51 单片机多任务编程设计及应用

徐 华[1,2]

(1. 重庆大学软件工程学院，重庆 400044；2. 中国兵器工业第五九研究所，重庆 400039)

摘 要：本文论述了在 51 系列单片机系统中，一种多任务系统编程设计方法。该方法不基于实时操作系统 RTOS 分时操作的思路和采用中断切换任务。本文通过一个具有 4 位 LED 数码显示，12 键的键盘扫描和用串行口与其它系统交换数据的简单系统为例，说明该方法编程具有硬件设计简单、单片机工作效率高，实时性强等特点。该方法编程采用汇编语言，但根据相同的原理和思路，也不难用到 C51 语言编程上面。

关键词：51 单片机；多任务；动态显示；键盘显示控制

中图分类号：TP313 **文献标识码**：A doi: 10.3969/j.issn.1003-6970.2011.03.006

51 Single-Chip Computer Multitask Programming Process and it's Application

XU Hua[1,2]

(1.*College of Software and Engineering*,*Chongqing University*,*Chongqing* 400044, *China*;
2.*No*.59 *Institute of China Ordnance Industry*,*Chongqing* 400039, *China*)

【Abstract】 This paper explain One Multitask Programming Process on 51 Single-Chip Computer System. Unlike the Principle of Real-time Operating System , This Process is neither based on the time sharing Operation nor using interrupt for switchover task. One simple system consisted of the 4 digits LED digit display, the 12 keys scan keyboard and the series port used for communication with other system is taken for example, to illustrate this process is used in programming, Single Chip Computer system has characteristic of simpler hardware, higher efficiency of operation and better real-time. This process is based on assemble language, but it is easy for C51 language according to the same principle and thinking.

【Key words】 51 Single-chip Computer; Multitask; Dynamic scan Display; Keyboard and display control

0 引 言

51 单片机在微型智能控制系统中应用广泛。随着人们对控制系统要求的不断提高，针对 51 单片机不具备实时多任务支持功能，在控制系统的进一步发展很受限制的情况，人们做了很多关于 51 单片机多任务实时编程的研究和实验。罗江等在四川省教育厅资助的基金项目《基于多任务机制的 51 单片机在微型智能控制系统中的应用研究》中，借鉴多任务操作系统的设计特点，提出了利用时间片分配机制，实现多任务分时轮流执行，和利用中断强行切换任务的多任务编程方法[1]；在 www.ddvip.com 的单片机技术交流中，阮元提出了依据分时操作系统思想实现单片机多任务编程的方法[2]；厦门大学的王辉堂等在一安防系统的设计中，通过对嵌入式实时操作系统 RTOS 的分析，认为其核心是利用中断切换任务，提出了用 C51 编程实现的多任务编程方案[3]；美国 Keil 公司开发的 MCS51 系列单片机的实时多任务操作系统 RTX51，占用定时器 T0 中断产生时间片来切换任务[4-5]；此外，还有时间片轮转算法[6]、时分多线程[7]等多种单片机多任务设计方法在实际系统中应用。

综上所述，在目前 51 单片机多任务编程工作中，大多采用了基于实时操作系统 RTOS 分时操作的思路和采用中断切换任务。但也有人认为，这种任务切换产生大量数据，额外占用系统资源，不适合资源有限的单片机系统[8]。本文提出的 51 单片机多任务编程方法，不采用时间片，而是基于一个完整过程切换任务，并将任务调度分配到各任务内部，任务的切换和调度都不占用系统额外资源。通过实际系统应用证明，用该方法设计系统，硬件电路更简单，单片机工作效率更高。

本文首先阐述多任务编程设计方法的原理，然后介绍几种典型任务的设计方法，最后通过一个简单的实例来说明该方法的可行性。

1 多任务设计原理

多任务要求系统在同一时间执行多个任务，对于一个处理器，并不可能在同一时间运行多个任务程序，而是按时间片在各个任务间快速切换执行来完成多任务要求的。这是基于实时操作系统 RTOS 的方法。本文提出的方法也是按时间片切换任务的，但有所不同的是，执行任务的时间不是由定时器平均分配的，而是按照执行任务中一个完整过程的时间来自动分配的。

在单片机系统设计中，可按系统的功能或模块划分为任

作者简介：徐华 (1961-)，男，高级工程师，重庆大学工程硕士研究生，主要研究领域：自动控制工程，软件工程，小区智能化系统。

务,而每个任务可按具体作业细分为各个过程。可见任务由过程组成。按时间片分配任务的设计,任务的调度可分为两级,一是对任务的调度,二是在每项任务中对过程的调度。由于采用两级调度,比较繁琐,也占用了系统较多资源。如果不考虑对任务的调度,直接调度各任务的过程,其系统运行的效率就能极大地提高。同时,在任务中主动设置切换点自动切换任务,切换时不需保留大量现场数据,系统效率就会更高。再进一步,将过程的调度分配到各个实际任务之中,不设专门的过程调度表。系统的运行效率就会达到最高。为此本文提出的 51 单片机多任务编程的原理是:

