



全国优秀教材特等奖



“十四五”职业教育国家规划教材



“十三五”职业教育国家规划教材

国家精品课
配套教材

高等院校“互联网+”系列精品教材

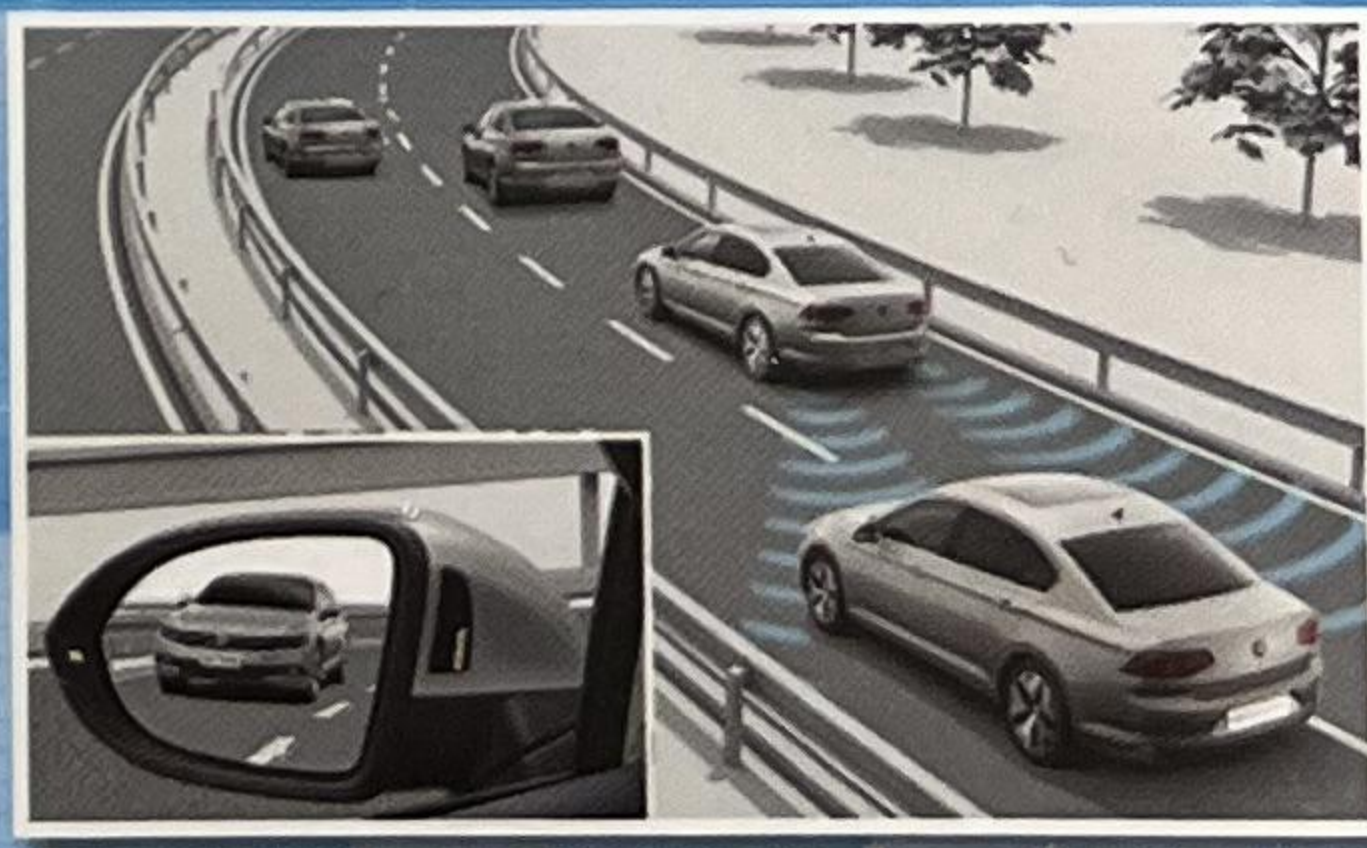
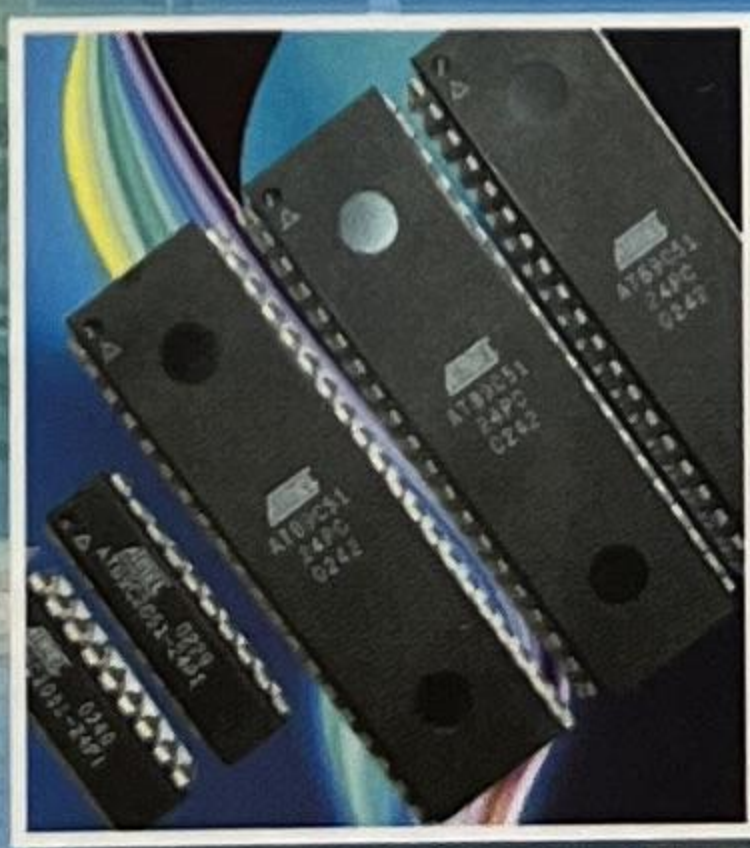
单片机应用技术

