手把手教你学单片机的C语言程序设计(十)

开关语句和循环语句

◆吕超亚

开关语句

switch/case 开关语句是一种多分支选择语句，是用来实现多方向条件分支的语句。虽然从理论上讲采用条件语句也可以实现多方向条件分支，但是当分支较多时会使条件语句的嵌套层次太多，程序冗长，可读性降低。开关语句可直接处理多分支选择，使程序结构清晰，使用方便。开关语句是用关键字 switch 构成的，它的一般形式如下：

```
switch(表达式)
{
 case 常量表达式 1:   {语句 1;} break;
 case 常量表达式 2:   {语句 2;} break;
                ⋮
 case 常量表达式 n:   {语句 n;}break;
 default:        {语句 d;}break;
}
```

开关语句的执行过程是：

1.当 switch 后面表达式的值与某一"case"后面的常量表达式的值相等时，就执行该"case"后面的语句，然后遇到 break 语句而退出 switch 语句。若所有"case"中常量表达式的值都没有与表达式的值相匹配，就执行 default 后面的 d 语句。

2.switch 后面括号内的表达式，可以是整型或字符型表达式，也可以是枚举类型数据。

3.每一个 case 常量表达式的值必须不同，否则就会出现自相矛盾的现象(对同一个值，有两种或者多种解决方案提供)。

4. 每个 case 和 default 的出现次序不影响执行结果，可先出现"default"再出现其它的"case"。

4.假如在 case 语句的最后没有加"break:"，则流程控制转移到下一个 case 继续执行。因此，在执行一个 case 分支后，使流程跳出 switch 结构，即终止 switch 语句的执行，可用一个 break 语句完成。

实验一

在 LED/128*64 图形液晶试验板上实现：

输入年份 year 和月份 month 后，程序计算出该月有多少天 day。S4 键用作状态设定(输入年、月或显示天数的切换)，S1 键用作输入年份的低 2 位(如输入 1986 年的 86)或输入月份，S2 键用作输入年份的高 2 位(如输入 1986 年的 19)。

该程序设计的关键是要判断当年是否为闰年。闰年的 2 月有 29 天，平年的 2 月只有 28 天。闰年的条件是：年份数 year 能被 4 整除，但不能被 100 整除；或者年份数 year 能被 400 整除。