1) 将系统的各任务依次排成队列,处理器依次执行各任务,在执行完最后一项任务后,马上回头执行第一任务,并以此循环。

2) 在每次执行一项任务时,只执行该任务的其中一个过程,并完整执行。其它的过程需等待下一轮执行该任务时才有机会执行。这可保证系统尽快从本任务切换到其它任务。

3) 每个过程不包括循环延时、等待等浪费 CPU 时间的程序,循环延时、等待等程序将作为一个特殊过程独立设计。

4) 常规任务(甚至实时任务) 都设有一个空执行过程,以便系统跳过该任务的执行。优先任务或抢占式任务可利用此功能保证任务优先执行或独占运行。

5) 系统不设专门的过程调度表,过程的调度在实际任务之中进行。

6) 中断只用于调度过程,不处理任务,更符合系统实时性要求。

7) 系统、任务和过程间均通过全局变量共享和交换数据。任务、过程间切换不保留现场数据和传递参数。

由此可见、系统按完整过程(最小作业单元)自动切换任务,不需保留临时现场数据,不需定时被动切换,不需额外的调度表。和单任务编程相比,多任务编程也没有占用系统任何额外资源。其结构和代码的可读性也没有较大的改变。

实际上,设计操作系统与设计应用系统最本质的区别是,操作系统面临的任务是不确定的,在设计完成后还允许应用添加新的任务;而应用系统面临的任务是事先定制的。因此,多任务应用系统的设计可不完全基于操作系统分时的思路。本文提出的多任务编程方法,就不需要强制的时间片划分,也不需要占用中断切换任务。

2　多任务编程的实现方法

2.1　一般任务的设计

任务分为实时任务和常规任务,实时任务是需无条件执行的任务,即在系统每次扫描任务队列时都必需执行的任务。通常是显示,外围检测等实时任务;常规任务是在满足一定条件时才启动的任务,通常被中断或其它任务调度,也能被中断或其它任务中止。为将实时任务与常规任务排在同一队列中执行,我们为常规任务增设了一个特殊过程,在调度这个过程时,该任务不执行任务的实际操作。这个过程称为空过程。例如,下面例举的任务包括一个空过程和三个实际操作过程,在任何其它任务中,通过屏蔽本任务的空过程,即 SETB　00H,就可启动本任务,同理, CLR　00H 就可在任何时候中止本任务的执行:

```
PROC:JNB  00H,    PROCE  ;空过程调度,不执行本任务
JB    11H,  PROC2  ;11H 置位时,执行过程 2
JB    10H,  PROC1  ;10H 置位时,执行过程 1
PROC0:……;过程 0:本过程通常为任务初始化
SETB  10H          ;调度过程 1
AJMP  PROCE
PROC1:    …….    ;过程 1
SETB  11H          ;调度过程 2
AJMP  PROCE
PROC2:    ……      ;过程 2
CLR   11H
CLR   10H
CLR   00H          ;任务完成,调度本任务的空过程
PROCE:NOP          ;空过程
```

由任务设计可以看出,过程是通过消息(位寻址变量置位)来调度的。每次执行任务只执行该任务的一个过程,并且过程不能包括循环延时、等待等代码,以保证系统轮流执行多任务的实时性。如果一个过程必需包括等待或延时,则将过程从等待或延时处分解为两个过程,并在两个过程中插入专门的延时或等待过程。延时和等待过程的设计说明如下。

2.2　延时等待过程设计

延时,等待等过程在程序设计中是必不可少的,为不影响系统的实时性,多任务编程的延时和等待需要特殊设计。

延时分为短延时和长延时,短延时是通过一个和几个计数变量计数就能完成的延时,长延时则通常利用计数变量配合计时器中断来完成延时。短延时可设计成任务中的一个特殊过程,长延时可设计成任务中的一个特殊过程或一个独立的特殊任务。

1) 短延时,以下程序设计了一个特殊过程 PROC1:一个计数变量 30H 递减实现延时,其计数周期是系统执行任务队列中全部任务所需的时间。

```
PORC:JNB   00H,  PROCE
     JB          10H,  PROC1
     MOV   30H,  #03H     ;30H 的值控制延时时间的大小
     SETB  10H
     AJMP  PROCE
PROC1:DJNZ  30H, PROCE
```

```
;以每次执行完所有任务时间为周期的延时
    CLR  10H
    CLR  00H      ;00H 清 0 本任务执行完毕
PROCE:NOP
    ……            ;其它任务过程
LJMP PROC
```

