手把手教你学单片机的C语言程序设计(十二)

中断服务函数

◆吕超亚

C51编译器支持在C语言源程序中直接编写80C51单片机的中断服务函数程序。以前我们学习用汇编语言编写中断服务程序时,会对堆栈出栈的保护问题而觉得头痛。为了能够在C语言源程序中直接编写中断服务函数,C51编译器对函数的定义进行了扩展,增加了一个扩展关键字interrupt。关键字interrupt是函数定义时的一个选项,加上这个选项就可以将一个函数定义成中断服务函数。定义中断服务函数的一般形式为:

表1

n	中断源	中断向量8n+3
0	外部中断0	0003H
1	定时器/计数器0	000BH
2	外部中断1	0013H
3	定时器/计数器1	001BH
4	串行口	0023H

函数类型 函数名 (形式参数表) [interrupt n] [using n]

关键字interrupt后面的n是中断号,n的取值范围为0~31。编译器从8n+3处产生中断向量,具体的中断号n和中断向量取决于不同的单片机芯片。80C51单片机的常用中断源和中断向量如表1所示。

80C51系列单片机可以在内部RAM中使用4个不同的工作寄存器组,每个寄存器组中包含8个工作寄存器(R0~R7)。C51编译器扩展了一个关键字using,专门用来选择80C51单片机中不同的工作寄存器组。using后面的n是一个0~3的常整数,分别选中4个不同的工作寄存器组。在定义一个函数时using是一个选项,对于初学者,如果不用该选项,则由编译器选择一个寄存器组作绝对寄存器组访问。

关键字using对函数目标代码的影响如下:

在函数的入口处将当前工作寄存器组保护到堆栈中,指定的工作寄存器内容不会改变,函数返回之前将被保护的工作寄存器组从堆栈中恢复。

使用关键字using在函数中确定一个工作寄存器组时必须十分小心,要保证任何寄存器组的切换都只在控制的区域内发生,如果不做到这一点将产生不正确的函数结果。

另外,带using属性的函数,原则上不能返回bit类型的值。并且关键字using不允许用于外部函数,关键字interrupt也不允许用于外部函数,它对中断函数目标代码的影响如下:

在进入中断函数时,特殊功能寄存器ACC、B、DPH、DPL、PSW将被保存入栈。如果不使用寄存组切换,则将中断函数中所用到的全部工作寄存器都入栈。函数返回之前,所有的寄存器内容出栈。中断函数由80C51单片机指令RETI结束。

值得注意的是,编写80C51单片机中断函数时应严格遵循以下规则:

1.中断函数不能进行参数传递,如果中断函数中包含任何参数声明都将导致编译出错。

2.中断函数没有返回值,如果企图定义一个返回值将得到不正确的结果。因此最好在定义中断函数时将其定义为void类型,以明确说明没有返回值。

3. 在任何情况下都不能直接调用中断函数,否则会产生编译错误。因为中断函数的返回是由80C51单片机指令RETI完成的,RETI指令影响80C51单片机的硬件中断系统。

4.如果中断函数中用到浮点运算,必须保存浮点寄存器的状态,当没有其他程序执行浮点运算时可以不保存。

5. 如果在中断函数中调用了其他函数,则被调用函数所使用的寄存器组必须与中断函数相同。用户必须保证按要求使用相同的寄存器组,否则会产生不正确的结果。如果定义中断函数时没有使用using选项,则由编译器选择一个寄存器组作绝对寄存器组访问。

下面做一下有关中断的实验。

实验一

在LED/16*2字符液晶试验板上做一个键控计数的实验。采用中断方法实现,按动7#、8#、9#键中的任一个时,触发外中断0,实现计数或停止。

在我的文档中建立一个文件目录(cs29),然后建立cs29.uv2的工程项目,最后建立源程序文件(cs29.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*********************/
uint data cnt;                                 //6
bit bdata bitflag;                             //7
/*****************8*********************/
void init(void)                                //9
{ P0=0xff;P1=0xff;P2=0xff;P3=0x0f;//10
    bitflag=1;                                 //11
    EX0=1;                                     //12
    IT0=1;                                     //13
    EA=1;                                      //14
}                                              //15
/*******************16*********************/
void delay(uint k)                             //17
{                                              //18
uint data i,j;                                 //19
for(i=0;i<k;i++)                               //20
    {                                          //21
    for(j=0;j<121;j++){;}                      //22
    }                                          //23
}                                              //24
/*********************25*********************/
void main(void)                                //26
```