其逻辑关系为:year%4==0&&year%100!=0||year%400==0

当此表达式的值为真时，year 为闰年，否则为平年。

在我的文档中建立一个文件目录(cs21)，然后建立 cs21.uv2 的工程项目，最后建立源程序文件(cs21.c)。

输入下面的程序：

```
#include <REG51.H>// 序号(以下同):1
#define uchar unsigned char   //2
#define uint unsigned int     //3
uchar code SEG7[10]={0x3f,0x06,0x5b,
0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};//4
uchar ACT[4]={0xef,0xdf,0xbf,0x7f}; //5
//=========================6
uchar status_flag;                  //7
uint year;                          //8
uchar month;                        //9
uchar day;                          //10
uchar temp_year_l,temp_month;//11
uchar temp_year_h;                  //12
//------------------------- 13
void key_s1(void);                  //14
void key_s2(void);                  //15
void key_s4(void);                  //16
void delay(uint k);                 //17
uchar conv(uint year,uchar month);//18
//-------------------------19
void main(void)                     //20
{   uchar i;                        //21
    uint temp1,temp2;               //22
    while(1)                        //23
    {   key_s4();                   //24
        switch(status_flag)         //25
        {                           //26
    case 0: key_s1();
            temp1=temp_year_l;//27
        key_s2 ();temp2=temp_year_h;
break;  //28
    case 1:key_s1();
         month=temp_month;break;//29
        default :break;             //30
        }                           //31
        year=temp1+(temp2*100);//32
//-------------------------33
        day=conv(year,month);//34
//-------------------------35
        for(i=0;i<40;i++)           //36
        {                           //37
        switch(status_flag)         //38
        {                           //39
        case 0:P0=SEG7 [year% 10];
P2=ACT[0];delay(1);//40
            P0=SEG7[(year%100)/10];
P2=ACT[1];delay(1);//41
            P0=SEG7[(year/100)%10];
P2=ACT[2];delay(1);//42
            P0=SEG7 [year/1000];
P2=ACT[3];delay(1);break;   //43
        case 1:P0=SEG7 [month% 10];
P2=ACT[0];delay(2);//44
            P0=SEG7 [month/10];
P2=ACT[1];delay(2);break;//45
        case 2:if (day){P0=SEG7[day%
```

```
10];P2=ACT[2];delay(2);//46
        P0=SEG7 [day/10];P2=ACT[3];
delay(2);}//47
    else{P0=0x00;P2=0xff;delay(2);//48
        P0=0x00;P2=0xff;delay    (2);}
break;    //49
    default :break;                  //50
        }                            //51
        }                            //52
    }                                //53
}                                    //54
//-------------------------55
void delay(uint k)                   //56
{                                    //57
uint i,j;                            //58
for(i=0;i<k;i++){                    //59
for(j=0;j<121;j++)                   //60
{;}}                                 //61
}                                    //62
//-------------------------63
void key_s1(void)                    //64
{                                    //65
    P2=0xff;                         //66
     if(P2==0xfe)                    //67
     {                               //68
      switch(status_flag)            //69
      {                              //70
      case 0:temp_year_l++;          //71
        if          (temp_year_l>99)
temp_year_l=0;     break;//72
      case 1:temp_month++;           //73
        if          (temp_month>12)
temp_month=1;break;//74
      default :break;                //75
      }                              //76
    }                                //77
}                                    //78
//-------------------------79
void key_s2(void)                    //80
{                                    //81
    P2=0xff;                         //82
     if(P2==0xfd)                    //83
     {                               //84
      switch(status_flag)            //85
      {                              //86
      case 0:temp_year_h++;          //87
          if        (temp_year_h>99)
temp_year_h=0;break;//88
      default :break;                //89
      }                              //90
    }                                //91
}                                    //92
//------------------------ 93
void key_s4(void)                    //94
{                                    //95
    P2=0xff;                         //96
    if(P2==0xf7)status_flag++;       //97
    if(status_flag>2)status_flag=0;//98
}                                    //99
//------------------------100
uchar conv(uint year,uchar month) //101
{uchar len;                          //102
        switch(month)                //103
        {                            //104
        case 1:len=31;break;         //105
        case 3:len=31;break;         //106
        case 5:len=31;break;         //107
        case 7:len=31;break;         //108
        case 8:len=31;break;         //109
        case 10:len=31;break;        //110
        case 12:len=31;break;        //111
        case 4:len=30;break;         //112
        case 6:len=30;break;         //113
        case 9:len=30;break;         //114
        case 11:len=30;break;        //115
        case 2:if (year%4==0&&year%
100!=0||year%400==0)len=29;//116
        else len=28;break;//117
        default:return 0;break; //118
        }                            //119
        return len;                  //120
}                                    //121
```

编译通过后，将生成的 cs21.hex 文件烧录到 89S51 芯片中，将芯片插入到 LED/128*64 图形液晶试验板上，试验板上接通 5V 电源，4 个数码管显示年份“0000”。按下 S1、S2 键可输入年份 year；按一下 S4 键切换到输入月份 month，按下 S1 键可输入月份 month；再按一下 S4 键切换到显示该月天数 day，若输入条件不符合（如输入了 13 月），则显示屏熄灭以示输入出错。

