第 25卷 第 1期
2005年 1月

桂 林 工 学 院 学 报
JOURNAL OF GUILIN UNIVERSITY OF TECHNOLOGY

Vol. 25 No. 1
Jan. 2005

文章编号：1006－544X（2005）01－0128－05

基于 8051系列单片机的实时操作系统设计

李明强，程小辉，沈 旭

（桂林工学院 电子与计算机系，广西 桂林 541004）

摘 要：8051系列单片机系统广泛应用于工控、仪器仪表、通信等领域，为了避免其软件开发的重复性工作，提高所编软件的可靠性，结合自主开发的、基于 AT89C51单片机为核心的硬件平台，实现了一个基于该平台的实时操作系统. 该操作系统具有一个基于 C语言的、实时多任务的内核，有较好的移植性.

关键词：实时操作系统；嵌入式系统；单片机；调度器

中图分类号：TP316.2 文献标识码：A

单片机在嵌入式微控制系统应用中具有十分重要的地位. 在嵌入式系统中使用实时操作系统已经成为嵌入式应用的一种趋势，是单片机高水平应用开发的一个标志[1－4]. 一个好的实时操作系统可大大提高控制产品的研制效率，缩短开发时间，有利于多人的分工协作，用 RTOS开发的产品稳定性、可靠性也会得到提高.

1 硬件平台简介

1.1 8051系列微控制器简介

目前，8051系列芯片已达 400多种，可分成 3个主要类别：标准 8051系列、小型 8051系列和扩展 8051系列. 小型 8051系列是 8051系列芯片中低成本的类别，端口管脚数目少，不支持片外存储器，主要应用在低成本的消费类产品；扩展 8051系列是 8051芯片中加有扩展的片上设备，如 CAN总线控制器、DAC、ADC等，其端口管脚数目比较多，且最近的此类芯片都支持大容量的片外存储器，主要应用在工业及汽车系统中[5]. 本文所述操作系统的硬件平台使用的微控制器是标准 8051系列芯片. 因为小型以及扩展的 8051系列芯片都是由标准 8051系列芯片衍变而来，所以本系统也可以在任何基于 8051系列芯片的嵌入式系统上进行移植.

1.2 硬件平台

一个典型的单片机应用系统包括基本部分、输入部分（测控增强部分）和输出部分（外设增强部分）. 基本部分主要是单片机及其外围芯片的扩展（如 RAM和 ROM）、功能键盘、显示器的配置等，它们是通过内总线连接而成. 测控增强部分主要由传感器、变送器、转换器等接口及伺服驱动控制接口构成. 外设增强部分主要是外设接口，它通过 I/O口或扩展的 I/O口构成，可接打印机等外设. 该实时操作系统是基于 AT89C51单片机为核心的单片机系统硬件平台上实现的，其结构如图 1所示.

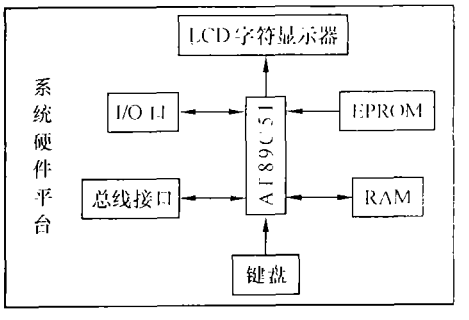

图 1 硬件平台结构

Fig. 1 Structure of hardware platform

收稿日期：2004－12－13

基金项目：广西区科技攻关项目（桂科攻 02350142）

作者简介：李明强（1980－），男，硕士研究生，研究方向：嵌入系统.

2 实时操作系统的设计

2.1 实时系统的特点

对于实时响应时间，不同的应用系统有不同的要求，即使在同一个应用系统内部，不同的应用场合也有不同的要求．在实时系统里，除了实时响应外，另外一个问题是系统的流通量或吞吐量，实时系统的吞吐量可以定义为每单位时间系统可以处理的事件个数[2]．

实时任务具有截止期，分硬实时、软实时和固实时．硬实时是指如果响应超过了任务截止期，可能会导致严重后果；软实时是指虽然响应超过了任务截止期，但还是有一定的价值；固实时是指如果超过了截止期才响应处理，就完全没有价值了．实时任务具有可预测性，在最坏情况下的任务的执行时间以及所需数据和资源，都要求对最坏情况的预测与实际的差别尽可能的小，即使在出现峰值负载时，也应满足其截止期．

2.2 实时操作系统与实时系统的关系及其特点

实时操作系统是实时系统软件的基础，所实现的功能是系统资源的管理和机器功能的扩充[2]．因此实时操作系统和通用操作系统是有一定差别的，有其自己的一些重要特征，包括进程切换快、中断被屏蔽时间短、规模小等．

