# 手把手教你学单片机的C语言程序设计(十一)

◆吕超亚

## 函数的定义

C语言程序是由函数构成的,函数是C语言中的一种基本模块。在《手教手教你学单片机的C语言程序设计(三)》中,我们已经介绍了C语言程序的组成结构,即C语言程序是由函数构成的,一个C源程序至少包括一个名为main()的函数(主函数),也可能包含其它函数。

C语言程序总是由主函数main()开始执行的,main()函数是一个控制程序流程的特殊函数,它是程序的起点。

所有函数在定义时是相互独立的,它们之间是平行关系,所以不能在一个函数内部定义另一个函数,即不能嵌套定义。函数之间可以互相调用,但不能调用主函数。

从使用者的角度来看,有两种函数:标准库函数和用户自定义功能子函数。标准库函数是编译器提供的,用户不必自己定义这些函数。C语言系统能够提供功能强大、资源丰富的标准函数库,作为使用者,在进行程序设计时应善于利用这些资源,以提高效率,节省开发时间。

函数定义的一般形式为:

函数类型标识符 函数名 (形式参数)

形式参数类型说明表列

{

局部变量定义

函数体语句

}

ANSIC标准允许在形式参数表中对形式参数的类型进行说明,因此也可这样定义:

函数类型标识符 函数名 (形式参数类型说明表列)

{

局部变量定义

函数体语句

}

其中:

"函数类型标识符"说明了函数返回值的类型,当"函数类型标识符"缺省时默认为整型。

"函数名"是程序设计人员自己定义的函数名字。

"形式参数类型说明表列"中列出的是在主调用函数与被调用函数之间传递数据的形式参数,如果定义的是无参函数,形式参数类型说明表列用void来注明。

"局部变量定义"是对在函数内部使用的局部变量进行定义。

"函数体语句"是为完成该函数的特定功能而设置的各种语句。

## 函数的参数和函数返回值

C语言采用函数之间的参数传递方式,便一个函数能对不同的变量进行处理,从而大大提高了函数的通用性与灵活性。在函数调用时,通过主调函数的实际参数与被调函数的形式参数之间进行数据传递来实现函数间参数的传递。在被调函数最后,通过return语句返回函数的返回值给主调函数。

return语句形式如下:

return (表达式);

对于不需要有返回值的函数,可以将该函数定义为"void"类型。void类型又称"空类型"。这样,编译器会保证在函数调用结束时不使函数返回任何值。为了使程序减少出错,保证函数的正确调用,凡是不要求有返回值的函数,都应将其定义成void类型。

在定义函数中指定的变量,当未出现函数调用的时候,它们并不占用内存中的存储单元。只有在发生函数调用的时候,函数的形参才被分配内存单元。在调用结束后,形参所占的内存单元也被释放。实参可以是常量、变量或表达式,要求实参必须有确定的值。在调用时将实参的值赋给形参变量(如果形参是数组名,则传递的是数组首地址而不是变量的值)。

从函数定义的形式看,又可划分为无参数函数、有参数函数、及空函数三种。

**1.无参数函数**

此种函数在被调用时无参数,主调函数并不将数据传送给被调用函数。无参数函数可以返回或不返回函数值,一般以不带返回值的为多。

**2.有参数函数**

调用此种函数时,在主调函数和被调函数之间有参数传递。也就是说,主调函数可以将数据传递给被调函数使用,被调函数中的数据也可以返回供主调函数使用。

**3. 空函数**

如果定义函数时只给出一对大括号{},不给出其局部变量和函数体语句(即函数体内部是"空"的),则该函数为"空函数"。这种空函数开始时只设计最基本的模块(空架子),其他作为扩充功能在以后需要时再加上,这样可使程序的结构清晰,可读性好,而且易于扩充。

## 函数的调用

C语言程序中函数是可以互相调用的。所谓函数调用就是在一个函数体中引用另外一个已经定义了的函数,前者称为主调用函数,后者称为被调用函数。主调用函数调用被调用函数的一般形式为:

函数名(实际参数表列)

其中,"函数名"指出被调用的函数。

"实际参数表列"中可以包含多个实际参数,各个参数之间用逗号隔开。实际参数的作用是将它的值传递给被

调用函数中的形式参数。需要注意的是，函数调用中的实际参数与函数定义中的形式参数必须在个数、类型及顺序上严格保持一致，以便将实际参数的值正确地传递给形式参数。否则在函数调用时会产生意想不到的错误结果。如果调用的是无参函数，则可以没有实际参数表列，但圆括号不能省略。

