第3章 PART3单片机的C51语言

51汇编语言能**直接操作单片机的系统硬件**，指令执行速度快。但其程序**可读性差，且编写、移植困难**。

C51是为51系列单片机设计的一种C语言，其特点：

- 结构化语言，**代码紧凑**——效率可与汇编语言媲美
- 接近真实语言，**程序可读性强**——易于调试、维护
- 库函数丰富，**编程工作量小**——产品开发周期短
- 机器级控制能力，**功能很强**——适合于嵌入式系统开发
- 与汇编指令无关，**易于掌握**——在单片机基础上上手快

C51语言已成为51系列单片机程序开发的主流语言。

第3章 PART3 单片机的C51语言

3.7 C51的程序结构

3.8 C51的数据结构

3.9 C51与汇编语言的混合编程

3.10 C51初步应用编程

C51与标准C语言对比

相同之处：语法规则、程序结构、编程方法

差异之处：数据结构（数据类型、存储模式）、中断处理（第5章）、端口扩展（第9章）

学习思路：

① 在标准C的基础上学习数据结构（数据类型、存储模式）

② 掌握C51软件开发方法（开发环境搭建）

③在C51的基础上学习单片机I/O口应用（进阶）

3.7 C51程序结构

随着80C51单片机硬件性能的提升，尤其是片内程序存储器容量的增大和时钟工作频率的提高，已基本**克服了高级语言产生代码长、运行速度慢、不适合单片机使用的致命缺点。**由此C51已经得到广泛的推广和应用.

```
#include<头文件>
函数类型说明
全局变量定义
main(){
    局部变量定义
    <程序体>
}
func1(){
    局部变量定义
    <程序体>
}
┆
funcN(){
    局部变量定义
    <程序体>
}
```

C51与标准C程序结构完全相同

① 程序由函数组成（一个主函数，或一个主函数和若干自定义函数）；

② 利用预处理命令对变量或函数进行集中定义或说明；

③ 函数和变量都需遵循先定义后使用的基本原则；

④ 主函数中的所有语句执行完毕,则程序结束。

预处理命令--------
函数说明---------
全局变量定义------
主函数------------
程序体
自定义函数---------
局部变量定义--------
程序体------

```
#include <REG51.H>          //51 单片机头文件
void delay();               //延时函数声明
sbit p1_0=P1^0;             //输出端口定义
main() {                    //主函数
    while(1) {              //无限循环体
        p1_0=0;             //P1.0="0",led 亮
        delay();            //延时
        p1_0=1;             //P1.0="1",led 灭
        delay();            //延时
    }
}

void delay (void) {         //延时函数
    unsigned char i;        //字符型变量 i 定义
    for (i=200;i>0;i--);    //循环延时
}
```

C51的库函数

C51的强大功能及其高效率的重要体现之一在于，其提供了丰富的可直接调用的库函数。C51的库函数分为**本征库函数和非本征库函数。**

C51提供的本征函数在编译时直接将固定的代码插入当前行，而不是用ACALL和LCALL语句实现的，这样就大大提供了函数访问的效率，而非本征函数则必须由ACALL及LCALL调用。

C51的本征库函数只有9个，列举如下：

crol，_cror_：将char型变量循环向左（右）移动指定位数后返回。

iror，_irol_：将int型变量循环向左（右）移动指定位数后返回。

lrol，_lror_：将long型变量循环向左（右）移动指定位数后返回。

nop：相当于插入汇编指令NOP。

testbit：相当于JBC bitvar测试该位变量并跳转同时清除。

chkfloat： 测试并返回源点数状态。

使用上述函数时，源程序开头必须包含 #include <intrins.h>一行。

第3章 PART3 单片机的C51语言

3.7 C51的程序结构

3.8 C51的数据结构

3.9 C51与汇编语言的混合编程

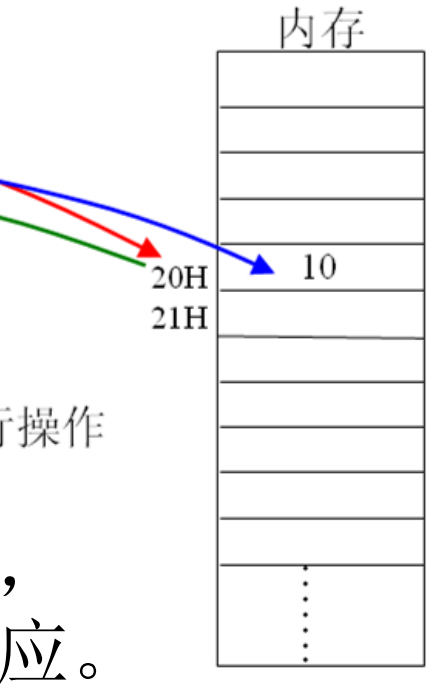
3.10 C51初步应用编程

1. C51的变量

在程序执行过程中，数值可以发生改变的量称为**变量**。

例如

```
main()
 {  char  a;
    a=10;
    printf("%d",a);
 }
```



程序中使用变量名对内存单元进行操作

变量名与存储单元地址相对应，
变量值与存储单元的内容相对应。

C51变量定义的四要素:

【存储种类】　数据类型　【存储类型】　变量名

（标准C）　（标准C+C51）（C51特有）（标准C）

＊ 括号项——可以缺省（但需有缺省值）

【存储种类】 数据类型 【存储类型】 变量名

- **自动变量（auto）：** 函数中所有的非静态局部变量
- 静态变量（static）：在变量前加上static关键字的变量。
- 外部变量（extern）：一般用作全局变量作用域的扩展
- 寄存器类型（register）：一般经常被使用的的变量（如某一变量需要计算几千次）可以设置成寄存器变量，register变量会被存储在寄存器中，计算速度远快于存在内存中的非register变量。

【存储种类】 数据类型 【存储类型】 变量名

数 据 类 型	长 度	大 小
unsigned char	1B	0～255
char	1B	−128～+127
unsigned int	2B	0～65535
int	2B	−32768～+32767
unsigned long	4B	0～4294967295
long	4B	−2147483648～+2147483647
float	4B	±1.175494E-38～±3.402823E+38
*	1B～3B	对象的地址
bit	位	0或1
sfr	1B	0～255
sfr16	2B	0～65535
sbit	位	0或1

C51扩充数据类型：bit、sfr或sfr16、sbit

bit 型

关键词bit用于定义一个位变量，语法规则：

bit bit_name [= 0或1];

例如：bit door = 0 ; //定义一个叫door的位变量且初值为0

与标准C的变量定义相似：

int a = 5 ; //定义一个初值为5的整形变量a

//语法规则：**int** int_name [= 常数];

sfr或sfr16型

MCU中包含的SFR，如何定义与这些单元相关的变量？

序号	特殊功能寄存器名称	符号	字节地址	位地址							
1	P0 口锁存器	P0	80H	87H	86H	85H	84H	83H	82H	81H	80H
2	堆栈指针	SP	81H								
3	数据地址指针（低 8 位）	DPL	82H								
4	数据地址指针（高 8 位）	DPH	83H								
5	电源控制寄存器	PCON	87H								
6	定时器/计数器控制寄存器	TCON	88H								
7	定时器/计数器方式控制寄存器	TMOD	89A								
8	定时器/计数器 0（低 8 位）	TL0	8AH								
9	定时器/计数器 0（高 8 位）	TL1	8BH								
10	定时器/计数器 1（低 8 位）	TH0	8CH								