2) 长延时,利用定时器中断配合计数变量设计延时,20H 清零设置一个等待过程。等待时间到,20H 由定时器中断程序置位。20H 的置位周期可由编程确定,本例为 5ms:

```
PORC:JNB      00H,  PROCE
    JB        10H,  PROC1
    MOV  30H,   #64H   ;30H 的值初始化为 100
    CLR   20H
    SETB  10H          ;调等待过程
    AJMP  PROCE
PROC1:JNB  20H, PROCE
    CLR  20H
DJNZ  30H,PROCE     ;以 5mS 为周期的延时
    CLR   10H       ;总延时为 500mS 左右
    CLR   00H       ;00H 清 0 本任务执行完毕
PROCE:NOP
……
LJMP  PROC
PRIT0:    ……        ;定时中断程序,定时设置为 500uS
DJNZ  70H,PRIT0D
    MOV  70H,   #0AH
    SETB  20H          ;定时周期为 5mS
    ……
PRIT0D:   NOP
……
    RETI
```

2.3　串行口连续发送数据任务

串行口发送数据需占用较长时间,连续发送多个字节数据将影响多任务系统的实时性。基于时间片的多任务切换系统,一般通过计算发送一帧数据所需的时间,采用约大于这一时间的定时方式,来作为发送两帧数据间的时间间歇,而不是在检测到发送数据缓冲区 SBUF 为空时立即发送后一帧数据。其连续发送数据的实时性较差,不能保证在数据总线上较好地与其它系统协调工作。本方法通过以下程序解决这一问题(假设数据缓冲区 30H-3FH 的 16 字节数据需要从串行口发送):

```
PROC:JNB  00H, PROCE   ;串行口发送数据程序
    JB       10H, PROC1
    MOV  R0, #30H
;发送数据任务初始化,指针 R0 设为 #30H
    MOV  R3, #10H      ;发送数据为 16 字节
    SETB  10H
PROC1:JB   30H,  PROCE
          ;等待过程:30H 在串行发送中断程序中清 0
    SETB  30H     ;30H 置 1,调度本任务的等待过程
    MOV  SBUF, @R0
    INC   R0
DJNZ  R3,  PROCE       ;
    CLR   10H                          ;
    CLR   00H     ;00H 清 0,所有数据发送完毕
PROCE:    NOP
……                ;其它任务过程
AJMP PROC
SINP:      ……
   ;串行中断程序
JNB  TI,  SINP1
     CLR  TI           ;发送缓冲区空
     CLR  30H   ;30H 清 0,结束发送任务的等待过程
SINP1:  JNB  RI,  SINPD
      CLR  RI
SINPD:  NOP
    ……
    RETI
```

3　单片机多任务编程实例

3.1　说明

本实例是从楼宇可视对讲系统的门口主机系统中简化出来的,可视对讲系统是一计算机支持的分布式多设备协同工作系统,总线制通信对串行口收发数据具有很高的实时要求。门口主机系统包括 4 位 LED 数码显示,12 键键盘,串行口发送、接受数据,处理数据,以及控制开锁、发出蜂鸣声,密码数据闪存,计时,切换工作状态等多项任务。采用多任务编程是十分必要的。本实例就其 LED 显示,键盘扫描和串行口交换数据功能来说明该系统的多任务编程过程。

3.2　硬件设计

能完成上述几项功能的硬件电路设计如图 1 所示,这是一个单片机最小系统。4 位 LED 数码管采用动态显示,键盘为矩阵扫描键盘,这两个模块在单任务设计中,都需要较多的延时过程,很难保证系统的实时性,操作按键时还会造成显示闪烁或停顿。为此,很多类似设计都增加外围硬件或采用专用芯片如 CH451 来驱动[9],而多任务编程可省去这些硬件。图中 75176 芯片的 AB 端接楼宇对讲系统的 485 总线,与其它设备交换数据协同工作。

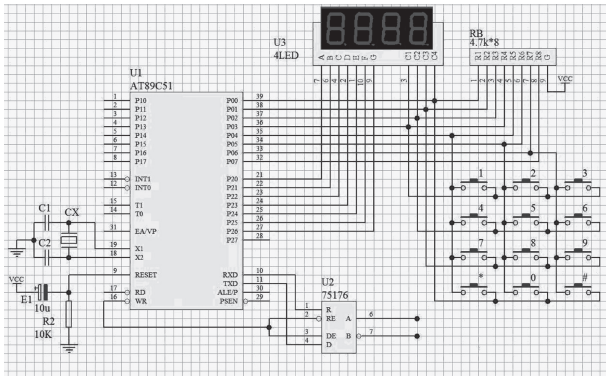

图 1 多任务编程实例硬件设计图

Fig. 1 Schematic for Multitask Programming Example

3.3 软件设计

3.3.1 编程说明

编程目的:

