文章编号:1006-5342(2004)06-0030-04

实时多任务操作系统RTX-51 TINY的应用*

张红兵

(咸宁学院 信息工程学院计算机系,湖北 咸宁 437005)

摘 要:RTX-51 Tiny是集成于Keil C51开发环境中的实时多任务操作系统,该系统可以应用于所有具有8051内核的单片机.它对于从前后台编程模式走向基于实时多任务操作系统的编程模式提供了非常好的平台.本文详细叙述了应用该操作系统进行单片机开发的过程,并给出了实例.

关键词:RTX-51 Tiny;实时;操作系统

中图分类号:TP368.2 **文献标识码**:A

1 采用实时多任务操作系统的意义

在8051系列单片机的开发过程中,人们习惯使用前后台系统来管理各个任务,即应用程序由多个任务组成,它们构成一个无限的循环,在循环中轮流执行各个任务,即一个任务完成后再执行另外一个任务.循环执行的任务被称为后台,当产生中断时,单片机执行中断服务程序,这被称为前台[1].如果某个事件发生了,只有等到程序循环到查询该事件的标志时才能得到处理.虽然在中断服务程序中也能及时处理异步事件,但是为了节省中断服务程序的执行时间,往往在中断服务程序中只是接收信息,然后退出中断,将需要处理的信息留待主循环程序中处理.因此,前后台式的系统不能很好地对异步事件及时响应.实时多任务操作系统(RTOS-Real Time Operating System)以分时方式运行多个任务,好像所有的任务都是"同时"运行,因此它能在任何时候在指定的时间内给出响应.当前,在单片机中应用嵌入式操作系统是大势所趋.

在8051系列单片机上运行操作系统并非易事,因为它们的内部RAM很有限,仅仅128或256字节,扩展外部RAM则会增加产品的成本.Keil公司在其单片机集成开发环境μVision51以上版本中集成了RTX-51 Tiny,它是一种实时多任务操作系统,利用它可以简化具有实时性要求的多任务复杂软件的设计,它可以在所有8051系列单片机上实现而无需扩展外部RAM.

2 RTX-51 Tiny的工作原理

RTX-51 Tiny采用了基于时间片的轮询的任务调度机制.CPU的执行时间被划分为若干时间片,每个时间片的具体时间可依据实际的需要进行选择.系统将应用程序划分为多个可以并行运行的任务,每个任务都是一个无限循环.系统为每个任务分配一个时间片,在该时间片内只执行这个任务.当时间到,在下一个时间片则无条件地执行另外一个任务.当所有的任务都轮询一次后,再回头执行第一个任务.由于时间片很短,一般从几个毫秒到几十个毫秒,因此看起来好像多个任务被同时执行了.

RTX-51 Tiny占用单片机内部定时器0,每隔一定的时间产生一次中断.相邻中断之间的时间就是时间片的长度.在其中断服务程序中进行任务调度以及判断执行了延时函数的任务的延时时间是否到.这种周期性的中断形成RT-X51的系统节拍.

RTX-51 Tiny最多可以定义16个任务,所有的任务都可以同时激活成为就绪态.其占用单片机内部资源除了定时器0外,还占用了7字节的

* 收稿日期:2004-11-02

DATA 空间,3 倍于任务数的字节的 IDATA 空间以及 900 字节的 CODE 空间. 系统时钟为 1 000～65 535 周期,任务切换时间为 100～700 周期.

RTX51 Tiny 提供的系统函数及功能为[2]:

(1)char isr_send_signal(unsigned char task_id); //在中断服务程序中向 task_id 确定的任务发送信号.

(2)char os_clear_signal(unsigned char task_id); //清除由 task_id 确定的任务的信号标志.

(3)char os_create_task(unsigned char task_id); //启动一个由 task_id 指定的任务,只有执行该函数后该任务才被纳入到时间片切换任务的队列中.

(4)char os_delete_task(unsigned char task_id); //停止一个由 task_id 指定的任务,当一个任务执行后不必再运行它了,就可以使用该函数将任务从任务队列中删除,但是该任务的代码依然存在,在需要的时候依然可以激活它.