下面分析程序。

序号 1（程序解释，以下同）：包含头文件 REG51.H。

序号 2~3：数据类型的宏定义。

序号 4：数码管 0~9 的字形码。

序号 5：数码管的位选码。

序号 6：程序分隔。

序号 7：定义无符号字符型全局变量 status_flag（状态标志）。

序号 8：定义无符号整型全局变量 year（年）。

序号 9：定义无符号字符型全局变量 month（月）。

序号 10：定义无符号字符型全局变量 day（天）。

序号 11~12：定义无符号字符型全局变量 temp_year_l、temp_month、temp_year_h（程序执行过程中的中间变量）。

序号 13：程序分隔。

序号 14~18：函数声明。

序号 19：程序分隔。

序号 20：定义函数名为 main 的主函数。

序号 21：main 的主函数开始。定义无符号字符型局部变量 i。

序号 22：定义无符号整型局部变量 temp1、temp2。

序号 23：while 循环语句进行无限循环。

序号 24：调用 S4 键判断子函数 key_s4()。

序号 25：switch 语句，根据表达式 status_flag 的值进行散转。

序号 26：switch 语句开始。

序号 27~28：调用 key_s1()、key_s2()子函数，并把按键输入值赋 temp1、temp2。

序号 29：调用 key_s1()子函数，并把按键输入值赋 month。

序号 30：一项也不符合，则直接退出。

序号 31：switch 语句结束。

序号 32：数学计算。

序号 33：程序分隔。

序号 34：调用 conv(year,month)子函数得到当月的天数。

序号 35：程序分隔。

序号 36~52：for 语句循环，用于点亮刷新 4 个数码管。

序号 37：for 语句开始。

序号 38：switch 语句，根据表达式 status_flag 的值进行散转。

序号 39：switch 语句开始。

序号 40~43：点亮 4 位数码管，显示年份 year。

序号 44~45：点亮右边 2 位数码管，显示月份 month。

序号 46~47：如果 day 的值为真（大于 0），点亮左边 2 位数码管，显示天数 day。

序号 48~49：否则熄灭数码管。

序号 50：一项也不符合，则直接退出。

序号 51：switch 语句结束。

序号 52：for 语句结束。

序号 53：while 循环语句结束。

序号 54：main 的主函数结束。

序号 55：程序分隔。

序号 52~62：延时子函数。

序号 63:程序分隔。
序号 64~78:key_s1()子函数。
序号 65:key_s1()子函数开始。
序号 66:P2 口置全 1 以便读取按键输入。
序号 67:如果 P2 等于 0xfe,说明 s1 键按下。
序号 68:进入 if 条件语句。
序号 69~76:switch 语句。
序号 70:switch 语句开始。
序号 71:status_flag 为 0 时,temp_year_l 递增。
序号 72:temp_year_l 变化范围 0~99(年份的低 2 位变化范围 00~99)。
序号 73:status_flag 为 1 时,temp_month 递增。
序号 74:temp_month 变化范围 1~12(月份的变化范围 1~12)。
序号 75:一项也不符合,则直接退出。
序号 76:switch 语句结束。
序号 77:if 条件语句结束。
序号 78:key_s1()子函数结束。
序号 79:程序分隔。
序号 80~92:key_s2()子函数,可参考序号 64~78 对 key_s1()子函数的分析。
序号 93:程序分隔。
序号 94~99:key_s4()子函数。
序号 95:key_s4()子函数开始。
序号 96:P2 口置全 1 以便读取按键输入。
序号 97:如果 P2 等于 0xf7,说明 s4 键按下,status_flag 递增。
序号 98:status_flag 变化范围 0~2(只能选择输入年份、输入月份、显示天数 3 种状态)。
序号 99:key_s4()子函数结束。
序号 100:程序分隔。
序号 101~121:conv()子函数,通过输入年份、月份,计算出当月的天数。
序号 102:conv()子函数开始,定义无符号字符型局部变量 len。
序号 103~119:switch 语句,根据月份(month),得到天数(len)。
序号 104:switch 语句开始。
序号 105~111:1、3、5、7、8、10、12 月的天数为 31 天。
序号 112~115:4、6、9、11 月的天数为 30 天。
序号 116:2 月的天数如闰年为 29 天。
序号 117:否则是平年为 28 天。
序号 118:如月份出错(如输入了 13 个月),天数返回 0。
序号 119:switch 语句结束。
序号 120:如月份正确,返回该月的天数。
序号 121:conv()子函数结束。

循环语句

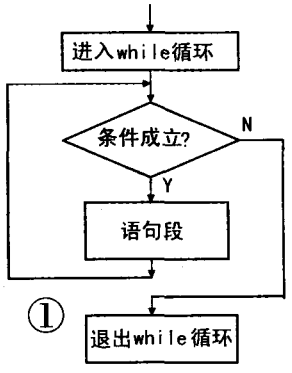
在许多实际问题中,需要程序进行有规律的重复执行,这时可以用循环语句来实现。在 C 语言中,用来实现循环的语句有 while 语句、do-while 语句、for 语句及 goto 语句等。

