项目3

单片机控制数码管的显示

项目目标

（1）LED 数码管的结构和控制方法。
（2）一维数组的应用。
（3）LED 数码管显示 0～9 的工作原理。
（4）单片机与数码管的接口技术。
（5）数码管的静态显示和动态显示。
（6）使用 Keil C51 软件编写控制程序。
（7）使用 Proteus 进行电路设计和实施仿真。

任务 3.1　单数码管轮流显示 0～9 十个数

任务目标

（1）LED 数码管的结构和控制方法。
（2）一维数组的应用。
（3）LED 数码管显示 0～9 的工作原理。
（4）单片机与数码管的接口技术。

任务需求

通过编写控制程序，并将程序烧写到单片机芯片中，使一个七段数码管开始显示 0，然后依次显示 1、2、3、4、5、6、7、8、9 九个数字，每个数字显示之间要有足够的时间间隔，以便显示清晰，在显示一遍之后继续从 0 开始循环到 9。

任务准备

所用元件见表 3.1。

表 3.1　元件列表

元件名称	Proteus 元件库关键字对照	元件规格
单片机	AT89C51	
瓷片电容	CAP	22pF

续表

元件名称	Proteus 元件库关键字对照	元件规格
电解电容	CAP-ELEC	22μF
晶振	CRYSTAL	12MHz
电阻	RES＊	1kΩ
组排	RESPACK-7	200Ω
共阴极七段数码管	7SEG-COM-CAT	无小数点

3.1.1　硬件知识

1. LED 数码管的结构

LED 数码管是单片机人机对话的一种常用的输出设备，在今后的学习实验中经常用到。LED 数码管能够显示单片机系统的工作状态、运算结果及其他一些信息。

单个数码管的外形图如图 3.1 所示，单个数码管的引脚图如图 3.2 所示。常用的显示 0～9 十个数的单个数码管由 8 个 LED 发光二极管组成(包括一个小数点)“.”，把每个发光二极管叫一个数码段。

2. LED 数码管的种类

LED 数码管按连接结构分为共阴极和共阳极两种。

(1) 共阳极数码管的内部结构如图 3.3(a)所示，8 个发光二极管的阳极(正极)连接在一起，作为公共端(com)，公共端接高电平。每个发光二极管的阴极(负极)是独立的，

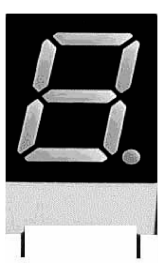

图 3.1　数码管外形图(有小数点)

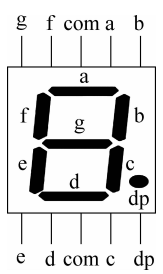

图 3.2　数码管引脚图(有小数点)

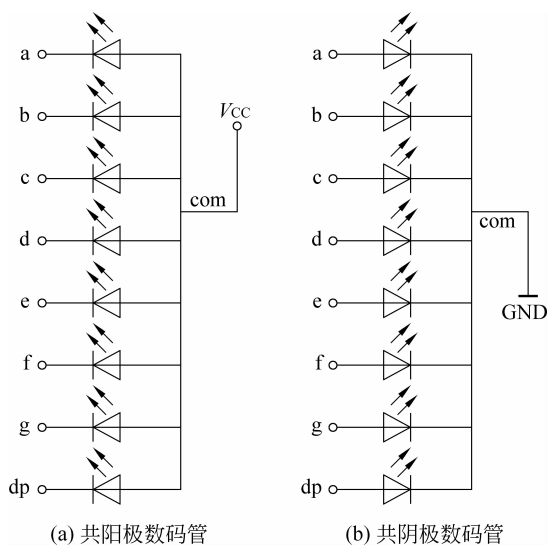

图 3.3　数码管的内部结构

作为每个数码段的控制端。如果某个控制端加低电平信号，则这个数码段导通，发光二极管亮。不同位置的数码管亮起能表示不同的数字。如想显示数字 1，只要保证 b、c 段导通，即 b、c 段的控制端接低电平即可。如果想要表示一个数字 8 并且小数点也点亮，就将 8 个控制端都加低电平，保证 8 个数码段都导通即可。