(5)char os_running_task_id(void); //返回当前任务的 task_id 号.

(6)char os_send_signal(unsigned char task_id); //向由 task_id 确定的任务发送信号,如果该任务正在等待信号,则使任务再次准备执行,否则信号被存储在所访问的任务信号标志内.

(7)char os_wait(unsigned char event_sel, unsigned char ticks, unsigned int dummy);

//暂停当前任务而等待一个或几个事件,诸如时间到、间隔、另一个任务或中断发送的信号. 其中 event_sel 为要等待的事件,只能在 K_IVL、K_SIG、K_TMO 中选择,ticks 为等待超时的定时器 0 产生的时钟脉冲数. dummy 默认为 0.

3　RTX51 Tiny 的使用

RTX51 Tiny 内置在 μVision51 中,因此使用起来很方便.

3.1　系统配置

在 μVision51 或 μVision2 集成开发环境下的 Options for Target 菜单的 Target 栏目,在 Operating 项选中 RTX-51 Tiny,则在编译时自动加入了 RTX-51 Tiny 系统,用户无需增加额外的文件. 在嵌入式系统启动后首先执行该背景程序,用户的应用程序是运行于该背景程序之上的各个任务,RTOS 根据各个任务的要求,进行资源管理、消息管理、任务调度和异常处理等工作.

用户还必须根据实时系统对时间的要求以及任务的多少调整系统配置,方法是对\keil\c51\rtx_tiny 目录下的系统配置文件 conf_tny.a51 的相应部分作出调整.

INT_REGBANK　EQU　1;定义定时器 0 产生中断时中断服务程序使用的寄存器组的组号,默认为寄存器组 1.

INT_CLOCK　EQU　10 000;用于决定定时器 0 多少时间产生一次中断,由它决定操作系统的时钟节拍. 系统默认为 10 000 个机器周期,对于采用 11.0592MHz 晶振的单片机而言,系统的时钟周期为 1.08507×10 000 = 10 850μs = 10.850 7ms.

TIMESHARING　EQU　5;定义任务切换的时间片的长度,即多少个系统时钟周期为一个时间片. 系统默认为 5 个系统时钟,如果晶振为 11.0592MHz,则时间片为

10.8507×5 = 54.253 5ms. 注意:该值不能设定为 0,否则系统不会进行任务切换.

RAMTOP EQU 0FFH;定义单片机中堆栈的最高地址. 注意如果是采用 8051 内核的单片机,内部 RAM 为 128 字节,该值应当修改为 7FH. 系统默认为 8052 内核,内部 RAM 为 256 字节.

FREE_STACK EQU 20;定义堆栈的空间大小,系统默认为 20 字节. 如果在任务切换过程中,堆栈中的自由空间不足,则会关闭中断并进入死循环. 因此要合理设定堆栈的大小.

3.2　使用 RTX-51 Tiny 时编写程序的方法

在程序的开始一定要加入包含文件 rtx51tny.h,即:#include <rtx51tny.h>.

使用 RTX-51 Tiny 时用户程序中不需要 main()函数,系统启动后会自动从任务 0 开始运行. 任务 0 是整个应用程序的开始,因此必不可少. 任务 0 是无须由其他程序启动的,但除此之外其它的任务都必须在启动它后才能运行. 任务的定义是程序中的一段任务函数,该任务既可能纳入到任务调度的排队队列中,也可能永远都不被执行,除非执行了启动该任务的命令.

任务的定义格式为:

```
void function (void) _task_ task_id {
        for(;;) {
            ; //任务循环体
        }
}
```

function 为任务的名称,_task_是定义任务的关键字,task_id 是任务号,其值范围为:0～15,每

个任务对应一个唯一的任务号. RTX-51 Tiny 不支持可剥夺式任务调度,所有的任务的优先级都是相同的,因此任务号的选择并没有什么特殊要求,但是要注意任务 0 是所有任务的开始,所以一般在任务 0 中启动其它的任务. 每个任务都是一个死循环,没有参数和返回值.

中断服务程序的编写方式与不使用 RTX-51 Tiny 操作系统下的编写方式相同.