1.while 语句

while 语句构成循环结构的一般形式如下:

while(条件表达式) {语句;}

其执行过程是:当条件表达式的结果为真(非 0 值)时,程序就重复执行后面的语句,一直执行到条件表达式的结果变化为假(0 值)时为止。这种循环结构是先检查条件表达式所给出的条件,再根据检查的结果决定是否执行后面的语句。如果条件表达式的结果一开始就为假,则后面的语句一次也不会被执行。这里的语句可以是复合语句。图 1 为 while 语句的流程图。



实验二

用 while 语句求 1+2+…+100 的结果并将结果在 LED/16*2 字符液晶试验板上输出显示。

在我的文档中建立一个文件目录(cs22),然后建立 cs22.uv2 的工程项目,最后建立源程序文件(cs22.c)。

输入下面的程序:

```
#include <REG51.H>// 序号(以下同):1
#define uchar unsigned char          //2
#define uint unsigned int            //3
uchar code SEG7[10]={0xc0,0xf9,0xa4,
0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};//4
//=======================5
void delay(uint k)                   //6
{                                    //7
uint i,j;                            //8
for(i=0;i<k;i++){                    //9
for(j=0;j<121;j++)                   //10
{;}}                                 //11
}                                    //12
//=======================13
void main(void)                      //14
{                                    //15
uint s,n;                            //16
s=0;                                 //17
n=1;                                 //18
    while(n<=100)                    //19
    {                                //20
    s=s+n;                           //21
    n=n+1;                           //22
    }                                //23
//-------------------------24
    while(1)                         //25
    {                                //26
    P0=SEG7[s%10];delay(1);//27
    P1=SEG7[(s%100)/10];delay(1);//28
    P2=SEG7[(s/100)%10];delay(1);//29
    P3=SEG7[s/1000];delay(1);//30
    }                                //31
}                                    //32
```

编译通过后,将生成的 cs22.hex 文件烧录到 89S51 芯片中,将芯片插入到 LED/16*2 字符液晶试验板上,试验板上接通 9V 电源,4 个数码管显示等差数列之和 1+2+…+100 的结果“5050”。

下面分析程序。

序号 1(程序解释,以下同):包含头文件 REG51.H。
序号 2~3:数据类型的宏定义。
序号 4:数码管 0~9 的字形码。
序号 5:程序分隔。
序号 6~12:延时子函数。
序号 13:程序分隔。
序号 14:定义函数名为 main 的主函数。
序号 15:main 主函数开始。
序号 16:定义无符号整型变量 s、n。
序号 17:s 赋初值 0。
序号 18:n 赋初值 1。
序号 19~23:while 循环语句。
序号 20:while 循环语句开始。
序号 21~22:数学计算。
序号 23:while 循环语句结束。
序号 24:分隔。

序号 25:while 循环语句进行无限循环。
序号 26:while 循环语句开始。
序号 27:点亮数码管个位。
序号 28:点亮数码管十位。
序号 29:点亮数码管百位。
序号 30:点亮数码管千位。
序号 31:while 循环语句结束。
序号 32:main 主函数结束。

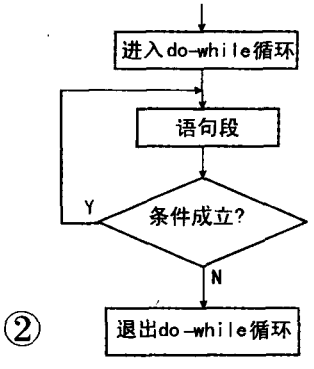
2.do-while 语句

do-while 语句构成循环结构的一般形式如下:

```
do
{语句;}
while(条件表达式);
```

其执行过程是:先执行给定的循环体语句,然后再检查条件表达式的结果。当条件表达式的值为真(非 0 值)时,则重复执行循环体语句,直到条件表达式的值变为假(0 值)时为止。因此,用 do-while 语句构成的循环结构在任何条件下,循环体语句至少会被执行一次。

对于同一个循环问题,可以用 while 语句处理,也可以用 do-while 结构处理。do-while 结构等价为一个语句加上一个 while 结构。do-while 结构适用于需要循环体语句执行至少一次以上的循环的情况。while 语句构成循环结构可以用于循环体语句一次也不执行的情况。图 2 为 do-while 语句的流程图。