关键词**sfr或sfr16**用于定义**SFR字节地址变量**，语法规则：

sfr 或 sfr16 sfr_name = 字节地址常数;

表 2.3 SFR 的名称及其分布

序号	特殊功能寄存器名称	符号	字节地址	位地址							
1	P0 口锁存器	P0	80H	87H	86H	85H	84H	83H	82H	81H	80H
2	堆栈指针	SP	81H								
3	数据地址指针（低 8 位）	DPL	82H								
4	数据地址指针（高 8 位）	DPH	83H								
5	电源控制寄存器	PCON	87H								
6	定时器/计数器控制寄存器	TCON	88H								
7	定时器/计数器方式控制寄存器	TMOD	89A								
8	定时器/计数器 0（低 8 位）	TL0	8AH								
9	定时器/计数器 0（高 8 位）	TL1	8BH								
10	定时器/计数器 1（低 8 位）	TH0	8CH								

例如，sfr P0 = 0x80; //定义P0口地址80H

sfr PCON = 0x87; //定义PCON地址87H

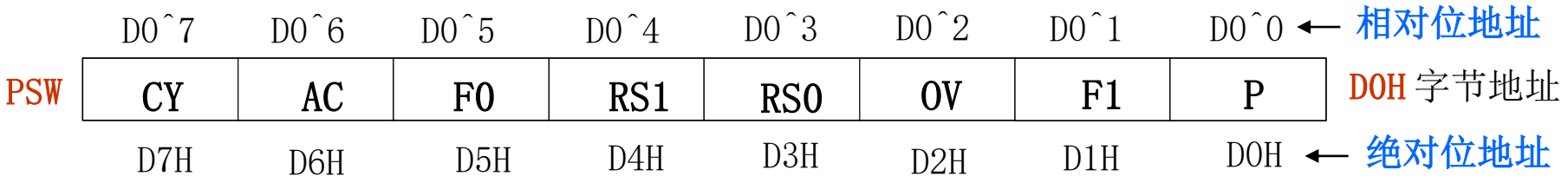
sfr16 DPTR=0x82; //定义DPTR的低端地址82H

注意：C语言中十六进制整数是数值前加0X前缀

sbit型

部分SFR具有位地址，如何定义与这些位地址相关的变量？

序号	特殊功能寄存器名称	符号	字节地址	位地址							
11	定时器/计数器 1（高 8 位）	TH1	8DH								
12	串行口锁存器	SBUF	99H								
13	P2 口锁存器	P2	A0H	A7H	A6H	A5H	A4H	A3H	A2H	A1H	A0H
14	中断允许控制寄存器	IE	A8H	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H
15	P3 口锁存器	P3	B0H	B7H	B6H	B5H	B4H	B3H	B2H	B1H	B0H
16	中断优先级控制寄存器	IP	B8H	BFH	BEH	BDH	BCH	BBH	BAH	B9H	B8H
17	P1 口锁存器	P1	90H	97H	96H	95H	94H	93H	92H	91H	90H
18	串行口控制寄存器	SCON	98H	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H
19	程序状态字	PSW	D0H	D7H	D6H	D5H	D4H	D3H	D2H	D1H	D0H
20	累加器	A	E0H	E7H	E6H	E5H	E4H	E3H	E2H	E1H	E0H
21	B 寄存器	B	F0H	F7H	F6H	F5H	F4H	F3H	F2H	F1H	F0H



两种位地址表达形式：相对位地址、绝对位地址

关键词sbit用于定义**SFR位地址变量**，三种定义形式：

1）将SFR的**绝对位地址**定义为位变量名

sbit bit_name = 位地址常数;

例如， sbit CY = 0xD7;

2）将SFR的**相对位地址**定义为位变量名

sbit bit_name = sfr字节地址 ^ 位位置;

例如， sbit CY = 0xD0^7;

3）将SFR的**相对位位置**定义位变量名

sbit bit_name = sfr_name ^ 位位置;

例如， sbit CY = PSW^7;

C51编译器在头文件“REG52.H”中定义了全部sfr/sfr16和sbit变量。

```
/*--------------------------------------------------------------------------
REG51.H

Header file for generic 80C51 and 80C31 microcontroller.
Copyright (c) 1988-2002 Keil Elektronik GmbH and Keil Software
All rights reserved.
--------------------------------------------------------------------------*/

#ifndef __REG51_H__
#define __REG51_H__

/*  BYTE Register  */
sfr P0   = 0x80;
sfr P1   = 0x90;
sfr P2   = 0xA0;
sfr P3   = 0xB0;
sfr PSW  = 0xD0;
sfr ACC  = 0xE0;
sfr B    = 0xF0;
sfr SP   = 0x81;
sfr DPL  = 0x82;
sfr DPH  = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0  = 0x8A;
```

```
/*  PSW   */
sbit CY   = 0xD7;
sbit AC   = 0xD6;
sbit F0   = 0xD5;
sbit RS1  = 0xD4;
sbit RS0  = 0xD3;
sbit OV   = 0xD2;
sbit P    = 0xD0;

/*  TCON  */
sbit TF1  = 0x8F;
sbit TR1  = 0x8E;
sbit TF0  = 0x8D;
sbit TR0  = 0x8C;
sbit IE1  = 0x8B;
sbit IT1  = 0x8A;
sbit IE0  = 0x89;
sbit IT0  = 0x88;

/*  IE   */
sbit EA   = 0xAF;
sbit ES   = 0xAC;
sbit ET1  = 0xAB;
```