1) 从键盘键入 4 位数据,键入过程中可按 * 键清除重输。

2) 每键入 1 位数据。LED 屏从最右边 1 位开始显示,原各位显示向左移 1 位。

3) 键入数据过程中,如果有 5s 钟没有继续按键,系统清除输入数据和显示。

4) 当输入完 4 位数据后,系统将 4 位数据从串行口发送到数据总线。

5) 当从数据总线上接收到数据 #0D8H 时,开始发送 4 位数据。

6) 75176 芯片平时为接收状态,当接收到 #0D8H 时,如果系统有数据要发送,则变为发送状态,发送数据,当数据发送完毕时,75176 芯片恢复为接收状态。

根据上述编程目的将系统编程划分为以下 7 个任务:

1) 4 位 LED 数码管动态显示;

2) 键盘扫描检测;当有按键时,启动第 3 任务。

3) 处理键盘检测结果,设置防抖动延时;计算键值;并在释放键后启动第 4 任务;

4) 将键值转换为输入数据和显示数据,分别送输入数据缓存区和显示缓存区;当键入 * 时清除已输入数据和显示;当输入第 1 位数据时,启动第 5 任务;当输入数据缓冲区满时,启动第 6 任务的一个条件;

5) 5s 延时,5s 过后没有继续按键时,清除已输入数据和显示;

6) 串行口发送数据,启动本任务同时需满足有数据要发送和接收到 #0D8H 两个条件;

7) 串行口接收数据,当接收到 #0D8H 时,启动第 6 任务的另一条件。

其它说明:

1) 在上述 7 个任务中,LED 显示和键盘检测是实时任务,其余是常规任务。

2) 系统采用执行其它任务的时间处理 LED 动态显示和键盘扫描检测任务所需的延时。

3) 系统采用定时器 0 中断处理 5s 延时。定时标志 20H 置位间歇为 25ms;计数变量 6EH 初始化和重置值为 200(#0C8H)。

4) 系统采用串行口中断处理串行数据的接收和发送。串行口接收中断启动系统第 7 项任务。

3.3.2 系统流程图

图 2 是 7 个任务轮流执行的系统总流程图

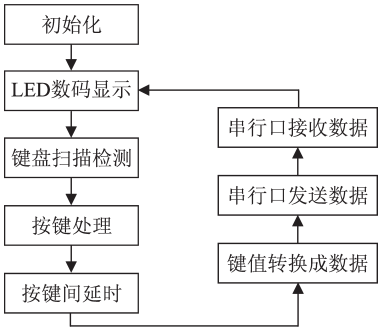

图 2　多任务编程实例系统流程图

Fig.2 System Flowchart of Multitask Programming Example

3.3.3 任务设计说明

1) 4 位 LED 数码管动态显示

在单任务设计中,4 位 LED 数码管显示是一次完成的[10],每显示 1 位,都要设计一定的延时,这会浪费 CPU 时间,影响其它任务的实时执行。多任务设计的方案是,每次执行显示任务只依次显示一位,然后就执行下一任务。系统利用执行其它任务的时间来为 LED 的每位显示延时,即在显示延时过程中,系统同时在执行其它任务。

系统设有一个显示缓冲区 7CH−7FH, LED 显示是通过自动扫描显示缓冲区进行的。

显示初始化:

```
MOV  30H, #3FH          ;0  30−39H 显示数字段表
MOV  31H, #06H          ;1
MOV  32H, #05BH         ;2
MOV  33H, #4FH          ;3
MOV  34H, #66H          ;4
MOV  35H, #6DH          ;5
MOV  36H, #7DH          ;6
MOV  37H, #07H          ;7
MOV  38H, #7FH          ;8
MOV  39H, #6FH          ;9
MOV   R0, #7FH          ; 启动显示位指针初始化
```

显示任务代码:

```
DISP: CJNE R0,    #7FH,  DISP1 ;DISP, LED 动态显示任务
MOV R3,  #0FEH          ; 初始化,先显示最左边位
```

```
        MOV R0,  #7CH      ;R0, 显示缓存指针初始化
        MOV 40H, #01H      ;40, 键盘扫描键值初始化
        AJMP DISP2
    DISP1:MOV A,   R3
        RL   A
        MOV R3,  A
        INC  R0
        INC  40H
    DISP2:MOV P2,  #00H   ; 熄灭显示
        MOV P0,  R3        ;R3 控制显示位
        MOV P2,@R0         ;7CH-7FH 4 位显示字缓存
    DISPE:NOP
```