C 语言中可以采用三种方式完成函数的调用。

1.函数语句

在主调函数中将函数调用作为一条语句，例如：

fun1();

这是无参调用，它不要求被调函数返回一个确定的值，

2.函数表达式

只要求它完成一定的操作。

在主调函数中将函数调用作为一个运算对象直接出现在表达式中，这种表达式称为函数表达式。例如：

c=power(x,n)+power(y,m);

这其实是一个赋值语句，它包括两个函数调用，每个函数调用都有一个返回值，将两个返回值相加的结果，赋值给变量 c。因此这种函数调用方式要求被调函数返回一个确定的值。

3.函数参数

在主调函数中将函数调用作为另一个函数调用的实际参数。例如：

m=max(a,max(b,c));

max(b,c)是一次函数调用，它的返回值作为函数 max 另一次调用的实参。最后 m 的值为变量 a、b、c 三者中值最大者。

这种在调用一个函数的过程中又调用了另外一个函数的方式，称为嵌套函数调用。

## 对被调用函数的说明

在一个函数中调用另一个函数(即被调函数)，需要具备如下的条件：

1.被调用的函数必须是已经存在的函数(库函数或者用户自定义过的函数)。

2.如果程序使用了库函数，或者使用不在同一文件中的另外的自定义函数，则要程序的开头用 #include 预处理命令将调用有关函数时所需要的信息包含到本文中来。对于自定义函数，如果不是在本文件中定义的，那么在程序开始要用 extern 修饰符进行原型声明。使用库函数时，用 #include<***.h> 的形式，使用自己编辑的函数头文件等时，用 #include"***.h/c"的格式。

下面做几个实验。

## 实验一

在 LED/16*2 字符液晶试验板上实现参数传递的函数调用。数码管的低 2 位显示 3 和 8。3 号键按下后(即 P3.0 为低)调用交换子函数 swap，使得数码管的低 2 位交换显示为 8 和 3。

在我的文档中建立一个文件目录(cs26)，然后建立 cs26.uv2 的工程项目，最后建立源程序文件(cs26.c)。

输入下面的程序：

```
#include <REG51.H>// 序号(以下同):1
#define uchar unsigned char   //2
uchar code SEG7[10]={0xc0,0xf9,0xa4,
0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};//3
sbit P3_0=P3^0;                //4
uchar number1,number2;         //5
//=======================6
void swap(uchar x,uchar y);    //7
//=======================8
void main(void)                //9
{                              //10
uchar a,b;                     //11
number1=3;number2=8;           //12
a=number1; b=number2;          //13
    while(1)                   //14
    {                          //15
    P1=SEG7[number1];          //16
    P0=SEG7[number2];          //17
    P3=0x0f;                   //18
    if(!P3_0)swap(a,b);        //19
    }                          //20
}                              //21
//=======================22
void swap(uchar x,uchar y)     //23
{                              //24
number1=y;                     //25
number2=x;                     //26
}                              //27
```

编译通过后，将生成的 cs26.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 2 个 LED 数码管显示"38"。按下 3 号键右边 2 个 LED 数码管显示"83"。

