手把手教你学单片机的C语言程序设计(五)

◆周兴华

常量

常量是在程序执行过程中其值不能改变的量。常量的数据类型有整型、浮点型、字符型和字符串型等,C51编译器还扩充了一种位(bit)标量。

实验一

在LED/16*2字符液晶试验板上实现乘法运算:两个乘数分别为常量与变量,其积在数码管上显示(最大显示到50)。

在我的文档中建立一个文件目录(cs7),然后建立cs7.uv2的工程项目,最后建立源程序文件(cs7.c)。

输入下面的程序:

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

编译通过后,将生成的cs7.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。发现右边2个LED数码管从"02"开始显示偶数,即"02"、"04"…,显示到"50"后不变。

对程序分析。

序号1(程序解释,以下同):包含头文件REG51.H。

序号2:数码管0~9的字形码。

序号3:定义CONST为常量2。该行的第一个非空白字符为#,表示该行是预处理器的伪指令语句行,它虽然处在源程序中,但并不产生程序代码,而是通知预处理器如何操作。这里的代用就是用CONST代替2。

序号4:程序分隔。

序号5~11:定义函数名为delay的延时子函数。

序号12:程序分隔。

序号13:定义函数名为main的主函数。

序号14:main的主函数开始。

序号15:定义无符号字符型变量x并赋初值1。定义无符号字符型变量y。

序号16:while循环语句,这里进行无限循环。

序号17:while循环语句开始。

序号18:将变量x与常量CONST相乘,其积放y中。

序号19:取出y的十位数送P1口显示。

序号20:取出y的个位数送P0口显示。

序号21:延时1秒以便观察清楚。

序号22:变量x加1以便下一次操作。

序号23:如果x等于26,则在此踏步(动态停机)。

序号24:while循环语句结束。

序号25:main的主函数结束。

变量及存储器类型

变量是一种在程序执行过程中其值可以变化的量。C语言程序中的每一个变量都必须有一个标识符作为它的变量名。在使用一个变量之前,必须先对该变量进行定义,指出它的数据类型和存储器类型,以便编译系统为它分配相应的存储单元。在C51中对变量进行定义的格式如下:

[存储种类]数据类型 [存储器类型]变量名表

如:auto int data x;

其中,"存储种类"和"存储器类型"是可选项。变量的存储种类有四种:自动(auto)、外部(extern)、静态(static)和寄存器(register)。在定义一个变量时如果省略存储种类选项,则该变量将为自动(auto)变量。

定义一个变量时除了需要说明其数据类型之外,Keil C51编译器还允许说明变量的存储器类型。Keil C5l编译器完全支持8051系列单片机的硬件结构,可以访问其硬件系统的所有部分。对于每个变量可以准确地赋予其存储器类型,从而可使之能够在单片机系统内准确地定位。

表1列出了Keil C51编译器所能识别的存储器类型

存储器类型	说明
data	直接访问内部数据存储器(128字节),访问速度最快
bdata	可位寻址内部数据存储器(16字节),允许位与字节混合访问
idata	间接访问内部数据存储器(256字节),允许访问全部内部地址
pdata	分页访问外部数据存储器(256字节),用MOVX @Ri指令访问
xdata	外部数据存储器(64K字节),用MOVX @DPTR指令访问
code	程序存储器(64K字节),用MOVC @A+DPTR指令访问

定义变量时如果省略“存储器类型”选项，则按编译模式 SMALL、COMPACT 或 LARGE 所规定的默认存储器类型确定变量的存储区域，不能位于寄存器中的参数传递变量和过程变量也保存在默认的存储器区域内。C51 编译器的 3 种存储器模式（默认的存储器类型）对变量的影响如下：

1.SMALL 变量被定义在 8051 单片机的内部数据存储器(data 区)中，因此对这种变量的访问速度最快。另外，所有的对象，包括堆栈，都必须嵌入内部数据存储器，而堆栈的长度是很重要的，实际栈长取决于不同函数的嵌套深度。

2.COMPACT 变量被定义在分页外部数据存储器(pdata 区)中，外部数据段的长度可达 256 字节。这时对变量的访问是通过寄存器间接寻址(MOVX @Ri)进行的，堆栈位于 8051 单片机内部数据存储器中。采用这种编译模式时，变量的高 8 位地址由 P2 口确定。因此，在采用这种模式的同时，必须适当改变启动程序 STARTUP.A51 中的参数：PDATASTART 和 PDATALEN；用 L5l 进行连接时还必须采用连接控制命令 PDATA 来对 P2 口地址进行定位，这样才能确保 P2 口为所需要的高 8 位地址。