2) 键盘扫描检测

矩阵键盘扫描检测的典型设计是依次置每行电平为低,对键盘行进行扫描,检测各列是否有键按下。由图 1 所示,矩阵键盘的行线和 LED 显示的位线是一致的,键盘扫描和动态显示扫描保持同步,很便于程序设计。所以,在多任务设计方案中,和 LED 显示任务一样,每次执行键盘检测任务时,只对一行进行检测。当没有按键按下时,系统依次对每行都进行检测;当有按键按下时,系统只对上次检测到按键按下的行进行检测,从而锁定对该按键的继续检测。

键盘扫描任务与 LED 显示任务通过共享全局变量 R0 保持同步:

```
GETK: JNB 00H, GETK0     ;GETK,按键扫描检测任务
    MOV  A,  R0
    CJNE  A, 41H, GETKE
GETK0:MOV  A,  P0   ; 键盘检测
    ANL  A,   #0F0H
    CJNE  A,  #0F0H, GETK1
    CLR  00H          ; 未检测到按键
    AJMP  GETKE
GETK1:SETB  00H       ; 已检测到按键
    MOV  42H, A       ;42H 缓存按键状态
    MOV  41H, R0      ;41H 记录检测到按键的行 , 以后锁定对这行的继续检测
    GETKE:NOP
```

3) 按键的处理

按键的处理主要包括计算按键持续按下的时间,键值计算和对按键释放后的处理。为防止按键抖动,持续按键需在一定的延时后,才能处理。多任务设计是在其任务中插入一个短延时过程。在处理完按键键值后,需等待释放按键。多任务设计通过设置消息(11H 置位) 屏蔽本任务和不启动后续相关任务的方式来设计等待过程:

```
KEYPR: JNB 00H, KEYPRD;KEYPR,处理按键任务
    JB 11H, KEYPRE          ;
    JB 10H, KEYPR1
    SETB  10H
    MOV 63H, #0FFH ; 第一步设置键盘防抖动延时时间
    MOV 64H, #04H
    AJMP KEYPRE
KEYPR1:DJNZ 63H,KEYPRE ; 第二步延时过程
    DJNZ 64H, KEYPRE
    SETB 11H          ; 第三步计算键值并等待释放键
    ANL 40H, #0FH ; 键值由 40H 的高 4 位和低 4 位组合。
KEYPR2:MOV A,42H; 根据 42H 缓存的按键状态进行计算
    JB  ACC.4,KEYPR3 ; 根据从 P0 口数据计算键盘值
    ORL  40H, #10H
    AJMP KEYPR5
KEYPR3:JB     ACC.5,KEYPR4
    ORL  40H, #20H
    AJMP  KEYPR5
KEYPR4:JB  ACC.6,KEYPRE
    ORL  40H, #30H
KEYPR5:MOV 43H, 40H ;43H 缓存已确定的键值
    AJMP KEYPRE
KEYPRD:CLR 10H      ; 按键释放,恢复到检测第一步
    JNB11H, KEYPRE
    CLR 11H          ; 如果已计算出键值
    SETB 01H         启动键值数据转换任务
KEYPRE:NOP
```