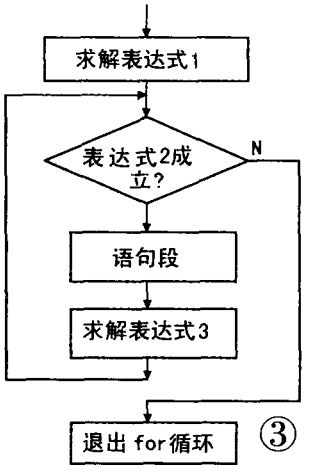


3.for 语句

采用 for 语句构成循环结构的一般形式如下:

for([初值设定表达式 1];[循环条件表达式 2];[更新表达式 3]) {语句;}

for 语句的执行过程是:先计算出初值表达式 1 的值作为循环控制变量的初值,再检查循环条件表达式 2 的结果,当满足循环条件时就执行循环体语句并计算更新表达式 3,然后再根据更新表达式 3 的计算结果来判断循环条件 2 是否满足……一直进行到循环条件表达式 2 的结果为假(0 值)时,退出循环体。图 3 为 for 语句的流程图。



在 C 语言程序的循环结构中,for 语句的使用最为灵活,它不仅可以用于循环次数已经确定的情形,而且可以用于循环次数不确定而只给出循环结束条件的情况。另外:for 语句中的 3 个表达式是相互独立的,并不一定要求 3 个表达式之间有依赖关系。并且 for 语句中的 3 个表达式都可能缺省,但无论缺省哪一个表达式,其中的两个分号都不能缺省。

例如,我们要把 50~100 之间的偶数取出相加,用 for 语句就显得十分方便。

实验三

用 for 语句求 50~100 之间的偶数之和并将结果在 LED/16*2 字符液晶试验板上输出显示。

在我的文档中建立一个文件目录(cs23),然后建立 cs23.uv2 的工程项目,最后建立源程序文件(cs23.c)。

输入下面的程序:

```
#include <REG51.H>// 序号(以下同):1
#define uchar unsigned char          //2
#define uint unsigned int            //3
uchar code SEG7[10]={0xc0,0xf9,0xa4,
0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};//4
//=======================5
void delay(uint k)                   //6
{                                    //7
uint i,j;                            //8
for(i=0;i<k;i++){                    //9
for(j=0;j<121;j++)                   //10
{;}}                                 //11
}                                    //12
//=======================13
void main(void)                      //14
{                                    //15
uint s,sum;                          //16
    for(s=50;s<=100;s=s+2) //17
    {sum=sum+s;}                     //18
//------------------------19
    while(1)                         //20
    {                                //21
    P0=SEG7[sum%10];delay(1);//22
 P1=SEG7[(sum%100)/10];delay(1);//23
 P2=SEG7[(sum/100)%10];delay(1);//24
    P3=SEG7[sum/1000];delay(1);//25
    }                                //26
}                                    //27
```

编译通过后,将生成的 cs23.hex 文件烧录到 89S51 芯片中,将芯片插入到 LED/16*2 字符液晶试验板上,试验板上接通 9V 电源,4 个数码管显示 50~100 之间的偶数之和的结果"1950"。

下面分析程序。

序号 1(程序解释,以下同):包含头文件 REG51.H。
序号 2~3:数据类型的宏定义。
序号 4:数码管 0~9 的字形码。
序号 5:程序分隔。
序号 6~12:延时子函数。
序号 13:程序分隔。
序号 6~12:延时子函数。
序号 13:程序分隔。
序号 14:定义函数名为 main 的主函数。
序号 15:main 主函数开始。
序号 16:定义无符号整型变量 s、sum。
序号 17~18:for 循环语句。
序号 18:数学计算。
序号 19:分隔。
序号 20~26:while 循环语句进行无限循环,用于点亮 4 个数码管。
序号 27:main 主函数结束。