用一条预处理命令**#include <REG52.H>**把这个头文件包含到C51程序中，无需重新定义即可直接使用它们的名称。

应用举例：

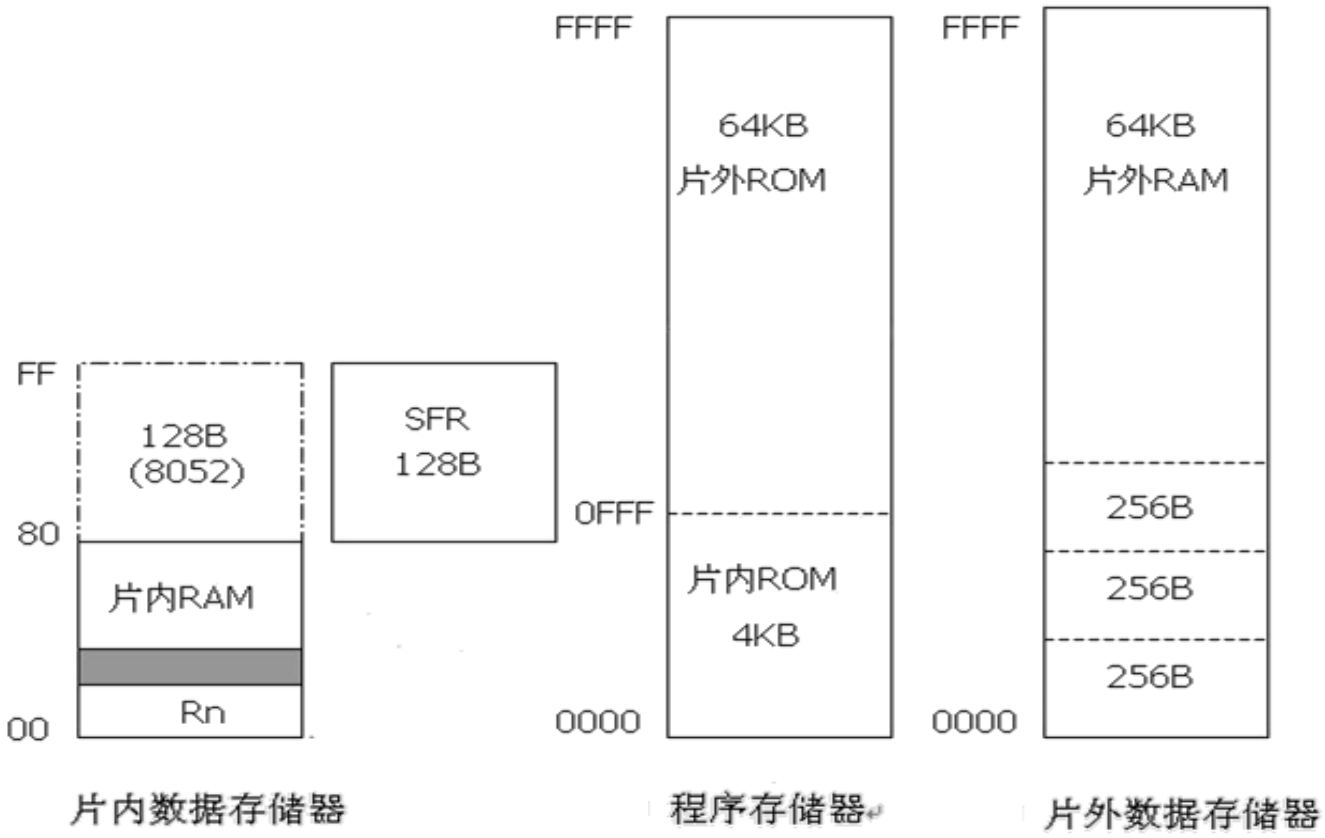
```
#include <REG51.H>          //51 单片机头文件
void delay();               //延时函数声明
sbit p1_0=P1^0;             //输出端口定义
main() {                    //主函数
    while(1) {              //无限循环体
        p1_0=0;             //P1.0="0",led 亮
        delay();            //延时
        p1_0=1;             //P1.0="1",led 灭
        delay();            //延时
    }
}

void delay (void) {         //延时函数
    unsigned char i;        //字符型变量 i 定义
    for (i=200;i>0;i--);    //循环延时
}
```

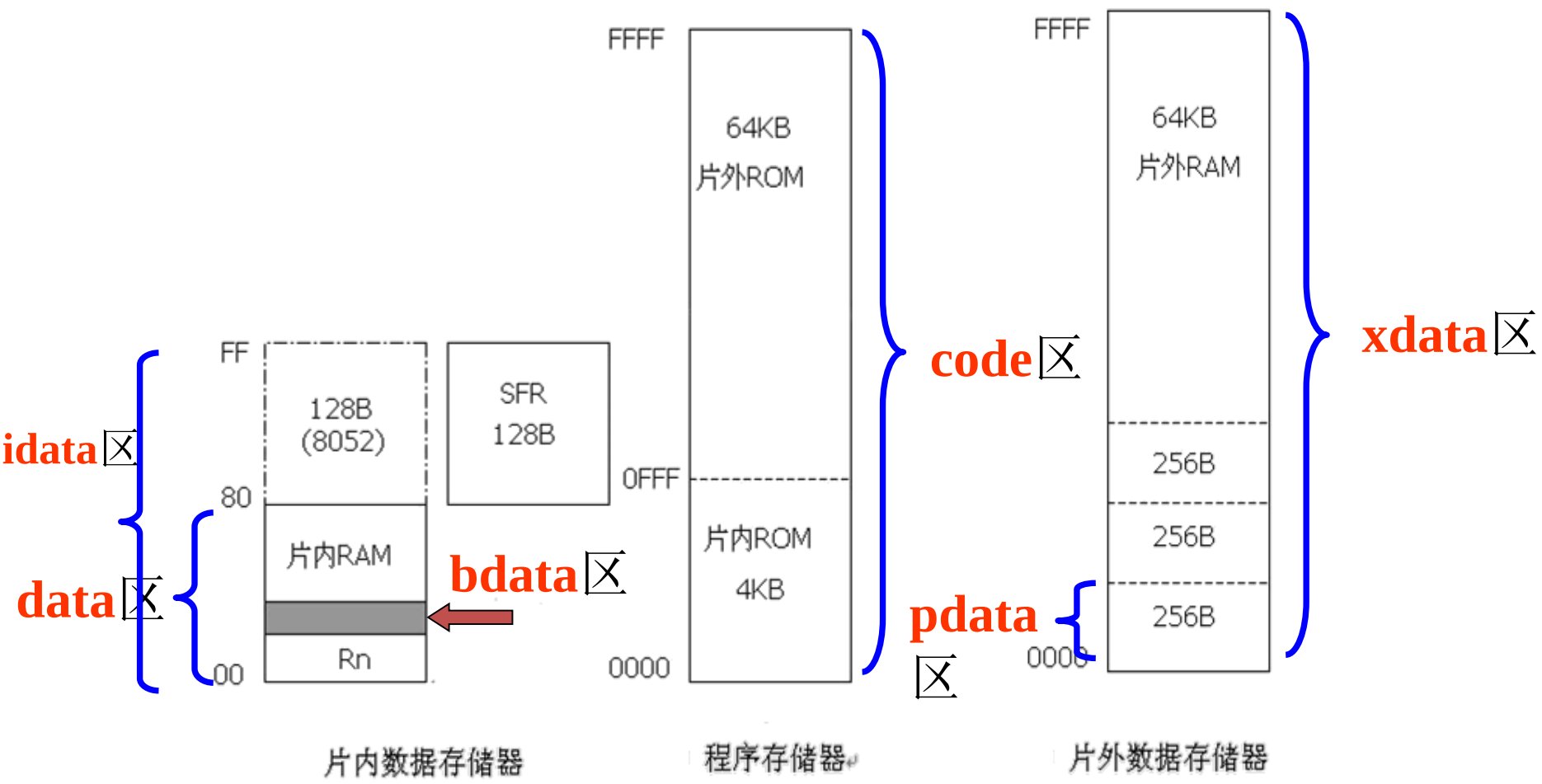
【存储种类】 数据类型 【存储类型】 变量名

51单片机的 3个逻辑存储空间：

片内数据存储器、片外数据存储器和程序存储器。



建立C51存储类型与存储空间的对应关系



1）data区

对data区的寻址是最快的，所以应该把使用频率高的变量放在data区，由于空间有限，必须注意使用data区，data区除了包含程序变量外，还包含了堆栈和寄存器组data区。

2）bdata区

当在DATA区的位寻址区定义变量，这个变量就可进行位寻址，并且声明位变量。这对状态寄存器来说十分有用，因为它可以单独使用变量的每一位，而不一定要用位变量名引用位变量。