我们分析一下程序。

序号 1(程序解释，以下同)：包含头文件 REG51.H。

序号 2：数据类型的宏定义。

序号 3：数码管 0~9 的字形码。

序号 4：定义 P3.0。

序号 5：定义两个无符号的字符型全局变量 number1、number2。

序号 6：程序分隔。

序号 7：子函数 swap 声明。

序号 8：程序分隔。

序号 9：定义函数名为 main 的主函数。

序号 10：main 主函数开始。

序号 11：定义无符号字符型局部变量 a、b。

序号 12：number1 赋值 3、number2 赋值 8。

序号 13：number1 值送 a、number2 赋值送 b。

序号 14：无限循环。

序号 15：无限循环语句开始。

序号 16：number1 的值查表送 P1 口显示。

序号 17：number2 的值查表送 P0 口显示。

序号 18：P3 口置 0x0f，以便读取 P3.0 的输入状态。

序号 19：如果 P3.0 为低电平，调用 swap 子函数，实现 number1、number2 两个数的交换。

序号 20：无限循环语句结束。

序号 21：main 主函数结束。

序号 22：程序分隔。

序号 23：定义函数名为 swap 的子函数。

序号 24：swap 子函数开始。

序号 25：y 传递的值送 number1。

序号 26：x 传递的值送 number2。

序号 27：swap 子函数结束。

## 实验二

在 LED/16*2 字符液晶试验板上实现三个数按大小自动排列。数码管的百、十、个位自动显示 a、b、c 三个数中的最大、中间、最小值。

在我的文档中建立一个文件目录(cs27)，然后建立 cs27.uv2 的工程项目，最后建立源程序文件(cs27.c)。

输入下面的程序：

```
#include <REG51.H>// 序号(以下同):1
#define uchar unsigned char    //2
```

```
uchar code SEG7[10]={0xc0,0xf9,0xa4,
0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};//3
//*********************************************4
void main(void)                          //5
{                                        //6
uchar a=3,b=1,c=9;                       //7
//*********************************************8
    if((a<b)&&(a<c))                     //9
    {P0=SEG7[a];                         //10
  if(b<c){P1=SEG7[b];P2=SEG7[c];}//11
    else {P1=SEG7[c];P2=SEG7[b];}//12
    }                                    //13
    //*********************************14
    if((b<a)&&(b<c))                     //15
    {P0=SEG7[b];                         //16
 if(a<c){P1=SEG7[a];P2=SEG7[c];}//17
     else {P1=SEG7[c];P2=SEG7[a];}//18
    }                                    //19
    //*********************************20
    if((c<a)&&(c<b))                     //21
    {P0=SEG7[c];                         //22
  if(a<b){P1=SEG7[a];P2=SEG7[b];}//23
    else {P1=SEG7[b];P2=SEG7[a];}//24
    }                                     //25
    //*********************************26
while(1);                                //27
}                                        //28
```

编译通过后，将生成的 cs27.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 数码管显示“931”。试着给 a、b、c 三个变量重新赋值(0~9)后，再编译、烧片，通电显示，则数码管总是会按大、中、小自动排列显示。

我们分析程序。

序号 1（程序解释，以下同）：包含头文件 REG51.H。

序号 2：数据类型的宏定义。

序号 3：数码管 0~9 的字形码。

序号 4：程序分隔。

序号 5：定义函数名为 main 的主函数。

序号 6：main 主函数开始。

序号 7：定义无符号字符型局部变量 a、b、c 并赋值。

序号 8：程序分隔。

序号 9：如果 a<b 并且同时 a<c，进入 if((a<b)&&(a<c))判断语句。

序号 10：a 的字形码送 P0 口显示。

序号 11：如果 b<c，b 的字形码送 P1 口显示，c 的字形码送 P2 口显示。

序号 12：否则 c 的字形码送 P1 口显示，b 的字形码送 P2 口显示。

序号 13：if((a<b)&&(a<c))判断语句结束。

序号 14：程序分隔。

序号 15：如果 b<a 并且同时 b<c，进入 if((b<a)&&(b<c))判断语句。

序号 16：b 的字形码送 P0 口显示。

序号 17：如果 a<c，a 的字形码送 P1 口显示，c 的字形码送 P2 口显示。

序号 18：否则 c 的字形码送 P1 口显示，a 的字形码送 P2 口显示。

序号 19：if((b<a)&&(b<c))判断语句结束。

序号 20：程序分隔。

序号 21：如果 c<a 并且同时 c<b，进入 if((c<a)&&(c<b))判断语句。

序号 22：c 的字形码送 P0 口显示。

序号 23：如果 a<b，a 的字形码送 P1 口显示，b 的字形码送 P2 口显示。

序号 24：否则 b 的字形码送 P1 口显示，a 的字形码送 P2 口显示。

序号 25：if((c<a)&&(c<b))判断语句结束。

序号 26：程序分隔。

序号 27：死循环，动态停机。

序号 28：main 主函数结束。

## 实验三

在 LED/128*64 图形液晶试验板上，设计一个能进行华氏 - 摄氏温度转换的仪器。用按键 S1、S2 输入华氏温度值，按下 S4 键后显示出对应的摄氏温度值。为了便于学习，采用整数输入及整数运算 / 显示，不牵涉到小数部分。华氏温度值的输入范围为 -500 度 ~999 度。