3.LARGE 变量被定义在外部数据存储器(xdata 区，最大可达 64K 字节)中，使用数据指针 DPTR 来间接访问变量。这种访问数据的方法效率是不高的，尤其是对于 2 个或多个字节的变量，用这种数据访问方法对程序的代码长度影响非常大。另外一个不方便之处是这种数据指针不能对称操作。

8051 系列单片机具有 21 个特殊功能寄存器，它们离散分布在片内 RAM 的高 128 字节中。如定时器方式控制寄存器 TMOD、中断允许控制寄存器 IE 等。为了能够直接访问这些特殊功能寄存器，C51 编译器扩充了关键字 sfr 和 sfrl6，利用这种扩充关键字可以在 C 语言源程序中直接对 8051 单片机的特殊功能寄存器进行定义。定义方法如下：

sfr 特殊功能寄存器名 = 地址常数;

例如：sfr TMOD=0x89;// 定义定时 / 计数器方式控制寄存器，其地址为 89H。

这里需要注意的是，在关键字 sfr 后面必须是一个名字，名字可任意选取，但应符合一般习惯。等号后面必须是常数，不允许有带运算符的表达式，而且该常数必须在特殊功能寄存器的地址范围之内(80H~0FFH)。

在新一代的 8051 单片机中，特殊功能寄存器经常组合成 16 位来使用。为了有效地访问这种 16 位的特殊功能寄存器，可采用关键字 sfrl6，例如对 8052 单片机的定时器 T2，可采用如下的方法来定义：

sfr16 T2=0xCC;// 定义 T2，其地址为 T2L=0CCH，T2H=0CDH。

这里 T2 为特殊功能寄存器名，等号后面是它的低字节地址，其高字节地址必须在物理上直接位于低字节之后。这种定义方法适用于所有新一代的 8051 增强型单片机中新增加的特殊功能寄存器的定义。

在 8051 单片机应用系统中经常需要访问特殊功能寄存器中的某些位，C51 编译器为此提供了一种扩充关键字 sbit，利用它可以访问可位寻址对象。使用方法有如下 3 种：

1.sbit 位变量名 = 位地址

这种方法将位的绝对地址赋给位变量，位地址必须位于 80H~0FFH 之间。例如：

sbit OV=0xD2;

sbit CY=0xD7;

2.sbit 位变量名 = 特殊功能寄存器名 ^ 位位置

当可寻址位位于特殊功能寄存器中时可采用这种方法，“位位置”是一个 0~7 之间的常数。例如：

sbit OV=PSW^2;

sbit CY=PSW^7;

3.sbit 位变量名 = 字节地址 ^ 位位置

这种方法以一个常数（字节地址）作为基址，该常数必须在 80H~0FFH 之间。“位位置”是一个 0~7 之间的常数。例如：

sbit OV=0xD0^2;

sbit CY=0xD0^7;

当位对象位于 8051 单片机内部存储器的可位寻址区 bdata 时称之为“可位寻址对象”。C51 编译时会将对象放入 8051 单片机内部可位寻址区。例如：

int bdata my_x=12345;

使用关键字可以独立访问可位寻址对象中的某一位。例如：

sbit my_bit0=my_x^0;

sbit my_bit15=my_x^15;

操作符后面的位位置的最大值(即“^”后面的值)取决于指定的基址类型，对于 char 来说是 0~7；对于 int 来说是 0~15；对于 long 来说是 0~31。

从变量的作用范围来看，有全局变量和局部变量之分。

全局变量是指在程序开始处或各个功能函数的外面所定义的变量，在程序开始处定义的全局变量在整个程序中有效，可供程序中所有的函数共同使用，而在各功能函数外面定义的全局变量只对从定义处开始往后的各个函数有效，只有从定义处往后的各个功能函数可以使用该变量，定义处前面的函数则不能使用它。

局部变量是指在函数内部或以花括号{}围起来的功能块内部所定义的变量，局部变量只在定义它的函数或功能块以内有效，在该函数或功能块以外则不能使用它。因此局部变量可以与全局变量同名，但在这种情况下局部变量的优先级较高，而同名的全局变量在该功能块内被暂时屏蔽。