3）idata区

idata区也可以存放使用比较频繁的变量，使用寄存器作为指针进行寻址。在寄存器中设置8位地址进行间接寻址，与外部存储器寻址比较，它的指令执行周期和代码长度都比较短。

4）pdata和xdata区

在这两个区声明变量和在其他区的语法是一样的，pdata区只有256B，而xdata区可达65536B

5）code区

code区即80C51的程序代码区，所以代码区的数据是不可改变的，80C51的代码区不可重写。一般代码区中可存放数据表，跳转向量和状态表。

存储器类型	描　述
data	**直接寻址的片内数据存储器低128B，访问速度最快**
bdata	**可位寻址的片内数据存储器（地址20H～2FH共16B）允许位和字节混合访问**
idata	**间接寻址片内数据存储器256B。允许访问片内全部地址**
pdata	**分页寻址片外数据存储器256B，使用指令MOVX @Rn访问，需要2个指令周期**
xdata	**寻址片外数据存储器64KB，使用指令MOVX @DPTR访问**
code	**寻址程序存储器区64KB，使用指令MOVC @A+DPTR访问**

【存储种类】 数据类型 【存储类型】 变量名

变量名可以由字母、数字和下划线三种字符组成，且第一个字符必须为字母或下划线，变量名长度随编译系统而定。

变量名具有字母大小写的敏感性，如SUM和sum代表不同的变量。

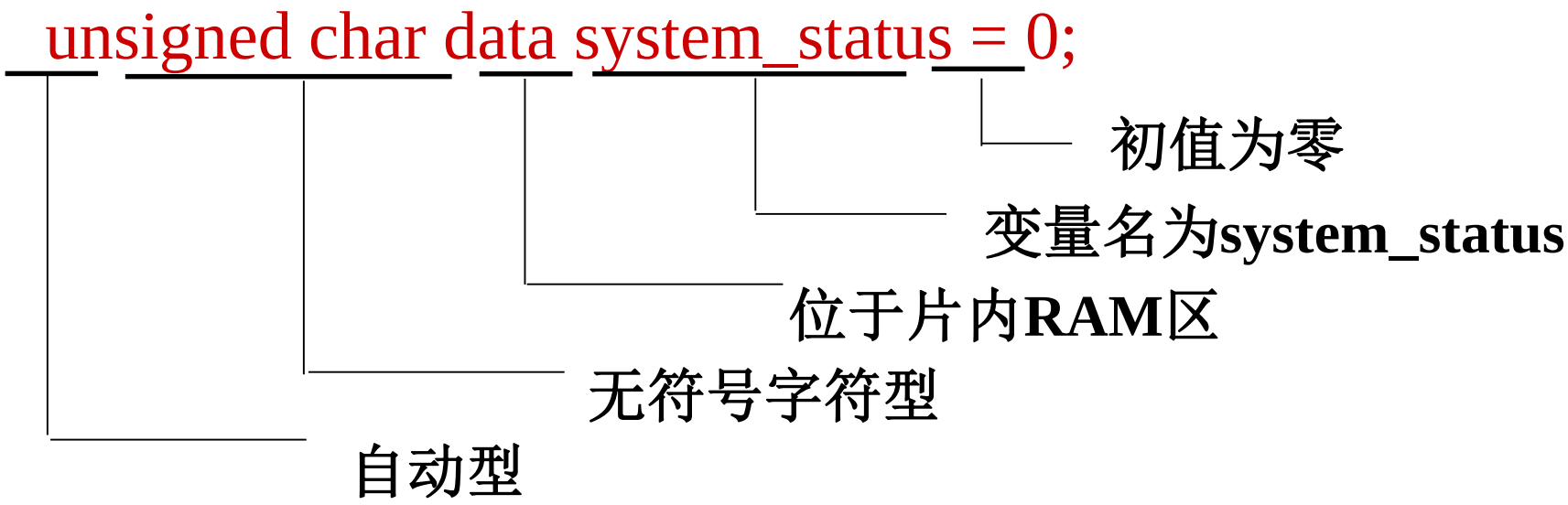
强调：头文件中定义的变量都是大写的，若程序采取小写变量则需要重新定义。

变量名不得使用标准C语言和C51语言的关键字。

C51扩展的若干关键字一览表

关键字	用　途	说　明
_at_	地址定位	为变量进行存储器绝对空间地址定位
alien	函数特性声明	声明与 PL/M-51 编译器的接口
bdata	存储器类型说明	可位寻址的内部数据存储器
bit	位变量声明	声明一个位变量或位函数
code	存储器类型声明	程序存储器
compact	存储模式声明	声明一个紧凑编译存储模式
data	存储器类型说明	直接寻址的内部数据存储器
far	远变量声明	Keil 用 3BYTE 指针来引用它
idata	存储器类型声明	间接寻址的内部数据存储器
interrupt	中断函数声明	定义一个中断服务函数
large	存储模式声明	声明一个大编译存储模式

数据结构定义举例



//定义system_status为无符号字符型自动变量，该变量位于data区中且初值为0。

2. C51的指针

标准C语言指针的一般定义形式为：

数据类型 *指针变量名 [= &被指向变量名]；

其中，指针变量指向一个由“数据类型”说明的变量。被指向变量和指针变量都位于C编译器默认的存储区中。

例如：
```
int a =’A’;
int *p1= &a;
```

这表示p1是一个指向int型变量的指针变量，此时p1的值为int型变量a的地址，而a和p1两个变量都位于C编译器默认的内存区域中。

对于C51来讲，指针定义应包括以下信息:

1）指针变量的存储类型（自身位于哪个存储区中）？
2）被指向变量的数据类型和存储类型？

C51指针的一般定义形式：

数据类型 [存储类型1] * [存储类型2] 变量名 [=&被指向变量名];

数据类型——被指向变量的类型，如int型或char型

存储类型1——被指向变量所在的存储区，缺省时由地址赋值关系决定

存储类型2——指针变量所在的存储区，缺省时为编译器默认的存储区

例1　若采用SMALL编译模式，试解释下述定义的含义。

```
char xdata a = 'A';
char *ptr = &a;
```

解：ptr是一个指向char型变量的指针，它本身位于SMALL编译模式默认的data存储区里，此时它指向位于xdata存储区里的char型变量a的地址。

Small模式常用于data区，compact模式常用于pdata区Large模式常用于xdata区。

例2　试解释下述定义的含义

```
char xdata a = 'A';
char *ptr = &a;
char idata b = 'B';
ptr = &b;
```

前两句与例1相同

```
char xdata a = 'A';
char *ptr = &a;
```

解：以char *ptr形式定义的指针变量，既可指向位于xdata存储区的char型变量a的地址，也可指向位于idata存储区的char型变量b的地址（由赋值操作关系决定）。

为了提高源代码输入效率及程序的可移植性，通常我们对C51编译器的数据类型重新定义！

```
#define   byte    unsigned char
#define   uchar   unsigned char
#define   BYTE    unsigned char
#define   word    unsigned int
#define   uint    unsigned int
#define   WORD    unsigned int
#define   ulong   unsigned long
```