4.goto 语句

goto 语句是一个无条件转向语句,它的一般形式如下:

```
goto 语句标号;
```

其中语句标号是一个带冒号":"的标识符,标识符标识语句的地址。当执行跳转语句时,使控制跳转到标识符指向的地址,从该语句继续执行程序。将goto语句和if语句一起使用,可以构成一个循环结构。但更常见的是在C语言程序中采用goto语句来跳出多重循环,需要注意的是只能用goto语句从内层循环跳到外层循环,而不允许从外层循环跳到内层循环。

5. break语句和continue语句

上面我们介绍的三种循环结构都是当循环条件不满足时,结束循环的。如果循环条件不止一个或者需要中途退出循环时,实现起来比较困难。此时可以考虑使用break语句或continue语句。

break语句

break语句除了可以用在switch语句中,还可以用在循环体中。在循环体中遇见break语句,立即结束循环,跳到循环体外,执行循环结构后面的语句。break语句的一般形式为:

break;

break语句只能跳出它所处的那一层循环,而不像goto语句可以直接从最内层循环中跳出来。由此可见,要退出多重循环时,采用goto语句比较方便。需要指出的是,break语句只能用于开关语句和循环语句之中,它是一种具有特殊功能的无条件转移语句。

continue语句

continue语句也是一种中断语句,它一般用在循环结构中,其功能是结束本次循环,即跳过循环体中下面尚未执行的语句,把程序流程转移到当前循环语句的下一个循环周期,并根据循环控制条件决定是否重复执行该循环体。continue语句的一般形式如下:

continue;

continue语句和break语句的区别在于:continue语句只结束本次循环而不是终止整个循环的执行;break语句则是结束整个循环,不再进行条件判断。

最后,我们来做break与continue语句的对比实验,感性认识一下它们的区别。

实验四

用for语句在LED/16*2字符液晶试验板上做一个0~9递增数值测试,当数值小于5时,用break语句结束循环。

在我的文档中建立一个文件目录(cs24),然后建立cs24.uv2的工程项目,最后建立源程序文件(cs24.c)。

输入下面的程序:

```
#include <REG51.H>// 序号(以下同):1
#define uchar unsigned char          //2
#define uint unsigned int            //3
uchar code SEG7[10]={0xc0,0xf9,0xa4,
0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};//4
//=========================5
void delay(uint k)                   //6
{                                    //7
uint i,j;                            //8
for(i=0;i<k;i++){                    //9
for(j=0;j<121;j++)                   //10
{;}}                                 //11
}                                    //12
//=========================13
void main(void)                      //14
{                                    //15
uchar cnt=0;                         //16
P0=SEG7[0];                          //17
delay(1000);                         //18
    for(cnt=0;cnt<10;cnt++)  //19
    {                                //20
    if(cnt<5)break;                  //21
    P0=SEG7[cnt];                    //22
    delay(1000);                     //23
    }                                //24
while(1);                            //25
}                                    //26
```

编译通过后,将生成的cs24.hex文件烧录到89S51芯片中,将芯片插入到LED/16*2字符液晶试验板上,试验板上接通9V电源,右边一个数码管显示"0"。

分析一下程序。

序号1(程序解释,以下同):包含头文件REG51.H。

序号2~3:数据类型的宏定义。

序号4:数码管0~9的字形码。

序号5:程序分隔。

序号6~12:延时子函数。

序号13:程序分隔。

序号6~12:延时子函数。

序号13:程序分隔。

序号14:定义函数名为main的主函数。

序号15:main主函数开始。

序号16:定义无符号字符型变量cnt并赋初值0。

序号17:右边一个数码管显示"0"。

序号18:延时1秒。

序号19~24:for循环语句。

序号20:for循环语句开始。

序号21:如果cnt小于5,用break语句退出for循环。

序号22:右边一个数码管显示cnt的值。

序号23:延时1秒。

序号24:for循环语句结束。

序号25:while语句构成的死循环,程序原地踏步,动态停机。

序号26:main主函数结束。

小结:由于在cnt小于5时,break语句已跳出for循环圈(结束循环),因此不能显示1~9。

实验五

用for语句在LED/16*2字符液晶试验板上做一个0~9递增数值测试,当数值小于5时,用continue语句结束本次循环(进入下一次循环)。

在我的文档中建立一个文件目录(cs25),然后建立cs25.uv2的工程项目,最后建立源程序文件(cs25.c)。

输入下面的程序:

```
#include <REG51.H>// 序号(以下同):1
#define uchar unsigned char          //2
#define uint unsigned int            //3
uchar code SEG7[10]={0xc0,0xf9,0xa4,
0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};//4
//=========================5
void delay(uint k)                   //6
{                                    //7
uint i,j;                            //8
for(i=0;i<k;i++){                    //9
for(j=0;j<121;j++)                   //10
{;}}                                 //11
}                                    //12
//=========================13
void main(void)                      //14
{                                    //15
uchar cnt=0;                         //16
P0=SEG7[0];                          //17
```