从变量的存在时间来看又可分为静态存储变量和动态存储变量。

静态存储变量是指在程序运行期间其存储空间固定不变的变量，动态存储变量是指该变量的存储空间不确定，在程序运行期间根据需要动态地为该变量分配存储空间。一般来说全局变量为静态存储变量，局部变量为动态存储变量。

在进行程序设计的时侯经常需要给一些变量赋以初值，C 语言允许在定义变量的同时给变量赋初值。例如：

unsigned char data val=5;

int xdata y=10000;

实验二

在 LED/16*2 字符液晶试验板上实现两个局部变量 val1、val2 的显示：val1 的值在右边 2 个数码管上显示，从 1 到 99 变化。Val2 的值在左边 2 个数码管上显示，从 1 到 99 显示奇数。

在我的文档中建立一个文件目录(cs8)，然后建立 cs8.uv2 的工程项目，最后建立源程序文件(cs8.c)。

输入下面的程序：

```
#include <REG51.H>// 序号(以下同):1
unsigned char code SEG7 [10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,};//2
/*===========3===========*/
void delay(unsigned int k);                    //4
//----------5----------
void main(void)                                //6
{                                              //7
unsigned char val1=1,val2=1;                   //8
while(1)                                       //9
{                                              //10
    P1= SEG7[ val1/10];                        //11
    P0= SEG7[ val1%10];                        //12
    P3= SEG7[ val2/10];                        //13
    P2= SEG7[ val2%10];                        //14
    delay(1000);                               //15
    val1=val1+1;                               //16
    if(val1==100){val1=1;}                     //17
    val2=val2+2;                               //18
    if(val2>99){val2=1;}                       //19
    }                                          //20
}                                              //21
//------------------22-----
void delay(unsigned int k)                     //23
{                                              //24
unsigned int i,j;                              //25
for(i=0;i<k;i++){                              //26
for(j=0;j<121;j++)                             //27
{;}}                                           //28
}                                              //29
```

编译通过后，将生成的 cs8.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 个数码管上显示从 1 到 99 变化。左边 2 个数码管上显示从 1 到 99 之间的奇数。

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：数码管 0~9 的字形码。

序号 3：程序分隔。

序号 4：延时子函数声明。

序号 5：程序分隔。

序号 6：定义函数名为 main 的主函数。

序号 7：main 的主函数开始。

序号 8：定义无符号字符型变量 val1、val2 并赋初值 1。

序号 9：while 循环语句，这里进行无限循环。

序号 10：while 循环语句开始。

序号 11：取出 val1 的十位数送 P1 口显示。

序号 12：取出 val1 的个位数送 P0 口显示。

序号 13：取出 val2 的十位数送 P3 口显示。

序号 14：取出 val2 的个位数送 P2 口显示。

序号 15：延时 1 秒以便观察清楚。

序号 16：变量 val1 加 1 以便下一次操作。

序号 17：如果 val1 等于 100，则重置为 1。

序号 18：变量 val2 加 2 以便下一次操作。

序号 19：如果 val2 大于 99，则重置为 1。

序号 20：while 循环语句结束。

序号 21：main 的主函数结束。

序号 22：程序分隔。

序号 23~29：延时子函数。

刚才实验的两个变量 val1、val2 均为局部变量，其作用范围在主函数 main()内。下面我们进行全局变量的实验。

实验三

在 LED/16*2 字符液晶试验板上实现全局变量 globe_x 的显示。两个子函数模块分别对全局变量 globe_x 进行加或减操作。

在我的文档中建立一个文件目录(cs9)，然后建立 cs9.uv2 的工程项目，最后建立源程序文件(cs9.c)。

输入下面的程序：

```
#include <REG51.H>// 序号(以下同):1
unsigned char code SEG7 [10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,};//2
int data globe_x;                              //3
//----------4----------
void add(void);                                //5
void subb(void);                               //6
void delay(unsigned int k);                    //7
//----------8----------
void main(void)                                //9
{                                              //10
while(1)                                       //11
{ add();                                       //12
P2= SEG7[globe_x /100];//13
P1= SEG7[(globe_x %100)/10];                   //14
    P0= SEG7[globe_x%10];                      //15
    delay(1000);                               //16
    subb();                                    //17
P2= SEG7[globe_x /100];                        //18
P1= SEG7[(globe_x %100)/10];                   //19
    P0= SEG7[globe_x%10]; //20
    delay(1000);                               //21
    if(globe_x>999) {globe_x=0;}               //22
    }                                          //23
}                                              //24
//----------25------
void add(void)                                 //26
{                                              //27
globe_x =globe_x+3;                            //28
}                                              //29
//----------30------
 void subb(void)                               //31
 {                                             //32
 globe_x =globe_x-2;                           //33
 }                                             //34
   //----------35-------
 void delay(unsigned int k)                    //36
 {                                             //37
   unsigned int i,j;                           //38
 for(i=0;i<k;i++){                             //39
 for(j=0;j<121;j++)                            //40
 {;}}                                          //41
 }                                             //42
```