例如

```
unsigned char  v1;
uchar  v1;
```

这2个变量定义等价的

绝对地址的访问

1. 使用指针

采用指针的方法，可实现在C51程序中对任意指定的存储器地址进行操作。

```
uchar xdata *xdp; /*定义一个指向xdata存储器空间的指针*/
char data *dp;     /*定义一个指向DATA存储器空间的指针*/
uchar idata *idp;  /*定义一个指向idata存储器空间的指针*/
xdp=0x1000;       /*xdata指针赋值，指向xdata存储器地址1000H处*/
*xdp=0x5A;        /*将数据5AH送到xdata的1000H单元*/
```

2. 使用C51运行库中预定义宏

```
#define CBYTE((unsigned char volatile *)0x50000L)
#define DBYTE((unsigned char volatile *)0x40000L)
#define PBYTE((unsigned char volatile *)0x30000L)
#define XBYTE((unsigned char volatile *)0x20000L)
#define CWORD((unsigned int volatile *)0x50000L)
#define DWORD((unsigned int volatile *)0x40000L)
#define PWORD((unsigned int volatile *)0x30000L)
#define XWORD((unsigned int volatile *)0x20000L)
```

源程序开头必须包含 **#include <absacc.h>**一行。

例如

```
ui_var1 = XWORD [0x0002];
/*访问外部 RAM 0004H～0005H地址的内容*/
uc_var1 = XBYTE [0x0002];
/*访问外部 RAM 0002H地址的内容*/
```