摄氏温度值 c 与华氏温度值 f 的换算公式为：

c=(f-32)*5/9

在我的文档中建立一个文件目录(cs28)，然后建立 cs28.uv2 的工程项目，最后建立源程序文件(cs28.c)。

输入下面的程序：

```
#include <REG51.H>// 序号(以下同)：1
#include <MATH.H>                    //2
#define uchar unsigned char          //3
#define uint unsigned int            //4
uchar code SEG7[10]={0x3f,0x06,0x5b,
0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};//5
uchar ACT[4]={0xef,0xdf,0xbf,0x7f}; //6
/*****************************7************/
int c,f;                             //8
int temp;                            //9
uchar status;                        //10
/*****************************11*******/
void key_s1(void);                   //12
void key_s2(void);                   //13
void key_s4(void);                   //14
int conv(int fin);                   //15
void delay_1ms(void);                //16
/*****************************17*******/
void main(void)                      //18
{int temp_f,temp_c;                  //19
while(1)                             //20
{                                    //21
 key_s1();                           //22
 key_s2();                           //23
 key_s4();                           //24
 if(status==0)                       //25
 {                                   //26
        if(f<0)                      //27
        {temp_f=abs(f);              //28
P0=SEG7[temp_f%10];P2=ACT[0];//29
        delay_1ms();                 //30
P0=SEG7 [(temp_f%100)/10];P2=ACT[1];
//31
       delay_1ms();                  //32
P0=SEG7 [(temp_f/100)%10];P2=ACT[2];
//33
        delay_1ms();                 //34
        P0=0x40;P2=ACT[3];           //35
         delay_1ms();                //36
         }                           //37
         else                        //38
         {                           //39
    P0=SEG7[f%10];P2=ACT[0];//40
        delay_1ms();                 //41
 P0=SEG7[(f%100)/10];P2=ACT[1];//42
        delay_1ms();                 //43
 P0=SEG7[(f/100)%10];P2=ACT[2];//44
        delay_1ms();                 //45
        }                            //46
 }                                   //47
 else                                //48
 {                                   //49
        c=conv(f);                   //50
    if(c<0)                          //51
        {temp_c=abs(c);              //52
P0=SEG7[temp_c%10];P2=ACT[0];//53
         delay_1ms();                //54
P0=SEG7[(temp_c%100)/10];P2=ACT[1];
//55
         delay_1ms();                //56
P0=SEG7[(temp_c/100)%10];P2=ACT[2];
//57
         delay_1ms();                //58
            P0=0x40;P2=ACT[3];//59
         delay_1ms();                //60
        }                            //61
        else                         //62
        {                            //63
```

```
    P0=SEG7[c%10];P2=ACT[0];//64
        delay_1ms();          //65
P0=SEG7[(c%100)/10];P2=ACT[1];//66
        delay_1ms();          //67
P0=SEG7[(c/100)%10];P2=ACT[2];//68
        delay_1ms();          //69
        }                     //70
 }                            //71
}                             //72
}                             //73
/*********************************74*******/
void key_s1(void)             //75
{                             //76
P2=0xff;                      //77
 if(P2==0xfe)                 //78
 {if(temp>50)temp=0;          //79
    if(temp==0)f++;           //80
    if(f>999)f=999;           //81
    temp++;                   //82
 }                            //83
}                             //84
/*******************************85*********/
    void key_s2(void)         //86
{                             //87
P2=0xff;                      //88
 if(P2==0xfd)                 //89
 {if(temp>50)temp=0;          //90
    if(temp==0)f--;           //91
    if(f<-500)f=-500;         //92
    temp++;                   //93
 }                            //94
}                             //95
/*******************************96*********/
void key_s4(void)             //97
{                             //98
P2=0xff;                      //99
    if(P2==0xf7)status=1;     //100
    else status=0;            //101
}                             //102
/*******************************103*******/
int conv(int fin)             //104
{int ddata;                   //105
ddata=fin-32;                 //106
ddata=(ddata*5)/9;            //107
return ddata;                 //108
}                             //109
/*******************************110*******/
void delay_1ms(void)          //111
{                             //112
uint k;                       //113
for(k=0;k<121;k++);           //114
}                             //115
```

编译通过后，将生成的 cs28.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 稳压电源，右边 3 个 LED 数码管显示华氏温度值“000”。按下 S1 键，华氏温度值升高（最高为 999）；按下 S2 键，华氏温度值降低（最低为 -500）。按下 S4 键后，数码管即显示对应的摄氏温度值。是不是很有趣？也很实用？