(2) 共阴极数码管内部结构如图 3.3(b)所示，8 个发光二极管的阴极(负极)连接在一起，作为公共端(com)，公共端接低电平，每个发光二极管的阳极(正极)是独立的，作为每个数码段的控制端。如果某个控制端加高电平，则这个数码段导通，发光二极管亮。不同位置的数码管亮起能表示不同的数字。

对某个数码管中的各数码段加电压称为段选，所有数码段的段选组合在一起称为段选码，也称为字型码，字型码可以根据显示字符的形状和各段的顺序得出。

3. LED 数码管的字型码编码表

之前的知识中提到过，如果想显示数字 1 只要让 b、c 数码段亮起就行，如果想显示数字 8 只要让所有的数码段都亮起就可以。如果想显示数字，只要搞清楚哪几段数码段亮起，然后在相应的引脚加上正确的电平就可以了。也就是由单片机向数码管的控制端输出相应的字型码即可。

将单片机 P1 口的 P1.1、P1.2、P1.3、…、P1.7 七个引脚依次与数码管的 a、b、…、f、dp 七个段控制引脚相连接。如果使用的是共阳极数码管，com 端接＋5V，要显示数字 1 则数码管的 b、c 两个段应亮起，其他段熄灭，需向 P1 口传送数据 11111001(F9H)，该数据就是与数字 1 相对应的共阳极字型编码。若共阴极的数码管 com 接地，要显示数字 1，则数码管的 b、c 两个段应亮起，其他段熄灭，需向 P1 口传送数据 00000110(06H)，这就是数字 1 的共阴极字型码。

表 3.2 分别列出共阳极和共阴极数码管显示字符 0～9 所对应的字型码。

表 3.2　数码管字形编码

显示字符	共阳极数码管									共阴极数码管								
	dp	g	f	e	d	c	b	a	字型码	dp	g	f	e	d	c	b	a	字型码
0	1	1	0	0	0	0	0	0	C0H	0	0	1	1	1	1	1	1	3FH
1	1	1	1	1	1	1	0	1	F9H	0	0	0	0	0	1	1	0	06H
2	1	0	1	0	0	1	0	0	A4H	0	1	0	1	1	0	1	1	5BH
3	1	0	1	1	0	0	0	0	B0H	0	1	0	0	1	1	1	1	4FH
4	1	0	0	1	1	0	0	1	99H	0	1	1	0	0	1	1	0	66H
5	1	0	0	1	0	0	1	0	9ZH	0	1	1	0	1	1	0	1	6DH
6	1	0	0	0	0	0	1	0	8ZH	0	1	1	1	1	1	0	1	7DH
7	1	1	1	1	1	0	0	0	F8H	0	0	0	0	0	1	1	1	07H
8	1	0	0	0	0	0	0	0	80H	0	1	1	1	1	1	1	1	7FH
9	1	0	0	1	0	0	0	0	90H	0	1	1	0	1	1	1	1	6FH
.	0	1	1	1	1	1	1	1	7FH	1	0	0	0	0	0	0	0	80H
熄灭	1	1	1	1	1	1	1	1	FFH	0	0	0	0	0	0	0	0	00H

4. 排阻的应用

排阻就是一排电阻的简称，是若干个参数完全相同的电阻，它们的一个引脚都连到一起，作为公共引脚，其余引脚正常引出。所以如果一个排阻是由 n 个电阻构成的，那么它就有 $n+1$ 只引脚。一般来说，最左边的那个是公共引脚，它在排阻上一般用一个色点标出来。

