单片机常用程序框架之分时轮询

TonyCode 3月15日

以下文章来源于最后一个bug ，作者未知bug

最后一个bug

1)聊聊编程，计算机技术，电力电子技术，控制算法、linux等；2)谈谈人生、理想和生...

1、程序框架简介

根据多年的编程经验来看，单片机的程序框架大体分为三种分别是顺序执行架构、分时轮询架构和RTOS.（如果还有什么特别的框架欢迎大家留言学习）

顺序执行架构:该框架或许是我们大部分初学者最常用的一种代码编写格式了，比如说首先执行我们的按键检测，然后执行显示数码管，然后去做其他事情！这样一个任务一个任务执行，任务较少时该架构比较简单稳定，当任务比较复杂，逻辑分析就相对比较麻烦，而且程序之间耦合也比较大！需要开发者对程序足够的熟悉，且不便于扩展！

分时轮询架构:这可能是大部分有一定编程基础的程序员或者对小资源单片机进行开发所选用的一种程序架构，今天这也是我们介绍的主题，后面会进行详细介绍。

RTOS:这可能是大部分单片机编程老鸟所选用的一种架构了，RTOS对任务的管理非常丰富，能够让CPU获得一个更大的利用率！那么我们常用的有FreeRTOS,uCos,等等，一般会获得商业使用权等等，也有免费的！

上述的程序框架，各有优劣，需要我们根据具体的情况来选用对应的框架！

2、分时轮询框架详解

从名字上看该框架是通过时间事件发出消息，主任务通过轮流查询对应的时间事件进行运行，因为我们大部分的状态程序都是以时间为节点进行转移和控制的，那么该框架就能够使用，并且我们的中断仅仅是外部给予的一种信号，我们对应的中断服务函数里面进行处理便好，比如：我们的串口接受，当相应接受中断，我们便可以接受到缓存，然后置位相应的标志位，时间任务便会查询、处理。

缺点：该框架的缺点也是很明显的，就是对任务中特殊事件的处理不够及时，不过对于大部分我们大部分项目都还是可以接受的，并不需要实时的处理！

好了，废话不多说，上代码！

```
//TaskManage.h
#ifndef __TASKMANAGE__
#define __TASKMANAGE__

/****************************************************
 * Fuction :数据类型定义区
 * Author  :(公众号:最后一个bug)
 ****************************************************/
#define TRUE                    (1)
#define FALSE                   (0)
```

```
#ifndef NULL
#define NULL                        ((void*)(0))
#endif

typedef unsigned int   u32;
typedef unsigned short u16;
typedef unsigned char  u8;
typedef signed long    s32;
typedef signed short   s16;
typedef signed char    s8;

typedef volatile unsigned int   vu32;
typedef volatile unsigned short vu16;
typedef volatile unsigned char  vu8;

typedef volatile unsigned int  const vuc32;  /* Read Only */
typedef volatile unsigned short const vuc16;  /* Read Only */
typedef volatile unsigned char  const vuc8;   /* Read Only */


/*************************************************************
 * Fuction :任务类型定义区
 * Author  :(公众号:最后一个bug)
 *************************************************************/
#define TASK_NUM_MAX 20

//运行模式
#define TASK_STOP (0)
#define TASK_RUN  (1)

/*************************************************************
 * Fuction :类型定义区
 * Author  :(公众号:最后一个bug)
 *************************************************************/
#pragma pack(1)
typedef struct _tag_taskdata
{
    u8  statue;         //运行状态
    u32 time;           //运行周期
    u32 count_time;     //运行计数变量
    void (*fuc)(void);  //运行函数指针
} stTaskData;

typedef struct _tag_taskmanage
{
    stTaskData task[TASK_NUM_MAX]; //最大任务数管理
    u8 registerTaskNum;            //已经注册的任务
}stTaskManage;
#pragma pack()


/*************************************************************
 * Fuction :功能接口区域
 * Author  :(公众号:最后一个bug)
 *************************************************************/
extern void InitialTaskManager(void);
extern u8  RegisterTask(u32 time, void * taskFuc);
```

```
extern void Task_Process(void);
extern void Task_RunCheck(void);

#endif
```