3. 使用C51扩展关键字 _at_

使用_at_对指定的存储器空间的绝对地址进行定位，一般格式如下：

[存储器类型] 数据类型 标识符 _at_ 常数

例如

```
uchar xdata xram[0x8000]   _at_ 0x0000;
/*在外部RAM空间0000H处定义了一个一维数组
```

使用_at_定义的变量只能是全局变量。

第3章 PART3 单片机的C51语言

3.7 C51的程序结构

3.8 C51的数据结构

3.9 C51与汇编语言的混合编程（自学）

3.10 C51初步应用编程

第3章 PART3 单片机的C51语言

3.7 C51的程序结构

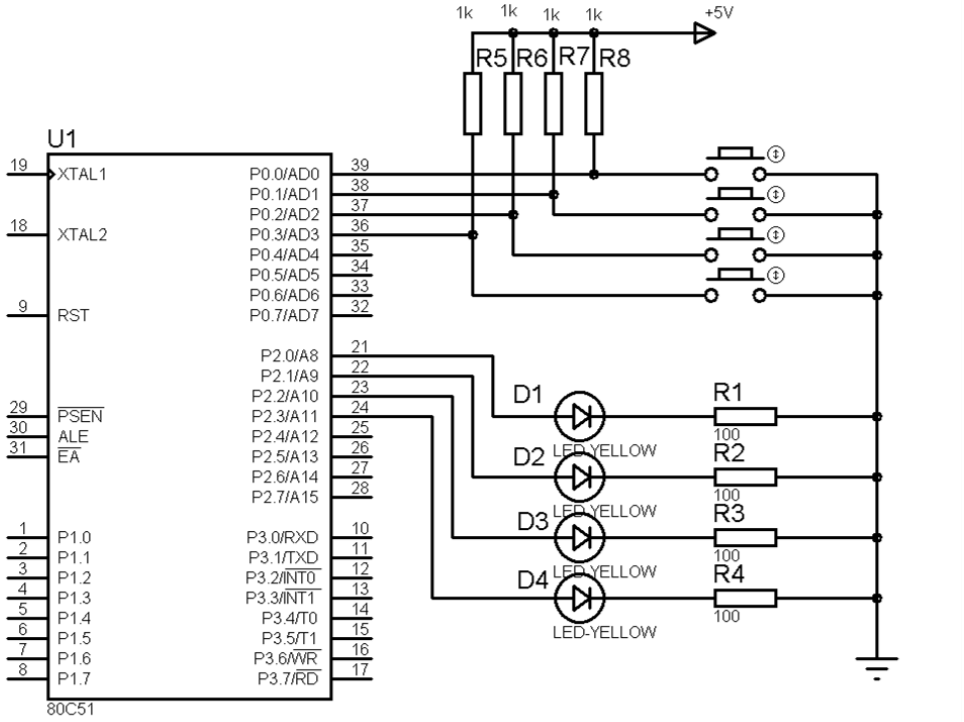
3.8 C51的数据结构

3.9 C51与汇编语言的混合编程（自学）

3.10 C51初步应用编程

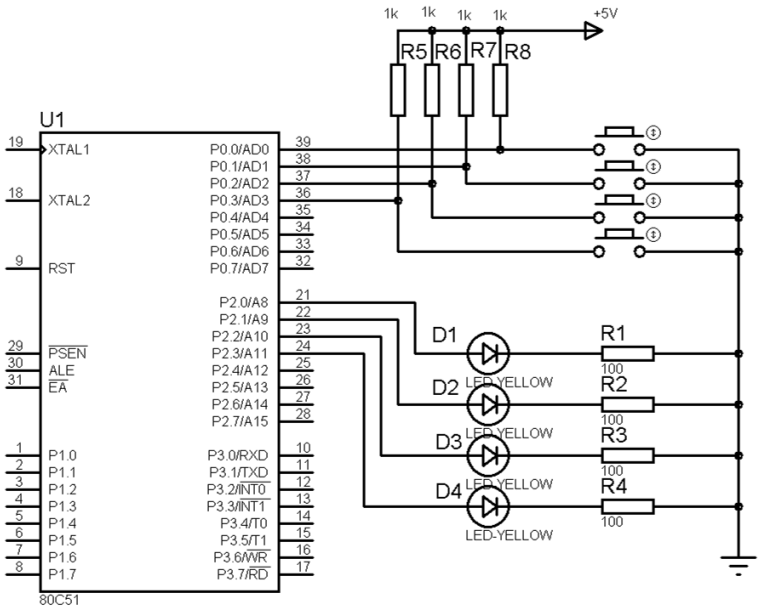
例1 独立按键识别

【要求】采用独立按键方式实现下述功能：开机时LED全熄，然后根据按键动作使相应灯亮，并将亮灯状态保持到按压其它键时为止。



独立按键——每个按键都彼此独立地各占有一位I/O口线。特点是电路简单，但占用I/O口线较多。

【分析】

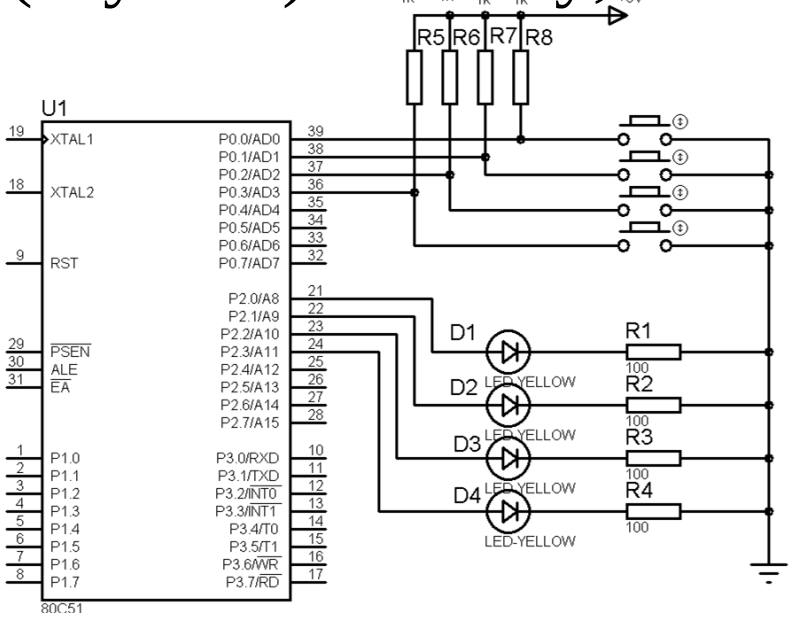


① 按键的闭合电平为0，但LED的驱动电平为1，故不能直接将P0口的状态送到P2口，而应使其先取反再送出；

② 为使按键抬起后LED能保持先前的点亮状态，需要在按键未压下期间禁止向P2输出P0状态值。

参考程序如下：

```
void main( ) {
   char key = 0;                    //定义按键变量
   P2=0;                            //初始状态为灯全灭
   while(1){
        key = ~P0 & 0x0f;     //读取按键状态,高4位清零
        if (key != 0) P2 = key;  //有按键动作时，P0状态值送P2
    }
}
```