排阻一般应用在数字电路上，比如作为某个并行口的上拉或者下拉电阻用。使用排阻比用若干只固定电阻更方便，排阻具有方向性，与色环电阻相比具有整齐、少占空间的优点。

在本任务中，排阻作为 P0 口的上拉电阻使用，其中排阻的公共一端接供电电压，其他引脚分别与 P0 口的每个引脚相连。Proteus 中排阻的符号为 RP。

3.1.2　C51 语言知识

1. 数组的应用

(1) 一维数组的定义

数组是有序数据的集合。数组中的每一个元素都属于同一个数据类型。数组用统一的数组名和下标来唯一地确定数组中的元素。

定义方式如下：

```
类型说明符　数组名[常量表达式]
```

类型说明符说明定义的是什么数据类型的数组，即数组内的元素是什么数据类型。数组名的命名规则和 C 语言中标识符的命名规则相同。常量表达式表明数组内元素的个数，即数组的长度。例如：

```
int b[10];
```

其中，int 为类型说明符，说明定义的数组为整型数组。b 为数组名，[10]说明该数组有 10 个元素，数组长度为 10。数组元素的下标从 0 到 9 分别为 b[0]、b[1]、b[2]、b[3]、…、b[9]，都为整数。例如：

```
code seven_seg[10];
```

其中，code 关键字是类型说明符，说明数组 seven_seg 是编码数组，code 表示是编码类型。数组的长度为 10，数组元素分别为 seven_seg[0]、seven_seg[1]、seven_seg[2]、…、seven_seg[9]。

数组的定义有以下几点注意事项。

① 数组名不可以和其他的变量名相同。例如：

```
int stu;
int stu[10];                    //这样重名的定义是错误的
```

② 同一说明语句中可以同时说明多个变量和数组。例如：

```
char a,b,stu[5],num[10];
```

(2) 一维数组的初始化

① 一维数组可以在定义的时候赋值,也可以先定义再赋值。例如:

```
int b[10]={0,1,2,3,4,5,6,7,8,9};
```

经过上面的定义和初始化后数组 b 中的元素为 b[0]=0、b[1]=1、b[2]=2、b[3]=3、…、b[9]=9。赋值就是将大括号内的数值按照从前到后的顺序和数组元素一一对应赋予初始值。例如:

```
code seven_seg[10]={0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90}
```

② 可以给全部的数组元素赋值,也可以只给一部分元素赋值。例如:

```
int b[10]={0,1,2,3,4,5}                    //这表示只给前 6 个元素赋值,后 4 个元素的值为 0
```

(3) 一维数组元素的引用

数组元素的表示与引用形式如下:

```
数组名[下标]
```

数组要先定义后使用,每次能够引用一个数组元素,但是不能引用整个数组。例如:

```
main()
{
    int b[10]={0,1,2,3,4,5,6,7,8,9};
    printf("%d",b[6]);                     //输出数组中第 6 个元素的值
}
```

2. 循环语句

(1) while 语句

while 语句的一般形式如下:

```
while(表达式)
{
    循环体
}
```

其中表达式是循环条件。

while 语句的语义是:计算表达式的值,当值为真(非 0)时,执行循环体语句。其执行过程可用图 3.4 表示。

例如,用 while 语句求 1～10 的和的代码如下。

```
main()
{
    int i,sum=0;
    i=1;
    while(i<=10)
    {
        sum=sum+i;
```

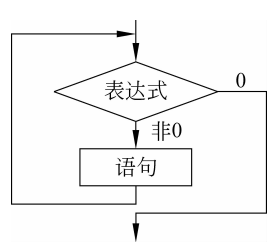

图 3.4 while 语句流程图

```
            i++;
        }
    }
```

在本项目中 while 循环语句为 while(1)，这是一条死循环命令。程序执行成功后数码管会一直循环显示 0～9 十个数字，指导人们终止程序的调试或仿真模式，在实际应用中，可以通过调节 while 语句中的循环条件控制循环执行过程。

(2) for 语句

在 C 语言中，for 语句使用最为灵活，它完全可以取代 while 语句。它的一般形式如下：

```
for(表达式 1;表达式 2;表达式 3)
{
    循环体
}
```