```
{     init();                          //27
      while(1)                         //28
      {                                //29
      if(bitflag)cnt++;                //30
      if(cnt>999)cnt=0;                //31
      P2=SEG7[cnt/100];                //32
      P1=SEG7[(cnt%100)/10];           //33
      P0=SEG7[cnt%10];                 //34
      delay(500);                      //35
      }                                //36
}                                      //37
/************************38*****************/
void extern_int0(void) interrupt 0 using 0
                                       //39
{                                      //40
bitflag=!bitflag;                      //41
}                                      //42
```

编译通过后，将生成的 cs29.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 3 个 LED 数码管从"000"开始进行加法计数。按动 7#、8#、9# 三个键中的一个时，计数暂停。再按动时，计数继续。

我们对程序进行分析。

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：定义无符号整型全局变量 cnt。
序号 7：定义全局位标志 bitflag。
序号 8：程序分隔。
序号 9：定义函数名为 init 的初始化子函数。
序号 10：init 子函数开始。P0~P3 口分别赋初值。
序号 11：bitflag 赋初值为 1。
序号 12：允许外中断 0。
序号 13：外中断 0 设为边沿触发。
序号 14：开 CPU 中断。
序号 15：init 函数结束。
序号 16：程序分隔。
序号 17~24：定义函数名为 delay 的延时子函数。
序号 25：程序分隔。
序号 26：定义函数名为 main 的主函数。
序号 27：main 主函数开始。调用 init 初始化子函数。
序号 28：无限循环。
序号 29：无限循环语句开始。
序号 30：如果位标志 bitflag 为 1，则变量 cnt 累加。
序号 31：变量 cnt 的范围为 0~999。
序号 32：显示计数值 cnt 的百位。
序号 33：显示计数值 cnt 的十位。
序号 34：显示计数值 cnt 的个位。
序号 35：延时 500ms。
序号 36：无限循环语句结束。
序号 37：main 主函数结束。
序号 38：程序分隔。
序号 39：定义函数名为 extern_int0 的外中断 0 服务函数，使用第一组寄存器。
序号 40：外中断 0 服务函数开始。
序号 41：这里的工作为取反位标志 bitflag。当然也可进行其它操作。
序号 42：外中断 0 服务函数结束。

实验二

在 LED/16*2 字符液晶试验板上做一个定时中断的实验。采用定时器 T0，定时长度设为 50ms，每一次定时溢出时引起定时器中断，在中断服务函数中对计时器 cnt 累加。每 1 秒中控制 LED 亮 0.1s、灭 0.9s。完成一个低功耗的路障灯工作。

在我的文档中建立一个文件目录（cs30），然后建立 cs30.uv2 的工程项目，最后建立源程序文件（cs30.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
/************************4*******************/
uchar data cnt;                        //5
/*******************6**********************/
sbit LAMP=P0^0;                        //7
/*****************8**********************/
void init(void)                        //9
{                                      //10
TMOD=0x01;                             //11
TH0=-(50000/256);                      //12
TL0=-(50000%256);                      //13
ET0=1;                                 //14
TR0=1;                                 //15
EA=1;                                  //16
}                                      //17
/*************************18*************/
void delay(uint k)                     //19
{                                      //20
uint data i,j;                         //21
      for(i=0;i<k;i++)                 //22
      {                                //23
      for(j=0;j<121;j++){;}            //24
      }                                //25
}                                      //26
/************************27****************/
void time0(void) interrupt 1           //28
{                                      //29
TH0=-(50000/256);                      //30
TL0=-(50000%256);                      //31
cnt++;                                 //32
if(cnt<=2)LAMP=0;                      //33
else LAMP=1;                           //34
if(cnt>=20)cnt=0;                      //35
}                                      //36
/*******************37*****************/
void main(void)                        //38
{                                      //39
      init();                          //40
      while(1)                         //41
      {                                //42
      delay(3000);                     //43
      }                                //44
}                                      //45
```

编译通过后，将生成的 cs30.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，右边的个位数码管的 a 段周期性地闪亮，亮的时间约为 0.1 秒，灭的时间约为 0.9 秒。在夜晚用于路障灯指示很合适，况且灯光不是连续点亮，耗电也较省。

对程序进行一下分析。

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：程序分隔。
序号 5：定义无符号整型全局变量 cnt。
序号 6：程序分隔。
序号 7：端口定义。
序号 8：程序分隔。
序号 9：定义函数名为 init 的初始化子函数。
序号 10：init 子函数开始。
序号 11：定时器 T0 方式 0。
序号 12~13：当晶振频率为 12MHz 时，定时初值为 50ms。实验板的晶振频率为 11.0592MHz，因此定时长度近似为 50ms。
序号 14：允许 T0 中断。
序号 15：启动 T0。
序号 16：开总中断。
序号 17：init 函数结束。