2.3 嵌入式系统的 2种触发方式

在嵌入式系统设计时，通常采用事件触发和时间触发 2种方式来实现[1]．

事件触发行为往往通过使用中断实现，事件触发系统在系统总体结构上往往通过提供多级中断服务程序来支持该功能．如果多个中断源可能在“随机的”时间间隔产生中断，则中断响应可能被遗漏[1]．8051的体系结构支持高和低两种不同的中断优先级，将可能出现中断信息丢失的情况．例如，有高优先级的中断 1发生，并在极短的时间内又发生了中断 2，这时无论中断 2是高优先级还是低优先级中断，中断 2的响应都会被延迟，并在一些情况下（如被延迟时间过长等）被完全忽略．

因此，在某些与安全相关的应用系统中选用时间触发方法，这样设计人员能预先安排可控的顺序，保证一次只处理一个事件，提高了系统的可靠性并降低 CPU的负荷，减少存储器的使用量．

2.4 调度器的设计

调度器可以看作是一个简单的操作系统．从底层的角度来看，调度器可看作是一个由许多不同任务共享的定时器中断服务程序[1]，允许以周期性或单次方式来调用任务，其分为合作式调度器与抢占式调度器两大类：合作式调度器提供的是一种单任务的系统结构，比较简单，用很少的代码就可实现，每次只需为一个任务分配存储器，具有可预测性和可靠性，但是在当前的程序运行期间，系统对外界的变化不敏感；抢占式调度器提供了一种多任务的系统结构，比较复杂，必须为抢占任务的所有中间状态分配存储器，对外部事件的响应速度快，但其可预测性和可靠性较低．

实时系统的任务都有一个截止期，单独使用抢占式调度方式或合作式调度方式都不能完全满足实时系统的要求，必须使用一种混合式调度器，根据实时系统的要求来设置定时器的定时时间．假设有 2个任务，1个必须在 20 ms内完成，另 1个必须在 30 ms内完成，因此取它们的最大公约数 10 ms来作为定时器的定时时间．使用定时器 0的模式 1来进行定时，则可以计算出 T0的初值为 T0D8F0H，程序实现如下（部分程序）：

```
#define pre_t0H 0xD8
#define pre_t0L 0xF0 /根据定时时间计算出的初值
void timer_t0_init(void)
{
TMOD = 0x10; /设置定时器 0的工作模式
TH0 = pre_t0H;
TL0 = pre_t0L; /送初值,定时时间为 10ms
TR0 = 1; /启动定时
ET0 = 1; //T0开中断
EA = 1; //CPU开中断
}
void attemper_timer0(void) interrupt2 using 1 /定时器 0的中断服务程序
{
/定时时间到,判断是否进行任务切换(选择最高优先级的任务来执行,代码略)
}
```

调度器根据在系统进行初始化时用户的预设值将中断和某些外设设置为抢占式任务，抢占式任务是最高优先权的任务．在调度期间，只有 1个抢占式任务，该任务一旦开始运行将连续运行到其完成，以减少任务切换所耗费的时间，因此要求这些中断处理任务和外设处理任务的时间必须小于定时器的时间间隔，尽量更短，否则会影

响系统对其他中断的响应或对其他设备的处理. 对于除用户预设的中断和外设之外的事件将采用分时方式进行处理.

2.5 任务及任务间通信

每个任务都分配在不同的存储空间, 通过指针变量 *TASK_pnum来指向任务的首地址. 每个任务都有自己的存在状态 (运行态、就绪态、休眠态、挂起态和被中断态)和优先级 (为了解决任务优先级翻转问题, 系统可以通过函数 void prio_change(unsigned int * task_adr, unsigned char prio_st, unsinged char prio_ced)来动态调整任务的优先级), 任务间使用信号量或消息队列来进行互相通信. 任务的数据结构定义如下:

```
typedef struct
{
unsigned char task_name;  /任务名
unsigned char realtask;  /是否为实时任务, 0为否, 其他为是
unsigned char prio;  /任务优先级
unsigned char state;  /任务状态
unsigned char err;  /出错信息
unsigned char *ptos;  /栈顶指针
unsigned int stk_size;  /堆栈大小 (以字节为单位 )
} TASK;
```

2.6 中断处理

系统中的实时任务不仅包括某些中断处理, 还包括部分外设处理任务. 中断是一种硬件机制, 用于通知 CPU 有异步事件发生了, 系统对中断 (T0除外) 分 2步处理, 第 1步是中断响应, 第 2步是中断处理, 共有 3种处理方式.

(1) 在中断发生之前不存在实时任务, 并且在中断处理过程中也没有其他的中断发生. 中断响应: 系统进入临界区, 保存当前任务的信息, 退出临界区并返回系统. 中断处理: 由调度器将该中断任务设置为实时任务并执行到结束, 系统进入临界区, 恢复被打断的非实时任务的信息, 并执行该任务.