它的执行过程如下。

① 先求解表达式 1。

② 求解表达式 2，若其值为真(非 0)，则执行 for 语句中指定的内嵌语句，然后执行③；若其值为假(0)，则结束循环，转到⑤。

③ 求解表达式 3。

④ 转回②继续执行。

⑤ 循环结束，执行 for 语句下面的一个语句。

for 语句最简单的应用形式也是最容易理解的形式如下：

```
for(循环变量赋初值;循环条件;循环变量增量)
{
    循环体
}
```

循环变量赋初值总是一个赋值语句，它用来给循环控制变量赋初值；循环条件是一关系表达式，它决定什么时候退出循环；循环变量增量，定义循环控制变量每循环一次后按什么方式变化。这 3 个部分之间用“;”分开。

例如本项目控制程序中的语句如下：

```
for(i=0;i<10;i++)
{
    P0=seven_seg[i];            //将 i 数值对应的编码送至 P0 口输出
    delay(240);                 //执行延时子函数
}
```

循环变量为 i，初始值为 0，循环结束的条件是当 i≥10 时，每次循环结束后变量 i 的值自动加 1。

3. 函数声明与调用

(1) 函数调用的一般形式

C 语言程序中是通过对函数的调用来执行函数体的，其过程与其他语言的子程序调

用相似。

C 语言中，函数调用的一般形式如下：

函数名(实际参数表)；

对无参函数调用时则无实际参数表。实际参数表中的参数可以是常数、变量或其他构造类型数据及表达式。各实参之间用逗号分隔。

(2) 被调用函数的声明。在主调函数中调用某函数之前应对该被调函数进行说明(声明)，这与使用变量之前要先进行变量说明是一样的。在主调函数中对被调函数做说明的目的是使编译系统知道被调函数返回值的类型，以便在主调函数中按此种类型对返回值做相应的处理。

其一般形式如下：

类型说明符 被调函数名(类型 形参 1,类型 形参 2,...,类型 形参 n)；

或

类型说明符 被调函数名(类型 1,类型 2,...,类型 n)；

括号内给出了形参的类型和形参名，或只给出形参类型。这便于编译系统进行检错，以防止可能出现的错误。

在本项目的代码中，也用到了函数的声明和调用。

3.1.3 任务实施

1. 硬件电路设计

整个项目的电路图设计都是在 ISIS 编辑区中完成的。

(1) 新建文档。打开 ISIS 6 Professional，单击工具栏上的“新建”按钮，新建一个设计文档。

(2) 选取元件。项目需要的元件见表 3.1。单击 P 按钮，弹出“选取元件”对话框，在此对话框左上角 keywords(关键词)一栏中输入元件名称 AT89C51，系统在对象库中进行搜索查找，并将与关键词匹配的元件显示在 Results 中。在 Results 栏的列表项中，选择 AT89C51 选项，则可将 AT89C51 添加至对象选择器窗口。按照此方法完成其他元件的选取，如果忘记关键词的完整写法，可以用“*”代替，如“CRY*”可以找到晶振。将被选取的元件都加入 ISIS 对象选择器中。

(3) 放置元件至图形编辑窗口。在对象选择器窗口中选中 AT89C51，将鼠标置于图形编辑窗口中该对象欲放置的位置，单击鼠标左键，该对象被完成放置。同理，将 BUTTON、RES 等放置到图形编辑窗口中。

若元件方向需要调整，先在 ISIS 对象选择器窗口中单击选中该元件，再单击工具栏上相应的转向按钮，把元件旋转到合适的方向后再将其放置于图形编辑窗口。

若对象位置需要移动，将鼠标指针移到该对象上右击，此时该对象的颜色变至红

色，表明该对象已被选中，按住鼠标左键拖动对象至新位置后，松开鼠标左键，完成移动操作。

若要解除选定，只须在编辑区的任意空白处右击。若需删除编辑区中的元件，只须用鼠标右键双击元件。

通过一系列的移动、旋转、放置等操作，将元件放在 ISIS 编辑窗口中合适的位置。

(4) 放置终端(电源、地)。放置电源操作：单击工具栏中的“终端”按钮，在对象选择器窗口中选择 POWER，再在编辑区中要放电源的位置单击即可。放置地(GROUND)的操作与此类似。