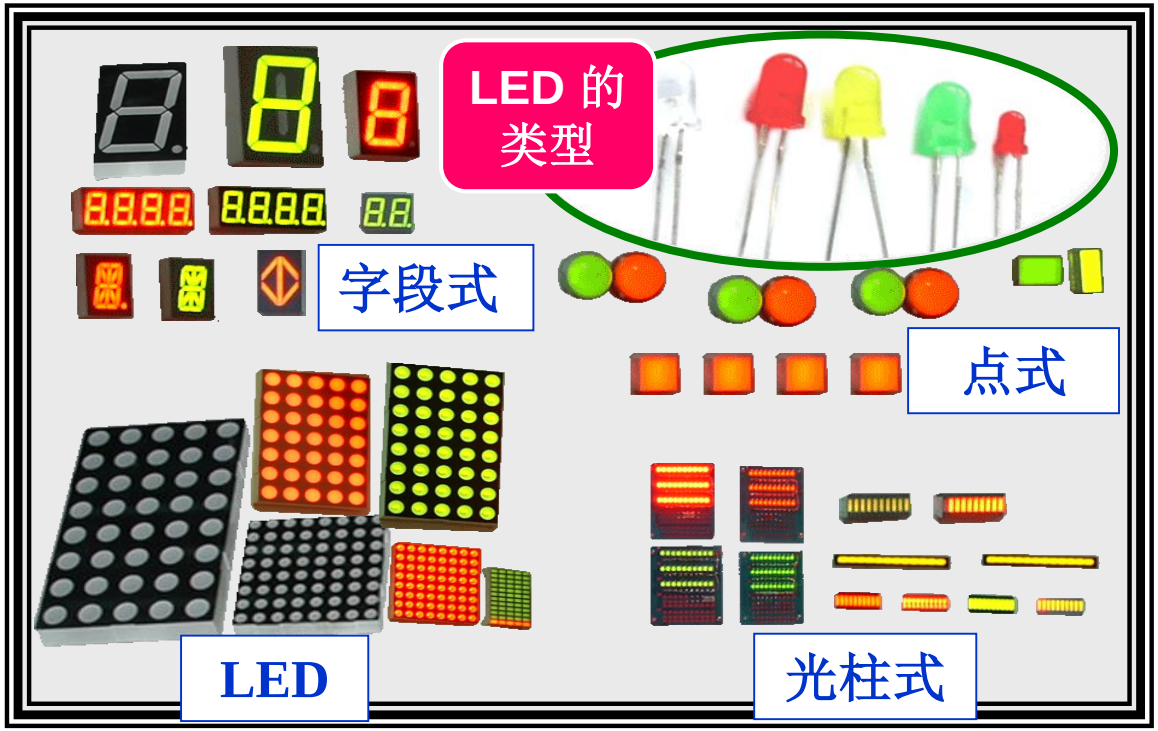
“取反”操作的优先级高于“与”操作



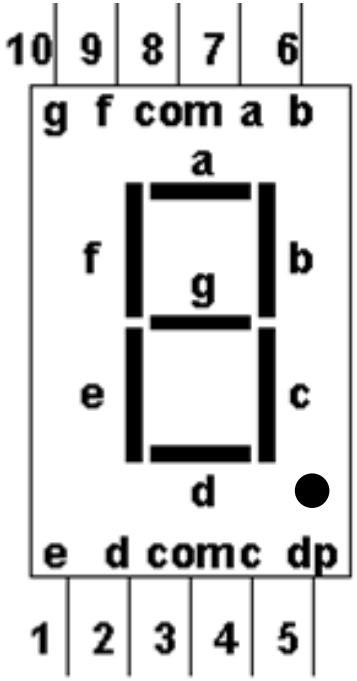


LED数码管原理与编程

LED显示元件——人机交互输出设备，其作用是指示中间运行结果与运行状态。

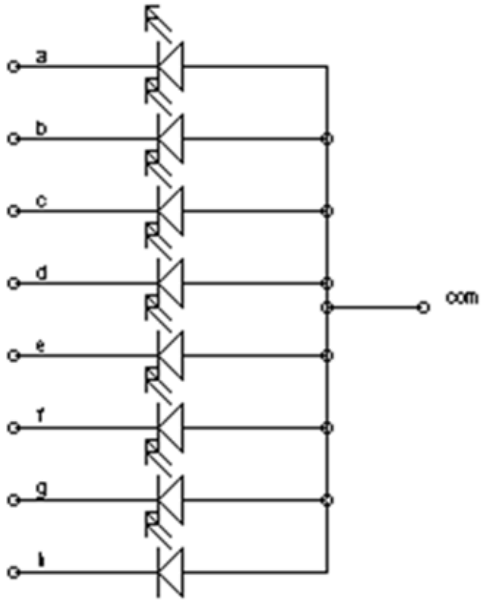


七段式LED显示器 （7-Segment Display）

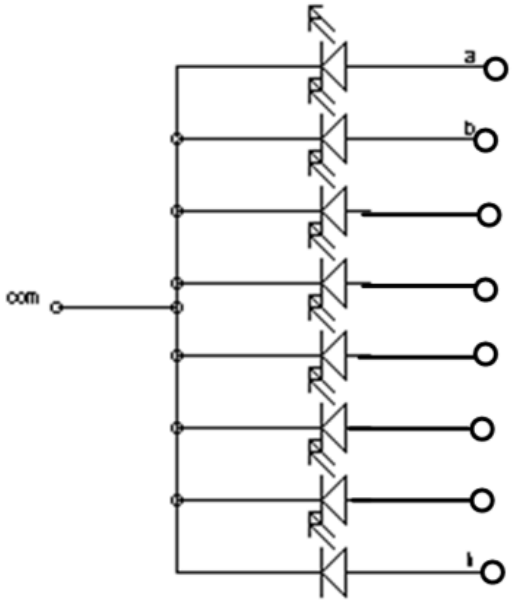


引脚配置

com为公共端

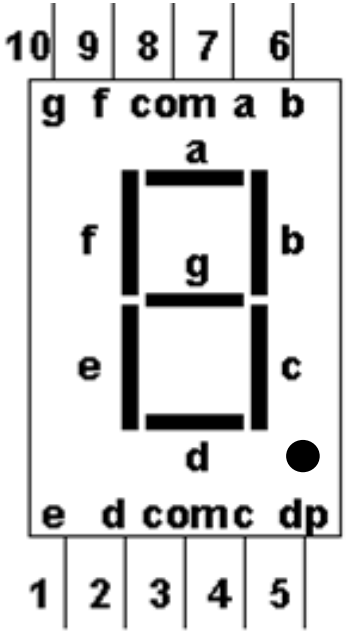


共阳极LED



共阴极LED

不同显示字符采需用不同LED组合关系实现（显示码，字模）



以共阴极为例，部分字符的显示码（字模）为：

显示字符	DP	g	f	e	d	c	b	a	显示码（共阴）
0	0	0	1	1	1	1	1	1	3FH
1	0	0	0	0	0	1	1	0	06H
2	0	1	0	1	1	0	1	1	5BH
3	0	1	0	0	1	1	1	1	4FH
4	0	1	1	0	0	1	1	0	66H
5	0	1	1	0	1	1	0	1	6DH

