY_TABLE[ready];
        }
        ready = READY_CLR_AND[prionum];
        OS_ENTER_CRITICAL();//关中断
        ThreadReadyList &= ready;//清就绪位
        OS_EXIT_CRITICAL();//开中断
        switch(prionum){
            case 0:
                break;
            case 8://最高优先级
                //任务 8
```

```
        break;
    case 7:
        //任务 7
        break;
        .....
    case 2:
        //任务 2
        break;
    case 1:
        //任务 1
        break;
    }
}
```

六、程序编写方法

1、主函数

```
void main(void)
{
    InitialMCU();
    ThreadReadyList = 0;
    while(1){
        ThreadScheduler ();
    }
}
```

2、中断函数

```
void ISR_Timer(void)
{
    TmrCtr ++;
    if(TmrCtr > 5){//40ms
        TmrCtr = 0;
        ThreadSet (8); //让定时执行的任务就绪
    }
}
```

```
void ISR_AD(void)
{
    _adf = 0;
    ADValue = _adrh;
    ThreadSet (3);//让计算任务就绪
}
```

3、任务函数

和其它函数没有区别

```
void AlarmOut()
```

```
{
    if(AlarmOutctr > 0){
        AlarmOutctr --;
        PFD_OUT = !PFD_OUT;
        TmrStart(4,15);//1s
    }else{
        ConctrolStat = END_STAT;
        PFD_OUT = 0;
    }
}
```

七、使用任务调度的优势

- 1、多个线程同时就绪时，高优先级先执行。
- 2、高优先级线程，最长等待时间是上一个正执行线程的完成时间
- 3、因为主循环时间最长时是最长线程的执行时间，所以有些中断中执行的代码可以移到任务中。
- 4、可以减少条件语句。
- 5、使软件结构更合理，清晰。

八、结语：

本文介绍的方法在 HOLTEK 系列 8 位 MCU 和 NXP 的 LPC900 中有数十个项目的应用。并且在这基础上把 switch 改为函数指针，加上事件队列和事件延迟后，在 LPC2000 的 ARM 上成功应用。

注：本文中任务和线程等同。