(5) 元件之间的连线。Proteus 的智能化可以在使用者想要画线的时候进行自动检测。下面来操作将电阻 R_1 的上端连接到电容 C_3 的下端。当鼠标指针靠近 R_1 右端的连接点时，跟着鼠标指针就会出现一个“×”图标，表明找到了 R_1 的连接点。此时单击。然后移动鼠标指针，靠近电容 C_3 的下端连接点时，跟着鼠标指针就会出现一个“×”图标，表明找到了连接点，单击完成电阻 R_1 和 C_3 的连线。

Proteus 具有线路自动路径功能(简称 WAR)，当选中两个连接点后，WAR 将选择一个合适的路径连线。WAR 可通过使用标准工具栏里的 WAR 命令按钮来关闭或打开，也可以在菜单栏的 Tools 下找到这个图标。

同理，可以完成其他连线。在此过程的任何时刻，都可以按 ESC 键或者右击来放弃画线，想要删除一条连线的时候只需要对准连线双击鼠标右键即可。

(6) 修改、设置元器件的属性。Proteus 库中的元件都有相应的属性，要设置修改元件的属性，只须右击 ISIS 编辑区中的该元件，再单击。例如，对于电阻 R_1，先右击弹出属性窗口，在窗口中已经将电阻的阻值修改为 1kΩ。图 3.5 是编辑完成的单数码管轮流显示 0～9 十个数的电路。

2. 软件设计

(1) 程序分析

流程图如图 3.6 所示。

(2) 程序设计

```
#include<reg51.h>
Unsigned char code seven_seg[10]={0X3F,0X06,0X5B,0X4F,0X66,0X6D,0X7D,0X07,0X7F,
                                  0X6F};          //定义数字编码数组
void delay(unsigned char i);                      //声明延迟子函数
void main()
{
    unsigned char i;                              //定义无符号字符型变量
    while(1)
    {
        for(i=0;i<10;i++)
        {
            P0=seven_seg[i];                      //将 i 数值对应的编码送至 P0 口输出
            delay(240);                           //执行延时子函数
```