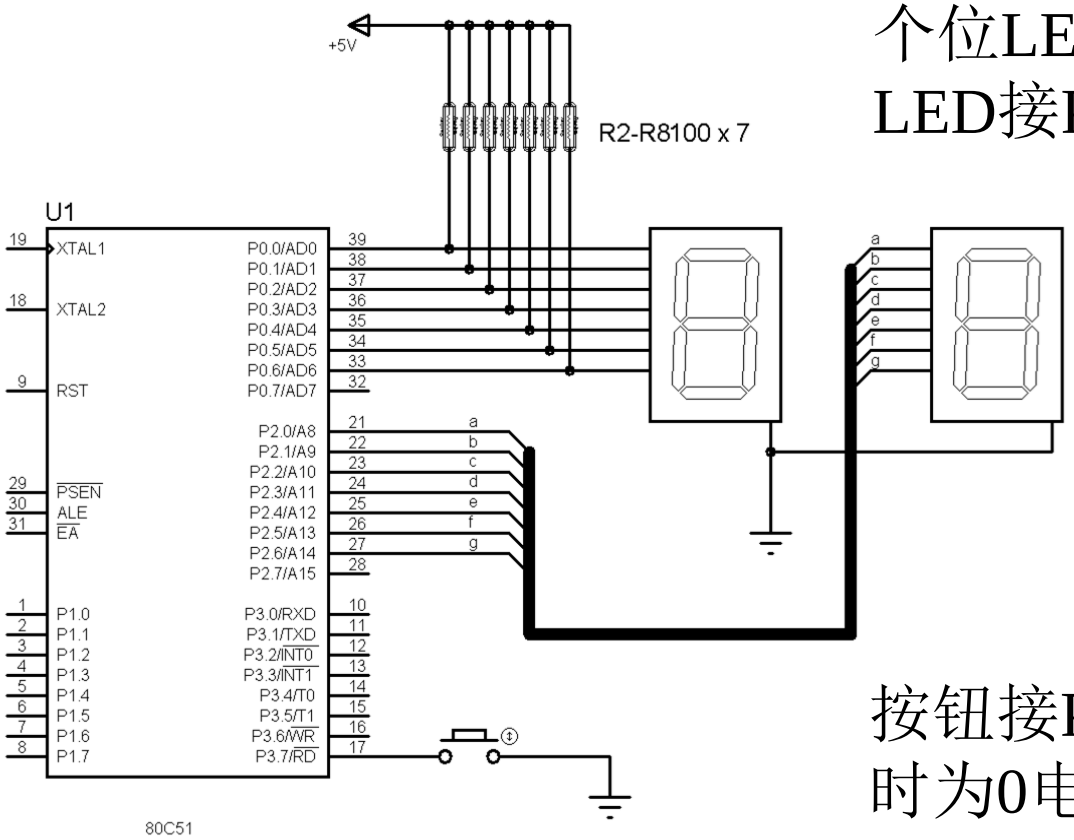
七段LED数码管的标准显示字符为0～9，A～F

其它字符：“L”字符的显示码——0011 1000B = 0x38

“H”字符的显示码——0111 0110B = 0x76

实例2　计数显示器

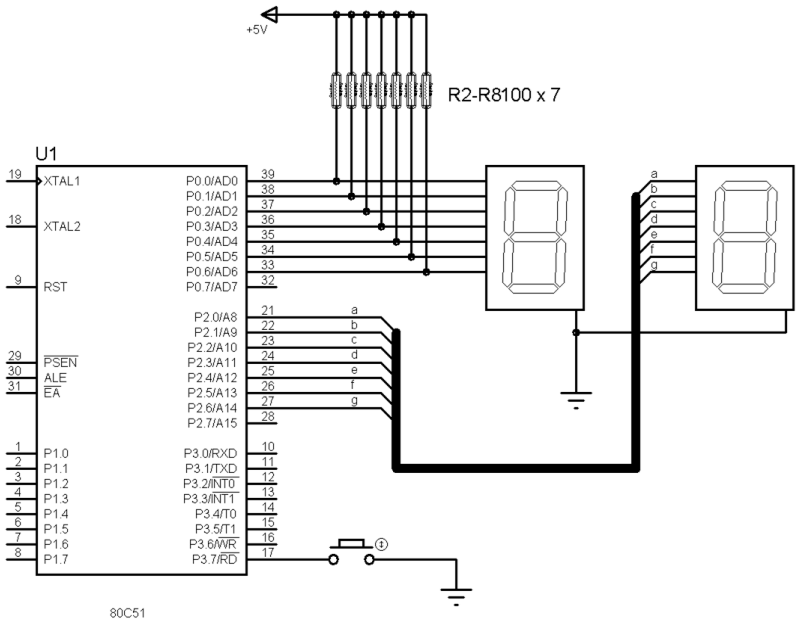
统计按键次数并以十进制形式显示，按键次数大于99后重新由0开始计数。

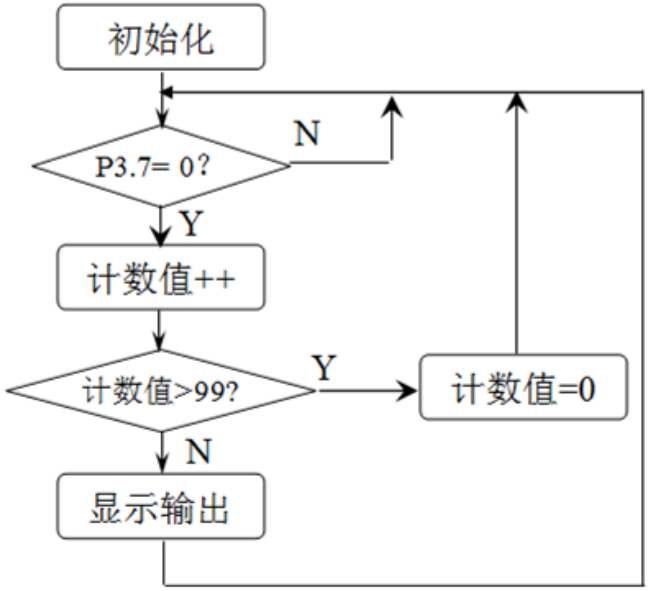


个位LED接P2口，十位LED接P0口（共阴型）

按钮接P3.7口线，按压时为0电平

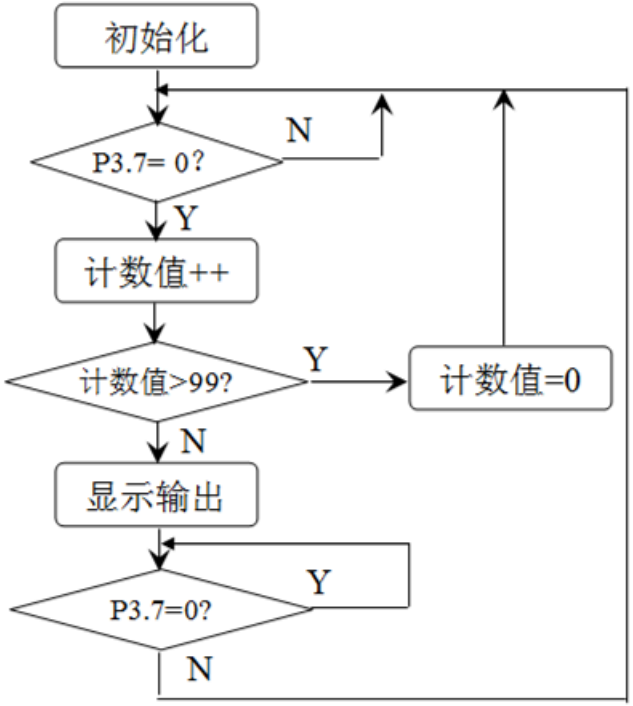
编程分析：

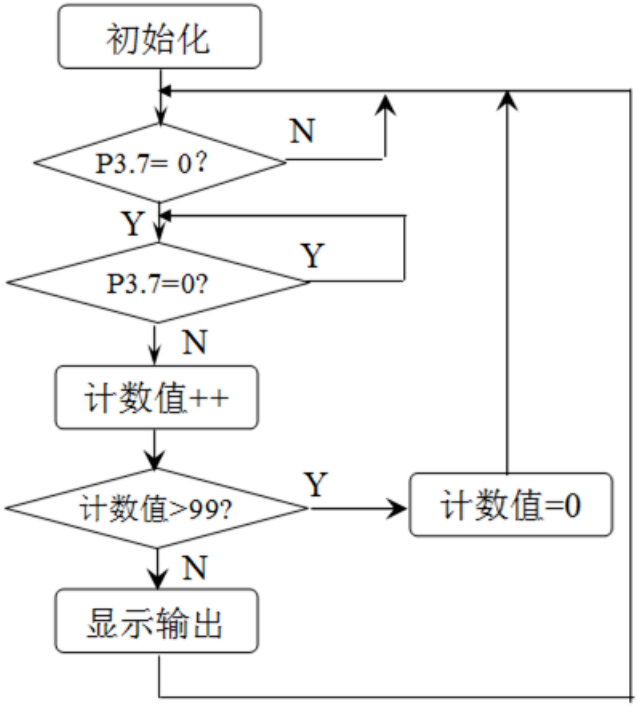




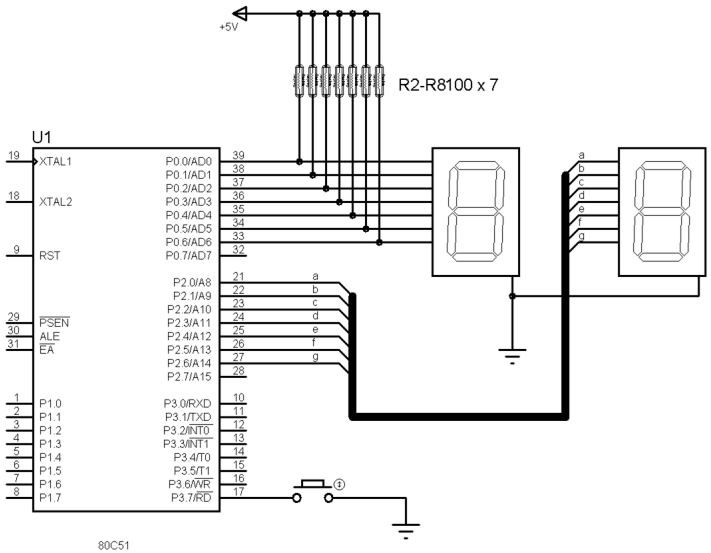
问题：按键压下时可能被连续计数

如何避免连续计数？


初始化
P3.7= 0?
N
Y
计数值++
计数值>99?
Y
计数值=0
N
显示输出
P3.7=0?
Y
N


初始化
P3.7= 0?
N
Y
P3.7=0?
Y
N
计数值++
计数值>99?
Y
计数值=0
N
显示输出

显示输出：如何将计数值拆成两个独立的数？



解决办法：

取模运算（%10）→个位　　P2 = table[count%10];

整除10运算（/10）→十位　　P 0 = table[count/10];

初始化：　　P2 = P0 = table[0];

参考程序

```
#include <reg51.H>
sbit P3_7=P3^7;                          //定义计数器端口
unsigned char count =0;                  //定义计数器
unsigned char code table[]=
    {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

void main(void) {
    P0=P2=table[0];                      //显示初值00
    while(1) {
        if(P3_7==0){                     //检测按键是否压下
            count++;                     //计数器增1
            if(count==100) count=0;      //判断循环是否超限
            P0=table[count/10];          //十位输出显示
            P2=table[count%10];          //个位输出显示
            while(P3_7==0);              //等待按键抬起，防止连续计数
        }
    }
}
```





3.10.2 中断服务程序

STC89C52的中断系统十分重要，**C51**编译器允许在**C**语言源程序中声明中断和编写中断服务程序，从而减轻了采用汇编程序编写中断服务程序的繁琐程度。

通过使用**interrupt**关键字来实现。定义中断服务程序的一般格式如下：

void 函数名( ) interrupt n [using m]

using *m*指明该中断服务程序所对应的工作寄存器组，取值范围：**0～3**。指定工作寄存器组的缺点是所有被中断调用的过程都必须使用同一个寄存器组，否则参数传递会发生错误。通常不设定**using *m***。

关键字interrupt后面的*n*是中断号，*n*的取值范围：0～31。编译程序从8*n*+3处产生中断向量，即在程序存储器8*n*+3地址处形成一条长跳转指令，转向中断号*n*的中断服务程序。

中断号n	中断源	中断向量
0	外部中断0	0003H
1	定时器0	000BH
2	外部中断1	0013H
3	定时器1	001BH
4	串行口	0023H
5	定时器2	002BH
6	外部中断2	0033H
7	外部中断3	003BH

本讲重难点

1、C51规定变量必须先定义后使用。C51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

2、扩展的四种数据类型：bit、sfr、sfr16、sbit

3、存储器类型指明该变量所处的单片机的内存空间。分别为data、bdata、idata、pdata、xdata和code。

4、Keil和Proteus的联合调试。