4 应用实例

在视频监控领域经常要用到协议转换器,它将一种协议的串行数据转换成另外一种协议的串行数据. 在设计一款协议转换器时使用了 RTX-51 Tiny,整个应用被分为三个任务和一个中断服务程序. 一个任务是从串口接受数据,对其结构和含意进行判断,并将其转换为对应协议的数据;一个是将转换后的数据从串口发送出去;还有一个是喂看门狗;当有串口数据成功发送或接受时,产生中断,在中断服务程序中向等待数据处理的任务发送信号. 示意性程序如下:

```
#include <rtx51tny.h>
#include <reg52.h>
#define RECEIVE 1 //任务 ID 的定义
#define TRANSMIT 2
#define WATCH_DOG 3
/*********************
    任务 0:初始化系统,启动其它的任务
*********************/
void initial(void) _task_ 0 {
    initialization(); //初始化系统
    os_create_task(RECEIVE); //启动接收任务
    os_create_task(TRANSMIT); //启动发送任务
    os_create_task(WATCH_DOG); //启动喂看门狗任务
    os_delete_task(0); //终止该任务
}
    /*********************
    任务 1:等待串口中断发送的信息,并将从串口接收到的数据进行分析和转换
*********************/
    void receive_task(void) _task_ RECEIVE {
    for(;;){
            os_wait(K_SIG,0,0); //等待信号
            ; //数据处理
        }
    }
/*********************
    任务 2:等待串口中断发送的信号,并向串口发送转换后的码流.
*********************/
    void transmit_task(void) _task_ TRANSMIT {
        for(;;){
            for(i=0;i<7;i++){
                os_wait(K_SIG,0,0); //等待信号
                ; //发送数据
            }
        }
    }
/*********************
    任务 3:定期喂看门狗
*********************/
    void watch_dog_task(void) _task_ WATCH_DOG {
    for(;;){
        watch_dog = ! watch_dog; //喂看门狗
        os_wait(K_TMO,1,0); //每隔一个时钟脉冲喂一次看门狗
        }
    }
    /*********************
    串口中断服务程序
*********************/
    void serial_ISR(void) interrupt 4 using 2 {
    if(RI){
        RI = 0;
        isr_send_signal(RECEIVE); //向任务 1 发送信号
    }
    else if(TI){
    TI = 0;
    if (updata_flag == 1) } //只有当刷新标志置位的时候才发送信号
    isr_send_signal(TRANSMIT); //向任务 2 发送信号
            }
        }
```

}

5　结束语

在嵌入式系统中应用实时多任务操作系统除了提高对事件的响应速度外，还能显著提高编程效率和稳定性，人们可以集中精力编写某个任务，而不必担心任务之间的关系，任务调度完全由操作系统来完成. RTX-51 Tiny 集成于 keil 公司提供的 C51 开发环境中，且常用的系统函数只有 5 个，因而用户将其嵌入到应用中十分方便. 但是在应用过程中也要注意几个问题，一是 RTX-51 Tiny 是基于时间片的任务轮询机制，不支持可剥夺式任务切换，因此在对事件的响应速度要求非常严格的情况下不很适用；二是系统本身占用了单片机的定时器 0，用户不能将定时器 0 用于其它用途. 三是系统本身占用了 CPU 的运行时间和部分 RAM、ROM，用户在使用的过程中都要考虑到这些因素.

参考文献：

[1]陈明计，周立功. 嵌入式实时操作系统 Small RTOS51 原理及应用[M]. 北京：北京航空航天大学出版社，2004.

[2]徐爱钧，彭秀华. 单片机高级语言 C51 Windows 环境编程与应用[M]. 北京：电子工业出版社，2001.

The Application of Real Time Operating System RTX-51 Tiny

ZHANG Hong—bing

(Department of Computer, School of Information Engineering,
Xianning College, Xianning 437005, China)

Abstract: RTX-51 Tiny is a real time operating system integrated in Keil C51 integrated development environment. It can be applied in all Micro controller with 8051 core. It helps electronic engineers to turn from Round-Robin model to real time operating system. An example illustrates the development processing with the operating system in detail.

Key words: RTX-51 Tiny; Real time; Operating system