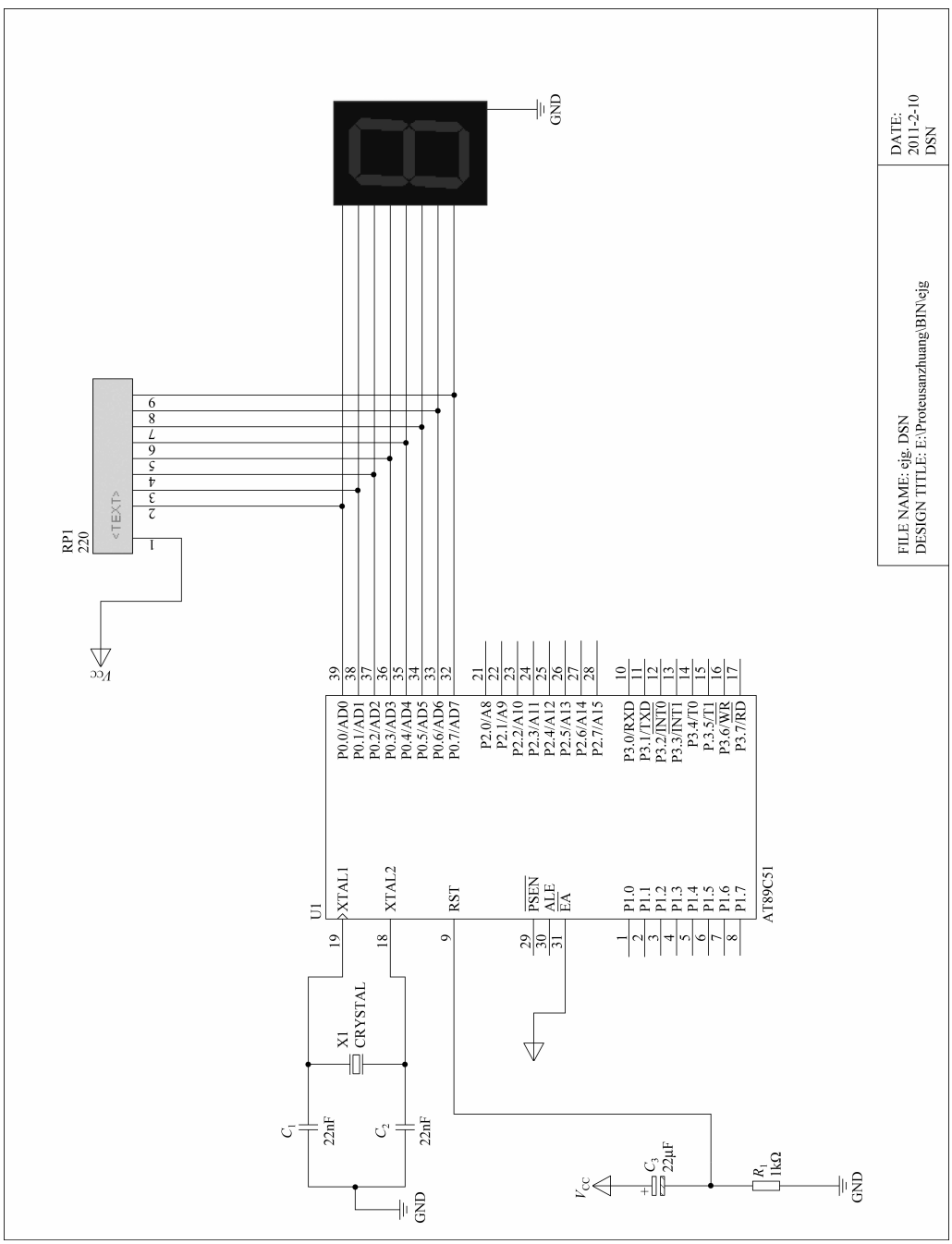

图 3.5 单数码管轮流显示 0～9 十个数电路原理图(Proteus 绘制)

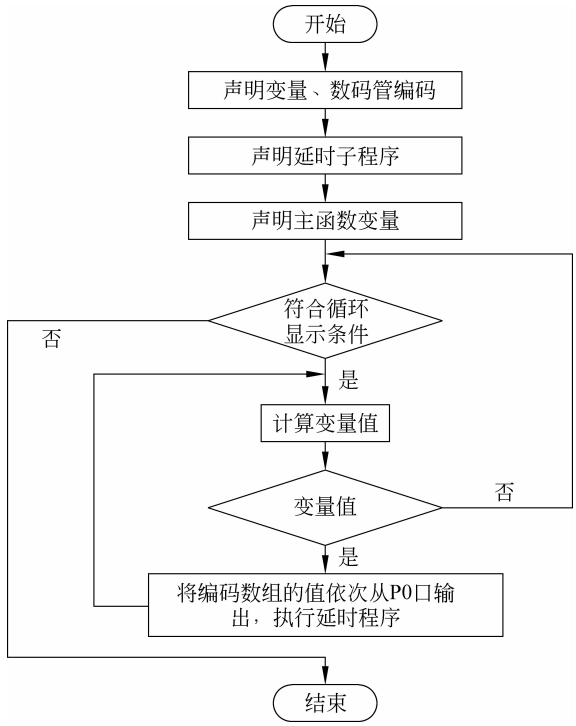

图 3.6　单数码管显示 0～9 源程序流程图

```
            }
        }
}
void delay(unsigned char i)
{
    unsigned char j,k;
    for(k=0;k<i;k++)
      for(j=0;j<255;j++);                    //利用双循环实现延时
}
```

任务 3.2　四数码管动态显示 0～9 十个数

任务目标

(1) 数码管的静态显示与动态显示。
(2) 进一步巩固一维数组的应用。
(3) 使用 Keil C51 软件编写控制程序。
(4) 使用 Proteus 进行电路设计和实施仿真。

任务需求

通过编写控制程序，并将程序烧写到单片机芯片中，使 4 个七段数码管动态显示 0，