我们对程序进行分析详解。

序号 1（程序解释，以下同）：包含头文件 REG51.H。

序号 2：包含头文件 MATH.H，该头文件是数学计算（如计算绝对值）所必需的。

序号 3~4：数据类型的宏定义。

序号 5：数码管 0~9 的字形码。

序号 6：4 个数码管的位选码。

序号 7：程序分隔。

序号 8：定义整型全局变量 c、f。c 为摄氏温度值，f 为华氏温度值。

序号 9：中间变量 temp。

序号 10：状态标志变量 status。

序号 11：程序分隔。

序号 12~16：函数声明。

序号 17：程序分隔。

序号 18：定义函数名为 main 的主函数。

序号 19：main 主函数开始。定义整型局部变量 temp_f、temp_c。

序号 20：无限循环。

序号 21：无限循环语句开始。

序号 22：调用扫描 S1 键子函数，使输入的华氏温度值升高。

序号 23：调用扫描 S2 键子函数，使输入的华氏温度值降低。

序号 24：调用扫描 S4 键子函数，以决定显示摄氏温度（status 为 1）还是华氏温度（status 为 0）。

序号 25：如果 status 为 0。

序号 26：进入 if(status==0)语句。

序号 27：如果华氏温度 f<0。

序号 28：进入 if(f<0)语句。取 f 的绝对值后送 temp_f。

序号 29：显示华氏温度的个位。

序号 30：延时 1ms。

序号 31：显示华氏温度的十位。

序号 32：延时 1ms。

序号 33：显示华氏温度的百位。

序号 34：延时 1ms。

序号 35：千位显示负号。

序号 36：延时 1ms。

序号 37：if(f<0) 语句结束。

序号 38：否则如果华氏温度 f>=0。

序号 39：进入 if(f<0) 的否则语句。

序号 40：显示华氏温度的个位。

序号 41：延时 1ms。

序号 42：显示华氏温度的十位。

序号 43：延时 1ms。

序号 44：显示华氏温度的百位。

序号 45：延时 1ms。

序号 46：if(f<0) 的否则语句结束。

序号 47：if(status==0)语句结束。

序号 48：进入 if(status==0)的否则语句。

序号 49：if(status==0)的否则语句开始。

序号 50：调用 conv 子函数，将华氏温度值转换为摄氏温度值。

序号 51：如转换出的摄氏温度值 c<0。

序号 52：进入 if(c<0) 语句。取 c 的绝对值后送 temp_c。

序号 53：显示摄氏温度的个位。

序号 54：延时 1ms。

序号 55：显示摄氏温度的十位。

序号 56：延时 1ms。

序号 57：显示摄氏温度的百位。

序号 58：延时 1ms。

序号 59：千位显示负号。

序号 60：延时 1ms。

序号 61：if(c<0) 语句结束。

序号 62：否则如果摄氏温度 f>=0。

序号 63：进入 if(c<0) 的否则语句。

序号 64：显示摄氏温度的个位。

序号 65：延时 1ms。

序号 66：显示摄氏温度的十位。

序号 67：延时 1ms。

序号 68：显示摄氏温度的百位。

序号 69：延时 1ms。

序号 70：if(c<0) 的否则语句结束。

序号 71：if(status==0)的否则语句结束。

序号 72：无限循环语句结束。

序号 73：main 主函数结束。

序号 74：程序分隔。

序号 75：定义函数名为 key_s1 的 S1 键扫描子函数。

序号 76：key_s1 子函数开始。

序号 77：P2 口置全高，准备读取输入状态。

序号 78：若 P2 为 0xfe，说明 S1 键被按下，进入 if(P2==0xfe)语句。

序号 79：temp 的计数值范围在 0~50 之间。

序号 80：在 temp 为 0 时改变（增大）华氏温度值 f，使之与人眼的视觉相符。

序号 81：华氏温度值的最大输入值为 999。

序号 82：temp 计数值递增。

序号 83：if(P2==0xfe)语句结束。

序号 84：key_s1 子函数结束。

序号 85：程序分隔。

序号 86：定义函数名为 key_s2 的 S2 键扫描子函数。

序号 87：key_s2 子函数开始。

序号 88：P2 口置全高，准备读取输入状态。

序号 89：若 P2 为 0xfd，说明 S2 键被按下，进入 if(P2==0xfd)语句。

序号 90：temp 的计数值范围在 0~50 之间。

# 用增强型51实验板控制步进电机

◆徐 玮

今天，我们为大家讲解一下步进电机的基本原理以及其使用方法。

步进电机是一种将电脉冲转化为角位移的执行机构。在没有超出负载的情况下，步进电机的转动速度、停止的位置只取决于送给电机脉冲信号的频率和脉冲数，而不会受到负载变化的影响，如：我们给步进电机加一个脉冲信号，电机则转过一个步距角。