4) 键值转换和数据处理

键值转换的目的是将键值转换为输入数据和显示数据,分别存入数据缓冲区和显示缓冲区。当输入数据缓冲区满时,系统将缓冲区数据从串行口发送到总线上,当输入数据是 * 时,系统清除缓冲区数据和显示。为检测输入一位数据后,继续输入是否被放弃,每输入一位数据,系统启动或重置一个 5s 延时。R2 为输入数据缓冲区指针,初始值为 #50H:

```
LOADC: JNB 01H, LOADCE ;LOADC,键值转换成数据
    CLR 01H          ; 和转换成 LED 段码值任务
    MOV A, 43H   ; 根据 43H 缓存的键值进行数据转换
PRO11: CJNE A, #11H, PRO12; 按 * 建清除数据和显示
    AJMP LOADC3
PRO12:CJNE   A, #12H,     PRO13  ;"7"
    MOV      R1,#37H
    AJMP     LOADC1
PRO13:CJNE   A,#13H,      PRO14  ;"4"
    MOV      R1,#34H
    AJMP     LOADC1
PRO14:CJNE   A,#14H,      PRO15  ;"1"
    MOV      R1,#31H
```

```
    AJMP    LOADC1
PRO15:CJNE  A,#21H,     PRO16  ;"0"
    MOV     R1,#30H
    AJMP    LOADC1
PRO16:CJNE  A,#22H,     PRO17  ;"8"
    MOV     R1,#38H
    AJMP    LOADC1
PRO17:CJNE  A,#23H,     PRO18  ;"5"
    MOV     R1,#35H
    AJMP    LOADC1
PRO18:CJNE  A,#24H,     PRO19  ;"2"
    MOV     R1,#32H
    AJMP    LOADC1
PRO19:CJNE  A,#31H,     PRO1A  ;"#"
    AJMP    LOADCE
PRO1A:CJNE  A,#32H,     PRO1B  ;"9"
    MOV     R1,#39H
    AJMP    LOADC1
PRO1B:CJNE  A,#33H,     PRO1C  ;"6"
    MOV     R1,#36H
    AJMP    LOADC1
PRO1C: CJNE A,#34H, LOADC0A ;"3"
    MOV     R1,#33H
LOADC1: PUSH 00H
    MOV  00H, R2      ;R2 指定数据缓存地址
    MOV @R0,01H       ; 根据 R1 的值存入数据缓存区
    POP00H
    MOV     7FH,  7EH
    MOV     7EH,  7DH
    MOV     7DH,  7CH
    MOV     7CH,  @R1    ; 与 R1 值对应的段码送显示缓存区
LOADC1C:CJNE  R2,#53H,  LOADC2
LOADC1L:SETB  17H   ; 输入数据达 4 位，准备从串行口发送
        AJMP  LOADCE
LOADC2:INC    R2      ; 数据缓存区指针加 1
    SETB  02H         ; 启动或重置 5s 延时任务
    MOV  6EH,  #0C8H      ; 初始或重置计数器
    AJMP  LOADCE
LOADC3:MOV R2,#50H    ; 清除数据缓存区数据和显示
    MOV  7FH,#00H
    MOV      7EH,#00H
    MOV      7DH,#00H
    MOV      7CH,#00H
    CLR      02H          ; 中止 5s 按键延时任务
LOADCE:NOP
```

5) 按键间延时

系统通过按键连续输入数据，要求按键间间隙时间不能超过 5s，超过 5s 被认为是放弃输入，系统将清除已输入数据和显示。在本实例中，按键间延时设计为一个特殊任务。该任务的 5s 计数值在键值转换中启动或重置，可保证每次按键后都有 5s 时间延时。该延时可在 5s 后自动结束或被其它任务中止。

按键延时代码中，21H 在定时器 0 中断程序中置位，设计置位周期为 50ms，计算变量 6EH 在键值转换任务中设置或重置：

```
KYWAIT:JNB  02H, KYWAITE  ;KYWAIT,等待延时任务
    JNB     21H,  KYWAITE
    CLR     21H
    DJNZ    6EH,  KYWAITE
    CLR     02H          ; 结束 5s 延时
    MOV     R2,   #50H   ; 清除按键输入和显示
    MOV     7FH,  #00H
    MOV     7EH,  #00H
    MOV     7DH,  #00H
    MOV     7CH,  #00H
KYWAITE:NOP
```

6) 串行口发送 4 字节数据

串行口连续发送数据任务是配合串行口发送中断进行的。除第 2 节典型示例外，串行口发送数据任务还可按下面方式设计：

```
TXGL:JNB  03H, TXGLE      ;TXGL, 串行发送任务
    JNB  17H, TXGLD ;17H 置位，已有要发送的数据
    JB  30H, TXGLE   ; 等待上一帧数据发送完毕
    SETB  30H  ; 设置等待过程
    JB    3CH,  TXGL2
    JB    3BH,      TXGL1
    JB    3AH,      TXGL0
    SETB  P3.6 ; 将 75176 转换成发送状态
    MOV  SBUF,      50H    ; 发送第一字节
    SETB  3AH
    AJMP  TXGLE
TXGL0:MOV  SBUF,51H       ; 发送第二字节
    SETB  3BH
    AJMP  TXGLE
TXGL1: MOV SBUF,52H       ; 发送第三字节
    SETB  3CH
    AJMP  TXGLE
TXGL2:MOV  SBUF,53H       ; 发送第四字节
```

```
    SETB   48H          ;48H 置位，数据发送完毕后，让
75176 恢复接收状态
    CLR    3CH
    CLR    3BH
    CLR    3AH
    CLR    17H
TXGLD:CLR  03H      ；结束串行发送任务
TXGLE:NOP
```

7) 串行口接收数据

为了串行口中断不对系统的实时性产生较大影响，对串行口接收数据的处理是在中断外多任务队列中进行的。限于篇幅，系统只考虑从管理中心接收到发送数据允许同步信号#0D8H，开始向管理中心发送数据：

```
SHJJS: JB   04H,  SHJJSE   ;SHJJS,串行接收数据任务
    CLR    04H
    MOV    A,4FH
    CJNE   A,#0D8H, SHJJSE   ；接收到数据 08H
    SETB   03H      ;03H 置位，启动串行发送任务
SHJJSE:NOP
```