序号 18:程序分隔。
序号 19~26:定义函数名为 delay 的延时子函数。
序号 27:程序分隔。
序号 28:定义函数名为 timer0 的 T0 中断服务函数,使用默认的寄存器组。
序号 29:timer0 中断服务函数开始。
序号 30~31:重装 50ms 定时初值。
序号 32:计时器 cnt 递增。
序号 33:cnt 的值小于等于 2(因每 50ms 中断一次,因此对应时间为 0~0.1s),点亮灯。
序号 34:否则熄灭灯。
序号 35:cnt 的值最大到 20 (对应时间为 1s),然后又从 0 开始。
序号 36:T0 中断服务函数结束。
序号 37:程序分隔。
序号 38:定义函数名为 main 的主函数。
序号 39:main 主函数开始。
序号 40:调用 init 初始化子函数。
序号 41:无限循环。
序号 42:无限循环语句开始。
序号 43:调用延时 3s 子函数。实际上 CPU 还可做其它事情。
序号 44:无限循环语句结束。
序号 45:main 主函数结束。

实验三

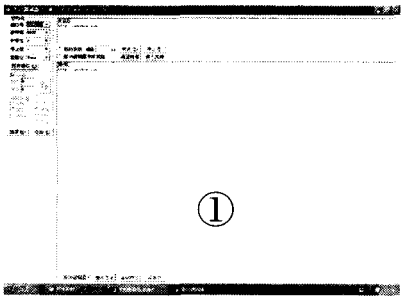
在 LED/128*64 图形液晶试验板上,进行单片机与 PC 机(个人电脑)的通信试验。实验过程为:PC 机发送一个字符给单片机,单片机收到后即在个、十位数码管上进行显示,同时将其回发给 PC 机。要求:单片机收到 PC 机发来的信号后用串口中断方式处理,而单片机回发给 PC 机时用查询方式。在我的文档中建立一个文件目录(cs31),然后建立 cs31.uv2 的工程项目,最后建立源程序文件(cs31.c)。