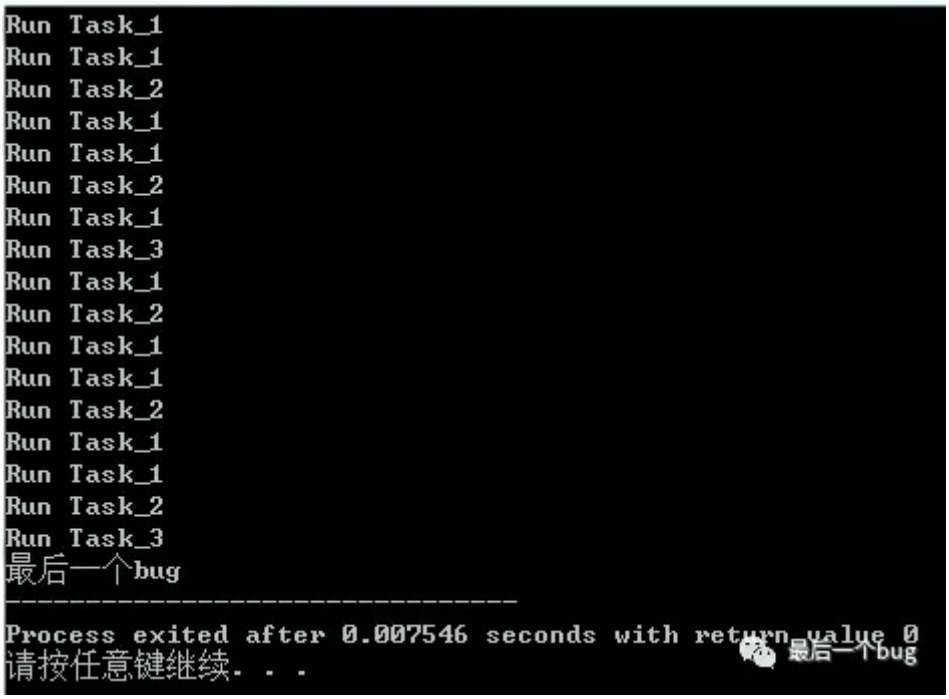
```
//TaskManage.c
#include "TaskManage.h"

/*********************************************************
 * descri  : 变量定义区
 * Author  :(公众号:最后一个bug)
 *********************************************************/
stTaskManage sTaskManage;

/*********************************************************
 * Fuction : InitialTaskManager
 * descri  : 初始化任务管理
 * Author  :(公众号:最后一个bug)
 *********************************************************/
 void InitialTaskManager(void)
 {
    u8 i = 0;
    for(i = 0;i< TASK_NUM_MAX;i++)
    {
        sTaskManage.task[i].statue      = TASK_STOP;    //运行标识
        sTaskManage.task[i].time        = 0;            //运行周期
        sTaskManage.task[i].count_time  = 0;            //运行计数变量
        sTaskManage.task[i].fuc         = NULL;         //运行函数指针
    }

    sTaskManage.registerTaskNum = 0;     //已经注册的任务计数清零
 }

/*********************************************************
 * Fuction : RegisterTask
 * descri  : 注册任务
 * Author  :(公众号:最后一个bug)
 *********************************************************/
 u8  RegisterTask(u32 time, void * taskFuc)
 {
    if(sTaskManage.registerTaskNum >= TASK_NUM_MAX)return FALSE;
    if(taskFuc == NULL)return FALSE;

    if(sTaskManage.task[sTaskManage.registerTaskNum].fuc == NULL) //找到没有使用的任务数据
    {
        sTaskManage.task[sTaskManage.registerTaskNum].statue     = TASK_STOP;    //运行状
        sTaskManage.task[sTaskManage.registerTaskNum].time       = time;         //运行周
        sTaskManage.task[sTaskManage.registerTaskNum].count_time = 0;            //运行计
        sTaskManage.task[sTaskManage.registerTaskNum].fuc        = taskFuc;      //运行函

        sTaskManage.registerTaskNum++;//已经注册的任务计数
        return TRUE;//注册成功
    }

    return FALSE; //全部注册完毕
```

```
 }

/************************************************************
 * Fuction : Task_Process
 * descri  : 任务处理过程
 * Author  :(公众号:最后一个bug)
 ************************************************************/
void Task_Process(void)
{
    u8 taskcount= 0;
    //遍历已经注册的任务
    for(taskcount = 0; taskcount < sTaskManage.registerTaskNum;taskcount++)
    {
        if(sTaskManage.task[taskcount].statue == TASK_RUN)//任务可以运行
        {

            (*sTaskManage.task[taskcount].fuc)();
            sTaskManage.task[taskcount].statue = TASK_STOP;
        }

    }

}

/************************************************************
 * Fuction : Task_RunCheck
 * descri  : 任务运行条件核对
 * Author  :(公众号:最后一个bug)
 ************************************************************/
void Task_RunCheck(void)
{
    u8 taskcount= 0;
    //遍历已经注册的任务
    for(taskcount = 0; taskcount < sTaskManage.registerTaskNum;taskcount++)
    {
        if((++sTaskManage.task[taskcount].count_time) >= sTaskManage.task[taskcount].tim
        {
            sTaskManage.task[taskcount].count_time = 0;
            sTaskManage.task[taskcount].statue     = TASK_RUN;

        }

    }

}
```

```
//使用例程
#include <stdio.h>
#include "TaskManage.h"

/************************************************************
 * Fuction : Task1
 * descri  : 任务1
 * Author  :(公众号:最后一个bug)
 ************************************************************/
```

```
void Task1(void)
{
    printf("Run Task_1\n");
}
/*************************************************************
 * Fuction : Task2
 * descri  : 任务2
 * Author  :(公众号:最后一个bug)
 *************************************************************/
void Task2(void)
{
    printf("Run Task_2\n");
}
/*************************************************************
 * Fuction : Task3
 * descri  : 任务3
 * Author  :(公众号:最后一个bug)
 *************************************************************/
void Task3(void)
{
    printf("Run Task_3\n");
}



int main(int argc, char *argv[])
{
    u16 SimuTime = 0;

    InitialTaskManager();
    RegisterTask(10,Task1);
    RegisterTask(20,Task2);
    RegisterTask(50,Task3);

    while(1)
    {
        Task_Process();

        //模拟定时器中断中调用该函数
        if((++SimuTime) <= 100)
        {
            Task_RunCheck();

        }
        else
        {
            break;
        }

    }

    printf("最后一个bug");
    return 0;

}
```

好了，我们的代码和测试文件代码都已经粘贴上去了，感兴趣的小伙伴可以进行移植、测试和扩展，这里我也附上我的运行现象大家尝个鲜！如下图所示：



该测试程序是100个时间基数任务执行情况，任务1和任务2的时间比例是1:2，任务1与任务3的时间比例也是1:5，刚好与我们注册时候的时间是一致的！

好了，今天的分时轮询单片机框架就写到这里，这里是公众号："最后一个bug"，感谢大家的关注、分享与转发，我们下期再见！

推荐阅读：

Arduino基础入门篇汇总

Arduino拾色器

Arduino提高篇16—六轴姿态MPU6050

关注微信公众号「TonyCode」，查看更多精彩内容。

长按识别图中二维码关注