8) 串行中断程序

在串行发送数据中断中，系统设计了 75176 芯片转换为接收状态的过程，48H 在串行发送最后一字节时置位，以便在发送完数据后，改变系统状态。

在串行接收数据中断中，04H 置位，通知系统从串行口接收到一字节数据，并已存入 4FH，同时 04H 启动串行口接收数据任务：

```
SINP:  PUSH   PSW          ;SINP,串行口中断程序
       JNB    TI, SINP0
       CLR    30H
       CLR    TI
       JNB    48H,   SINP0
       CLR    48H
       CLR    P3.2
SINP0: JNB    RI, SINP3
       CLR    RI
       MOV    4FH,   SBUF
       SETB   04H
SINP3: POP    PSW
       RETI
```

3.3.4 系统的构成

将系统的 7 个任务经过上述设计，然后排成队列，并在队列最后设置一个长调转指令到队列中最前一个任务，构成一个依次执行并轮流循环的系统，这就是多任务编程的系统实例。在系统中，各任务需然是依次排成队列执行的，但任务的排序是可任意改变的。在任务间任意插入新的任务，也不会影响原系统正常运行。基于这一特征，各任务被看成是并发执行的。

4 结论

单片机多任务编程方法可归纳为：

1) 在单片机多任务编程中，各任务依次排成队列轮流执行。

2) 每次执行任务只调用其一个过程来执行，可保证各任务间最快速地切换。

3) 和时间片任务切换不同，切换任务不占用堆栈和额外的系统资源。

4) 各任务、过程间使用全局变量共享或交换数据，避免各种参数传递。

5) 基于消息(位寻址变量的置位)，每项任务，都可被其它任务启动和中止。

通过对循环延时和等待过程的重新设计，单片机多任务设计的每项任务和每个过程都是连续执行的。这可保证多任务间的切换最快，单片机的运行效率最高。因为任务的过程是完整执行的，所以和时间片强制切换不同，任务和过程的切换不占用堆栈和额外的系统资源。因为任何任务都可中止和屏蔽其它任务的执行，本方法也可设计抢占式任务。只要给实时任务也设计一个空过程，抢占式任务甚至还可屏蔽实时任务。达到最优先，最快执行的目的。本方法基于消息，但没有单独的消息循环过程[11-13]，可最大限度地减少系统开支。

本文介绍的多任务编程方法是用汇编语言实现的，但显而易见，根据本方法的原理，用 C51 语言同样也可设计单片机多任务系统。

在过去采用传统方法设计复杂的单片机系统过程中，人们容易发现系统交叉调用多，重复代码多，系统运行效率差，容易逻辑混乱且难以调试。例如要完成本文实例的设计，键盘处理过程需反复进行键盘检测。为了不让显示中断，键盘处理防抖延时和等待释放键盘时，又要多次调用显示过程，串行口连续发送数据过程中，更是频繁调用显示。显示、键盘检测、键盘处理等任务不能成为低耦合性的独立模块。系统程序的层次结构和可读性也都较差，并很难移植和重用。鉴于此，为探索一个结构清晰，易调试，任务明确且可重用、提高开发效率，无相互调用，无重复代码的系统，一个新的编程方式开始了实践和研究，并被总结为多任务编程方法。在近八年不断利用新方法设计(如楼宇对讲系统、联网报警系统、门禁系统，安防监控、广播控制、一卡通等) 智能系统产品的同时，本多任务编程方法取得了成熟和发展。与传统设计相比，这些系统产品硬件资源少，运行效率高。其硬件功能更多以软件取代，所以运行更稳定，且易维护，性价比高，取得了更高的经济效益。

参考文献

[1] 罗江，户永清 .51 单片机多任务机制的实现策略研究 [J]. 四川文

(下转第 31 页)

I_0=0.5, a=0.014,μ=0.2,τ=1。初始值在区间[-1,1]中随机取,对10个城市的TSP进行10000次仿真的结果见表2:

表2　MWCNN的仿真结果

β	NG	RG	NI
0.030	4654	93.08%	132
0.026	4689	93.78%	164
0.022	4693	93.86%	181
0.018	4712	94.24%	221
0.014	4743	94.86%	353
0.010	4767	95.34%	371
0.008	4775	95.50%	412
0.006	4781	95.62%	434
0.004	4796	95.92%	523
0.002	4812	96.24%	589