(2) 在中断发生之前不存在实时任务, 但在中断处理过程中有其他的中断发生. 中断响应: 系统进入临界区, 保存当前任务的信息, 退出临界区并返回系统. 中断处理: 由调度器将该中断任务设置为实时任务并执行到结束, 选择所发生的中断中急需处理的任务设置为实时任务并立即执行该任务.

(3) 在中断发生之前已存在实时任务, 无论在中断处理过程中有没有其他的中断发生, 系统的处理过程都是一样的. 中断响应: 系统进入临界区, 修改中断发生标识, 退出临界区并返回系统. 中断处理: 系统继续执行当前实时任务直到结束, 选择所发生的中断中急需处理的任务设置为实时任务并立即执行该任务.

临界区是指在此段时间内, 不允许有任何事件来打断当前任务, 比如在保护和恢复中断现场时, 为了避免现场信息受到破坏, 一般会关掉中断进行处理. 如外部中断 1的处理程序如下 (通过外部中断 1和 P1口扩展了 4个中断, 外部中断 1比外部中断 0的优先级低).

```
#define init_open() EA=1 //CPU 开中断
#define init_close() EA=0 //CPU 关中断
extern unsigned char data init_symbol=0xE0 //该变量低 5位, 表示 5个中断发生的标识, 其中低 4位分别对应 P1.0 ~P1.3扩展的中断, 第 5位 (低到高 )表示中断 0
extern unsigned char data real_task=0
void init1_resp(void) interrupt 3 using 3
{
init_close();
if (real_task==0)
{ /保存当前任务现场 }
else
{ /保留 P1的低 4位, 第 5位清零, 然后与 init_symbol相或 }
real_task++;
init_open();
/返回系统
}
void init1_deal(void)
{
if( real_task==1)
{ /执行中断处理程序
init_close();
/恢复任务现场
init_open();
real_task--;
/返回系统执行被中断的任务
}
else
{ /继续当前任务
real_task--;
/判断下一个优先级高的中断并调用该中断处理程序 (代码略 )
}
}
```

2.7 存储器管理

使用链表对存储器进行管理，对每个未分配的空闲区按位置顺序进行拉链，每次分配时，从头搜索空闲表，找到第 1个满足要求的空闲区就进行分配，但其大小不一定刚好等于所需要的，将其分成 2个区，1个按所需分配出去，另 1个则仍为空闲区并留在原位置. 链表的数据结构定义如下：

```
typedef struct
{
MEM_chainb * prve, * next; /前指针和后指针
unsigned int addr_start; /空闲区的起始地址
unsigned int addr_size; /空闲区的大小 (以字节为单位计算 )
}MEM_TABLE;
```

2.8 外围设备的管理

外部设备主要使用该外设对应的驱动程序. 每个所使用到的设备都有自己的私有存储空间，数据结构随设备的类型不同有所变化. 对于需要 CPU周期性服务的设备 (如本系统中的 LCD和键盘)，给它们分配相应的任务，与用户任务一起调度，实现起来比较简单. 有些设备具有自己的中断 (如串口)，则可以利用消息队列将用户任务需要的服务通过消息队列排队、缓冲起来，利用中断功能服务. 对于既没有自己的中断又不需要 CPU周期性服务而且速度较慢的设备，定义一个宏，在宏中申请信号量，然后调用对设备操作的函数并释放信号量.

3 系统调试

由于当初考虑只是要实现一个实时操作系统，所以笔者设计的硬件平台只是向用户提供了总线接口 (增大了驱动后的总线接口)、外部中断接口 (对外部中断进行了扩展) 和 I/O口 (RAM的地址空间 0FF00H ~0FFFFH用作了 I/O口)，并没有将测控增强部分 (模拟量、开关量和数字量的检测) 及外设增强部分 (外设、伺服控制和开关量控制) 放入硬件平台，但在操作系统中编写了某些模块的驱动 (如 A/D模块的 AD574A, ADC0809, D/A模块的 DAC0832等). 将该操作系统烧入了程序存储器后再进行调试.

3.1 键盘和 LCD字符显示器的调试

该步调试要求在 LCD字符显示屏上显示预定键对应的英文字母. 在没有其他外部事件发生时，测试结果良好；有外部事件发生时，键盘的反应较慢，有时出现不能对按键的反应，并且 LCD字符显示器的刷新时间间隔变长. 解决方法：将软件延时程序改为用定时器进行延时.