在初中物理课上，我们就已经接触到过电动机的原理，现在我们所接触到的常规交流或直流电机，都是加上相应的电压后，电机开始转动，断电后电机则停止转动。但对于步进电机，我们既能控制它的转动方向，又能控制它的转动速度，如：我们想让它顺时针转2圈，逆时针转3圈，或者是先正转4圈再反转5圈。由此看来，步进电机的动作方式比常规电机显得更为灵活、方便，多用性。因此它已经涉及到了机械、电子及计算机等众多相关行业，如：打印机，绘图仪，机器人等应用。

我们所使用的是12V的步进电机，为了方便演示及方便我们学习，我们直接将步进电机插到实验板的专用接口上，通过5V电源来供电。增强型51实验板采用ULN2003步进电机驱动电路，驱动端口为p0.0，p0.1，p0.2，p0.3。

上面我们曾讲到步进电机是通过脉冲量和频率来控制电机的运行状况的，因此，我们只要通过控制单片机的p0.0，p0.1，p0.2，p0.3这几个端口的高低电平，就可以指定步进电机的转动方向及转动速度。那么，该如何来控制这些端口的电平呢？其实很简单，用表1表2就可非常直观地反映步进电机正、反转的控制时序。

如果要使步进电机正转，应该依次给单片机P0口送出如表1所示的控制字，即分别向P0口输出0xfe，0xfd，0xfb，0xf7四个值即可。

如果要使步进电机反转，应该依次给单片机P0口送出如表2所示的控制字，即分别向P0口输出0xf7，0xfb，0xfd，0xfe这四个值即可。

表1 步进电机正转时序表

| | 步数 | P0.3 | P0.2 | P0.1 | P0.0 |
|---|---|---|---|---|---|
| 0xfe | 1 | 1 | 1 | 1 | 0 |
| 0xfd | 2 | 1 | 1 | 0 | 1 |
| 0xfb | 3 | 1 | 0 | 1 | 1 |
| 0xf7 | 4 | 0 | 1 | 1 | 1 |

表2 步进电机反转时序表

| | 步数 | P0.3 | P0.2 | P0.1 | P0.0 |
|---|---|---|---|---|---|
| 0xf7 | 1 | 0 | 1 | 1 | 1 |
| 0xfb | 2 | 1 | 0 | 1 | 1 |
| 0xfd | 3 | 1 | 1 | 0 | 1 |
| 0xfe | 4 | 1 | 1 | 1 | 0 |

下面我们动手做一下步进电机的实验。

首先，将51微型仿真器和步进电机插到增强型51实验板的相应端口上，同时插上外接电源变压器。注意：步进电机一共有4条线，但我们与实验板相连的白色插口上共有6个孔，在插口的最左边空出了一个位置，用于标记插口方向，如图1所示。

然后安装仿真及编程软件KEIL（位于配套光盘“仿真软件KEIL”目录

序号91：在temp为0时改变（降低）华氏温度值f，使之与人眼的视觉相符。
序号92：华氏温度值的最小输入值为-500。
序号93：temp计数值递增。
序号94：if(P2==0xfd)语句结束。
序号95：key_s2子函数结束。
序号96：程序分隔。
序号97：定义函数名为key_s4的S4键扫描子函数。
序号98：key_s4子函数开始。
序号99：P2口置全高，准备读取输入状态。
序号100：若P2为0xf7，说明S4键被按下，状态标志status置1。
序号101：否则状态标志status置0。
序号102：key_s4子函数结束。
序号103：程序分隔。
序号104：定义函数名为conv子函数，将华氏温度值转换为摄氏温度值。
序号105：conv子函数开始，定义整型局部变量ddata。
序号106~107：数学计算。
序号108：返回计算的结果。
序号109：conv子函数结束。
序号110：程序分隔。
序号111~115：1ms延时子函数。由于我们只需固定的1ms延时，故该子函数并没有参数传递，也无返回值。