表1的符号和表2的意义相同。

从表1和表2的仿真结果对比看到,对于TCNN来说,在10000次仿真结果中找到最有解的比率较高,但是,我们提出的MWCNN最优解率更高,平均在93%以上,有的甚至达到96%以上;其次,在参数相同的情况下,MWCNN的平均迭代次数明显降低。因此,无论是在寻找全局最优解,跳出局部最小值上,还是在平均迭代次数上,都表现出TCNN无可比拟的优势。

从以上仿真结果可以看出,MWCNN具有更好的组合优化能力。

5　结论

本文在前人成果的基础上提出了墨西哥帽小波混沌神经元,此神经元的激励函数是Mexican Hat小波函数和Sigmoid复合函数,并引进了混沌特性,通过仿真发现此神经元既具有墨西哥帽小波的特性,也兼具了混沌以及神经元的特性,把三者优势巧妙的结合在一起,形成了一种全新的神经元。

基于此神经元建立了一个全新的墨西哥帽小波神经网络模型(MWCNN),通过仿真实验发现,MWCNN不仅具有混沌动力学特性,而且具有小波函数的正交性和函数逼近的唯一性。本文通过对参数u=0.2,0.6,1.0进行研究,验证其神经元的混沌性和收敛性的特性。并将MWCNN应用于TSP问题,与暂态神经网络相比,发现其有效性和函数逼近能力都有较大的提高,并取得较好的效果。

参考文献

[1] Chen L,Aihara K. chaotic simulated annealing by a neural network model with transient chaos[J]. Neural Networks,1995,8(6):915-930.

[2] Yamada T, Aihara K, Kotani M. Chaotic Neural Networks and the Travelling Salesman Problem[C], Proceedings of 1993 International Joint Conference on Neural Networks, 1993:1549-1552.

[3] Aihara K, Toyada M, Takabe T. Chaotic Neural Networks[J]. Physics Letters A,1990,144(6):333-340.

[4] 谭营,王保云,何振亚. 一种具有暂态混沌和时变增益的神经网络及其在优化计算中的应用[J],电子学报,1998,26(7):123-127.

[5] SHUAI J W, CHEN Z X, LIU R T. Self-evolution Neural Model[J]. Physics Letters A,1996,211(5):311-316

[6] POTAPOVE A,KALI M.Robust Chaose in Neural Networks [J]. Physics Letters A,2000,277(6):310-322.

[7] Zhang J, Walter G G, Miao Y B. Wavelet Neural Networks for function lernning[J]. IEEE Transaction on Signal Processing,1995,43(6):1485-1497.

[8] 侯媛彬,杜京义,汪　梅. 神经网络(M). 西安:西安电子科技大学出版社,2007.

[9] 徐耀群,孙明. Shannon小波混沌神经网络及其在TSP中的应用[C].Proceeding of the 25th Chinese ControlConference7-11August,2006,Haebin,Heilongjiang : 1172-1176.

[10] Hopfield, Tank D. Neural computation of decisions in optimization problems. Biology Cybernet-ics,1985 ,52(7): 141-152.

(上接第27页)

理学院学报(自然科学),2008,18 (2) :82-84.

[2] 阮元. 分时操作系统思想在单片机编程中的实现[OL].[2009-7-13].http://www.tech.ddvip.com.

[3] 王辉堂,颜志勇,陈文芗. 一种基于C51的多任务机制及应用[J]. 电子技术应用,2006,6(1):45-48.

[4] 李仕涌,谭南林. 多任务操作系统在嵌入式系统开发中的应用[J]. 北方交通大学学报,2002,26 (4) :79-82.

[5] 刘蕊. 基于实时操作系统RTX51的多任务处理设备设计[J]. 计算机光盘软件与应用,2010 9 (1) :25-26.

[6] 陈劲松,程新民,魏忠. 时间片轮转算法在单片机程序设计中的应用[J]. 电子技术应用,2003,29 (3) :79-82.

[7] 王江宁,吕军. 时分多线程在单片机系统中的应用研究[OL]. [2008-07-22].http://www.tech.ddvip.com.

[8] 潘永雄. 基于过程的单片机多任务程序结构及实现方法[J]. 电子工程师,2005,31 (3) .

[9] 胡全.51单片机的数码管动态显示技术[J]. 中国新技术新产品[J], 2009,13(1).

[10] 施隆照. 数码管显示驱动和键盘扫描控制器CH451及其应用[J]. 国外电子元器件,2004,1 (1) :53-57.

[11] 杨金,孙世新. 基于消息机制的单片机多任务系统的开发[J]. 计算机时代,2007,11 (1) .

[12] 周国柱,张道德,杨光友. 基于消息的单片机多任务编程[J]. 湖北工学院学报,2002,9 (1) .

[13] 于平. 从MCS51向AVR的认识转换[J]. 软件,2010,31(12): 48-50