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 a;                             //7
//**************************8**************
void init(void)                      //9
{                                    //10
TMOD=0x20;                           //11
TH1=0xfd;                            //12
TL1=0xfd;                            //13
SCON=0x50;                           //14
TR1=1;                               //15
ES=1;                                //16
EA=1;                                //17
}                                    //18
//*****************************************19
void delay(uint k)                   //20
{                                    //21
uint data i,j;                       //22
    for(i=0;i<k;i++)                 //23
    {                                //24
    for(j=0;j<121;j++){;}            //25
    }                                //26
}                                    //27
//*****************************************28
void main(void)                      //29
{                                    //30
init();                              //31
  while(1)                           //32
  {                                  //33
    P0=SEG7[a/10];                   //34
    P2=ACT[1];                       //35
    delay(1);                        //36
    P0=SEG7[a%10];                   //37
    P2=ACT[0];                       //38
    delay(1);                        //39
        if(RI)                       //40
        {RI=0;                       //41
        SBUF=a;                      //42
        while(!TI);                  //43
        TI=0;                        //44
        EA=1;}                       //45
  }                                  //46
}                                    //47
//*****************************************48
void serial_serve(void) interrupt 4  //49
{                                    //50
a=SBUF;                              //51
EA=0;                                //52
}                                    //53
```

编译通过后,将生成的 cs31.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。在做实验时,我们需要在 PC 机进行信息发送。这个过程可以自己用 VB6.0 设计一个人机界面,也可以使用 Windows 自带的超级终端,还有一种方法是上网下载一个小巧的串口调试软件。这里笔者使用的是一个名叫 COMPort Debuger(串口调试器软件)的免安装共享软件,其下载地址为:http://emouze.com。

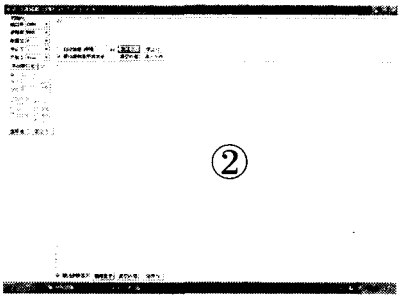
打开串口调试器软件,其界面如图 1 所示。右上方为发送区,右下方为接收区。左上方的初始化区域(如波特率、数据位等)不必更改。若你的 PC 机串口 COM1 已占用时,才可考虑改用COM2。



①

将 PC 机的串口与 LED/128*64 图形液晶试验板的串口连接好。发送与接收按 16 进制方式,然后打开串口。实验时,要先给 LED/128*64 图形液晶试验板上电(5V 稳压电源),然后才打开串口调试器软件,以免接收区乱码。

发送区输入"1",点发送,我们发现 LED/128*64 图形液晶试验板的右边两个数码管显示"01",同时接收区立即显示收到的"1"。发送区输入"A",点发送,我们发现试验板的右边两个数码管显示"10"(因为 16 进制的"A"等同于 10 进制的"10"),同时接收区立即显示收到的"A"。发送区再输入"20",点发送,我们发现 LED/128*64 图形液晶试验板的右边两个数码管显示 "32"(因为 16 进制的 "20" 等同于 10 进制的 "32"), 同时接收区立即显示收到的 "20",其界面如图 2 所示。



②

仪表放大器的设计与制作

★★

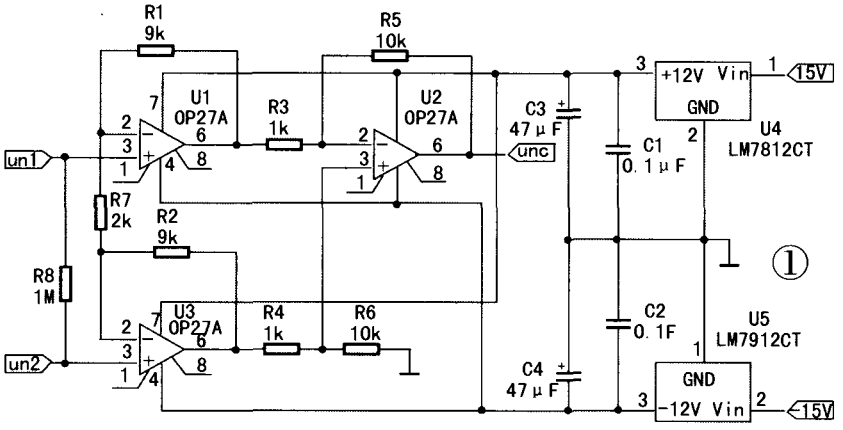
◆福建信息职业技术学院 陈 辉

本仪表放大器是由三个 OA27P 集成运算放大器组成，OA27P 的特点是低噪声、高速、低输入失调电压和卓越的共模抑制比。仪表放大器电路连接成比例运算电路形式，其中前两个运放组成第一级，二者都接成同相输入形式，因此具有很高的输入电阻。由于电路的结构对称，它们的漂移和失调都有互相抵消的作用。后一个运放组成差分放大器，将差分输入转换为单端输出。经计算，本设计中仪表放大器的电压放大倍数 $Au=\frac{R5}{R3}(1+\frac{2R1}{R2})=100$，结果将在仿真中验证。

仪表放大器的结构特点：使仪表放大器成为一种高输入电阻，高共模抑制比，具有较低的失调电压，失调电流、噪声及飘移的放大器。在使用时，在图 1 中 R4、R5、R6、R7 四个电阻要精密且匹配，否则将给放大倍数带来误差，而且将降低电路的共模抑制比。

二、仪表放大器电原理图

本设计采用 Protel99se 电路仿真软件，绘制电原理图后可自动生成印制



实验很成功，我们对程序进行详细分析。

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 4：0~9 数码管的字形码。

序号 5：4 位数码管的位选码。

序号 6：程序分隔。

序号 7：定义无符号字符型全局变量 a。

序号 8：程序分隔。

序号 9：定义函数名为 init 的初始化子函数。

序号 10：init 子函数开始。

序号 11：定时器 T1 方式 2。

序号 12~13：波特率 9600。

序号 14：串口方式 1，10 位可变波特率，允许接收。

序号 15：启动 T1。

序号 16：串口 1 开中断。

序号 17：开总中断。

序号 18：init 函数结束。

序号 19：程序分隔。

序号 20~27：定义函数名为 delay 的延时子函数。

序号 28：程序分隔。

序号 29：定义函数名为 main 的主函数。

序号 30：main 主函数开始。

序号 31：调用 init 初始化子函数。

序号 32：无限循环。

序号 33：无限循环语句开始。

序号 34~35：显示变量 a 的十位。

序号 36：延时 1ms。

序号 37~38：显示变量 a 的个位。

序号 39：延时 1ms。

序号 40：如果接收标志为 1，说明已收到信息。

序号 41：清除接收标志。

序号 42：将已经接收并存放在 a 的信息再送入 SBUF 发送出去。

序号 43：用查询的方法等待发送完毕。

序号 44：清除发送标志。

序号 45：重新打开总中断允许串口中断接收。

序号 46：无限循环语句结束。

序号 47：main 主函数结束。

序号 48：程序分隔。

序号 49：定义函数名为 serial_serve 的串口接收中断服务函数，使用默认的寄存器组。

序号 50：serial_serve 中断服务函数开始。

序号 51：将收到的信息存放在 a 中。

序号 52：关闭总中断，这样单片机发送时不使用中断方式。

序号 53：serial_serve 中断服务函数结束。