单片机系统的设计(二)

——从家用多功能定时控制器电路设计，谈单片机系统的硬件电路设计的方法

◆陈阳海

(4)语音电路、EEPROM、红外接收及输出驱动电路

语音芯片很多，本设计采用aP8842,所需语音信号已预先录入。图6是其引脚图,图7是其在CPU控制模式下的典型接线图,S1～S5选择30段语音,SBT为放音触发端。从图7可知,对19段语音的控制需占用单片机的6根I/O口线。关于aP8842的详细资料,请参考相关文章或上网查询。89S52单片机片内不含EEPROM,需要外扩。此处选用存储容量为512字节的串行EEPROM 24C04,它通过SCL和SDA与单片机进行通信,WP端为写保护,用单片机控制该端口,可减少误写,提高可靠性。可见控制24C04要占单片机的3根口线。红外接收采用一体化的红外接收头HD0038,其输出接至单片机,需要占用一个I/O口。输出驱动电路由三极管驱动电路和继电器等组成。继电器的触点通过接插件引出,以便控制相关电器。

(5)温度检测电路

实现温度检测的方法很多。这里我们采用负温热敏电阻Rt作温度传感器、用NE555组成的单稳电路作温度检测电路,当电容器C的容量固定时,其单稳脉冲的宽度与Rt的阻值成正比,单片机先输出单稳触发信号,然后检测单稳脉冲的宽度并查表,便可求得

表1

功能	占用I/O口 输入	输出	要求	对I/O口的限定
按键	4		普通I/O口	无
显示		16	普通I/O口	段选要求驱动能力
语音	5		普通I/O口	无
EEPROM	1	2	普通I/O口	无
红外接收	1		具有外中断功能	必须用INT0或INT1口
输出驱动		1	普通I/O口	无
温度检测	1	1	输入具有外中断功能	必须用INT0或INT1口
RS232口	1	1	I/O口第2功能	用P3.0、P3.1口
ISP编程	2	1	I/O口第2功能	用P1.5、P1.6、P1.7
可扩展功能	1		可对外部脉冲计数	必须用T0或T1口

```
delay(1000);                    //18
    for(cnt=0;cnt<10;cnt++) //19
    {                           //20
    if(cnt<5)continue;          //21
    P0=SEG7[cnt];               //22
    delay(1000);                //23
    }                           //24
while(1);                       //25
}                               //26
```

编译通过后，将生成的cs25.hex文件烧录到89S51芯片中，将芯片插入到LED/16*2字符液晶试验板上,试验板上接通9V电源,右边一个数码管开始显示"0",随后显示"5~9"。

下面分析一下程序。

序号1(程序解释，以下同)：包含头文件REG51.H。

序号2~3:数据类型的宏定义。

序号4:数码管0~9的字形码。

序号5:程序分隔。

序号6~12:延时子函数。

序号13:程序分隔。

序号6~12:延时子函数。

序号13:程序分隔。

序号14:定义函数名为main的主函数。

序号15:main主函数开始。

序号16：定义无符号字符型变量cnt并赋初值0。

序号17:右边一个数码管显示"0"。

序号18:延时1秒。

序号19~24:for循环语句。

序号20:for循环语句开始。

序号21:如果cnt小于5,用continue语句退出for循环。

序号22:右边一个数码管显示cnt的值。

序号23:延时1秒。

序号24:for循环语句结束。

序号25:while语句构成的死循环，程序原地踏步,动态停机。

序号26:main主函数结束。

小结：由于在cnt小于5时,continue语句跳出for语句的本次循环,进入下一次循环,因此不能显示1~4,但可显示5~9。