3.2 中断和外设调试

中断调试要求能及时响应中断并处理中断任务 (短任务)；外设调试要求能正确处理外设向 CPU发送的信息并能够将处理后的信息正确地送出去. 用预留的一个 I/O口 (实际是某一个地址) 来模拟中断发生，中断采用边沿触发方式，这样对外部地址的一个读或写操作占用的时间完全满足外部中断的高电平和低电平持续时间不短于一个状态周期的要求；用拨码开关、电阻、二极管和 74HC245模拟输入设备；用 74HC373、电阻和二极管模拟被控的外设. 要求在有中断发生时从指定设备上读出数据，并将读取的数据取反，取反后的数据写入到指定的输出设备. 测试结果显示，该操作系统能够在该硬件平台上良好的运行，但在模拟工业的某些环境下，在预留的接口上有时会有干扰信号输入，所以对系统软件进行了修改，即在操作系统进行初始化的时候，提示用户指出所用到的外部中断接口及硬件平台预留的 I/O口，系统对于除此之外的中断和预留的 I/O口所产生的输入信号一律拒绝响应. 如果用户需要动态的增加或删除外设，可通过键盘进入系统设定模式进行设置，这时系统会通过函数 void ex_init_set(unsigned char ex_init_num)和 void ex_prID_set (unsigned char ex_ID_num)来修改初始化时的用户对外部中断和外设的设定，在该模式下，系统将关掉所有外部中断和外设，只响应键盘和刷新屏幕，待设置完成之后再恢复到系统的实时模式下.

4 实时操作系统在智能温度调节器中的应用

4.1 智能温度调节器简介

智能温度调节器由系统控制模块、键盘模块、显示模块、温度采集模块和温度调节模块 5部分组成. 系统控制模块由单片机 (AT89C51) 和数据存储器 (62256) 组成；键盘模块由 1个 16键键盘构成；显示模块由 1个 4×16的字符型液晶显示器组成；温度采集模块由 1个温度传感器、1片 AD574A及相关电路组成；温度控制器由 1片 DAC0832及相关电路组成.

4.2 实时操作系统在该仪器中的应用

根据该系统的功能模块将其分成 5个普通任务和 1个中断任务，5个普通任务按优先级从高到低进行如下排列：数据采集任务、温度调节任务、键盘扫描任务、键盘处理任务和显示任务. 中断任务的功能是判断数据采集是否结束.

由于该仪器要求每 0.5 s采样 1次，屏幕每 2 s刷新 1次，键盘去抖动的时间为 50 ms，故将定时器 T0的定时时间设定为 50 ms，作为调度器的基本调度时间单位. 系统的存储空间进行了如下划分：16个字节作为键盘输入缓冲区，64个字节作为显示输出缓冲区，8个字节作为温度调节任务初始值的存放地址空间（包括基准值等），其余的作为温度采样值的存储空间，以方便观测温度的变化规律.

在该系统中温度的采样值是一个非常重要的变量，涉及到数据采集、温度调节和显示 3个任务，因此使用信号量进行这 3个任务之间的通信. 对信号量做如下设定：用 1个字的空间来存放该变量，低字节存放变量值，高字节表示现在的状态. 系统中只有 1个中断任务，因此中断任务的处理相对非常简单.

5 结束语

实时多任务操作系统是当前单片机软件系统的发展方向[3]，它使整个计算机系统实现了自动化，具有高效率和高可靠性，实用价值较高. 很多实时系统应用在非常重要的地方，所以对系统的可靠性和安全性有着比较高的要求. 实时操作系统在智能温度调节器上能够稳定的运行，但该系统在可靠性与安全性等方面仍需进一步的完善.

参考文献

[1] Michael J Pont. 时间触发嵌入式系统设计模式 [M]. 周敏译. 北京：中国电力出版社，2004：5－10.

[2] 郑宗汉. 实时系统软件基础 [M]. 北京：清华大学出版社，2003：7－9.

[3] 陈明计，周立功. 嵌入式实时操作系统 Small RTOS 51原理及应用 [M]. 北京：北京航空航天大学出版社，2004：12－14.

[4] 谢宜仁. 单片机实用技术问答 [M]. 北京：人民邮电出版社，2003：7－10.

[5] Michael J Pont. C语言嵌入式系统开发 [M]. 陈继辉译. 北京：中国电力出版社，2003：11－13.

Design of RTOS Based on the 8051 Microcontrollers System

LIM ing-qiang CHENG Xiao-hui SHEN Xu

(*Department of Electronics and Computer Science, Guilin University of Technology, Guilin* 541004, *China*)

Abstract: The system of 8051 single-chip computer is widely applied in industry control apparatus and communication fields. To avoid the repeated work in program writing and improve reliability of the software, a RTOS based on the platform can be realized by the system combined with the hardware platform of AT89C51 single-chip computer. The operation system has a realtime multitask core written in C, which is easy to be transplanted.

Key words: RTOS; embedded system; single-chip computer; scheduler