编译通过后，将生成的 cs9.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。我们发现，add()与 subb()两个子函数都能对全局变量 globe_x 进行操作。Add()的作用使 globe_x 每次加 3，subb()的作用使 globe_x 每次减 2。原因是 globe_x 为定义在程序起始处的全局变量，这样从它定义处之后的任何函数语句都能对其操作，即它的作用范围是“全局”性的。

程序分析。

序号 1(程序解释，以下同)：包含头文件 REG51.H。

序号 2：数码管 0~9 的字形码。

51系列单片机的基础知识

◆陈阳海

上一讲，我们结合一个简单的单片机应用系统及其设计过程，说明了单片机系统的开发与采用传统电路设计产品的不同之处，并结合单片机产品的设计流程指出初学者在学习单片机时应当关注的内容和学习的重点、难点。本讲将对一些与学习51系列单片机的硬件结构和程序设计密切相关的基础知识，包括单片机中的数和数据的表示方法、单片机中存储器的类型及特点、常用80C51单片机及其主要特点等知识进行介绍，以利读者更好地理解和掌握后续的内容。

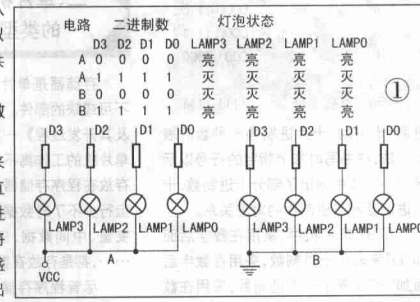
一、单片机中的数和数的表示方法

1. 单片机中的数

我们在日常生活中都是采用十进制来计数的，从1到9，逢10进1。而单片机的中央处理器CPU只能识别低电平和高电平，即"0"和"1"两种状态，所以在单片机中是用一连串的0和1来代表各种命令和数据的，也就是说，两种以上的状态或2以上的数，要通过多位0和1来表示，即单片机是采用二进制来计数的。比如，十进制数的2可以用两位二进制数的10来表示，十进制数的15可以用四位二进制数的1111表示，十进制数的256则要用八位二进制数的11111111表示……，四位二进制数可以表示从0到15共16个数，而八位二进制数则可以表示从0到255共256个数。

2. 单片机中数的表示方法

用二进制数来表示某种状态是直观的，如用四位二进制数可以直观地将图1中并接的4盏灯的亮或灭表示出来；用十进制表示各种常数，直接且便于计算；而用十六进制数0～9、A、B、C、D、E、F来表示4位二进制数书写起来则更加方便。因此，在单片机中有二




```
序号3:在data区定义全局变量globe_x。
序号4:程序分隔。
序号5:子函数add()声明。
序号6:子函数subb()声明。
序号7:延时子函数声明。
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的主函数开始。
序号11:while循环语句进行无限循环。
序号12:while循环语句开始。调用add()子函数。
序号13:取出全局变量globe_x的百位数送P2口显示。
序号14:取出全局变量globe_x的十位数送P1口显示。
序号15:取出全局变量globe_x的个位数送P0口显示。
序号16:延时1秒以便观察清楚。
序号17:调用subb()子函数。
序号18:取出全局变量globe_x的百位数送P2口显示。
序号19:取出全局变量globe_x的十位数送P1口显示。
序号20:取出全局变量globe_x的个位数送P0口显示。
序号21:延时1秒以便观察清楚。
序号22:若全局变量globe_x的值大于999,则置0。
序号23:while循环语句结束。
序号24:main的主函数结束。
序号25:程序分隔。
序号26:定义函数名为add的子函数。
序号27:add子函数开始。
序号28:全局变量globe_x加3。
序号29:add子函数结束。
序号30:程序分隔。
序号31:定义函数名为subb的子函数。
序号32:subb子函数开始。
序号33:全局变量globe_x减2。
序号34:subb子函数结束。
序号35:程序分隔。
序号36-42:延时子函数。
```