(C语言版) 第4版

◎ 王静霞 主编 ◎ 杨宏丽 刘 俐 副主编

★ 适用于应用型本科和高职高专院校等

- 真正的项目化教学经典教材
- 经4次修订内容更加科学合理
- 由国家资源库项目负责人主编
- 提供数百个二维码教学资源



今日努力，将成就明日梦想，加油！

扫一扫
书中二维码
看更多微课视频资源



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

参赛内容相关章节：



单片机应用系统经常需要连接一些外部设备，其中显示器和键盘是构成人机对话的一种基本方式，使用最为频繁。本项目将介绍常用的显示器件、键盘工作原理以及它们如何与单片机接口，如何相互传送信息等技术。

教学导航

知识重点	1. LED 数码管显示和接口； 3. LCD 液晶显示和接口； 5. 矩阵式按键接口	2. LED 大屏幕显示和接口； 4. 独立式按键接口；
知识难点	1. LED 动态显示接口； 3. LCD 液晶显示和接口；	2. LED 点阵大屏幕显示接口； 4. 矩阵式按键接口
推荐教学方式	从工作任务入手，让学生逐步熟悉各种显示器件和键盘的工作原理、接口及编程方法	
建议学时	12 学时	
推荐学习方法	1. 从简单任务入手，以发光二极管的发光控制为起点，再扩展到 8 个连在一起的发光二极管即数码管。学习数码管时可以先回忆发光二极管的控制，学习数码管接口控制时可以先接 1 个数码管再扩展到多个数码管。 2. 类比法，LED 数码管的动态显示和 LED 大屏幕显示的原理相似，可以比较学习	
必须掌握的理论知识	1. LED 数码管显示接口； 3. 独立式按键接口与矩阵式按键接口	2. LED 点阵大屏幕显示接口；
必须掌握的技能	1. LED 数码管显示控制； 3. 独立式按键接口和矩阵式按键接口	2. LED 大屏幕显示器的制作与调试；

任务 4-1 8 路抢答器设计

1. 目的与要求

通过对具有 8 个按键输入和 1 个数码管显示的抢答器的设计与制作, 让读者理解 C 语言中数组的基本概念和应用技术, 初步了解单片机与 LED 数码管的接口电路设计及编程控制方法。

系统要求用 8 个独立式按键作为抢答输入按键, 序号分别为 0~7, 当某一参赛者首先按下抢答按钮时, 在数码管上显示抢答成功的参赛者的序号, 此时抢答器不再接受其他输入信号, 直到按下系统复位按钮, 系统再次接受下一轮的抢答输入。

2. 电路设计

根据任务要求, 用一位共阳极 LED 数码管作为显示器件, 显示抢答器的状态信息, 数码管采用静态连接方式与单片机的 P1 口连接; 8 个按键连接到 P0 口, 将与 P0.0 引脚连接的按键 S_0 作为“0”号抢答输入, 与 P0.1 引脚连接的按键 S_1 为“1”号抢答输入, 依次类推。抢答器电路如图 4.1 所示。

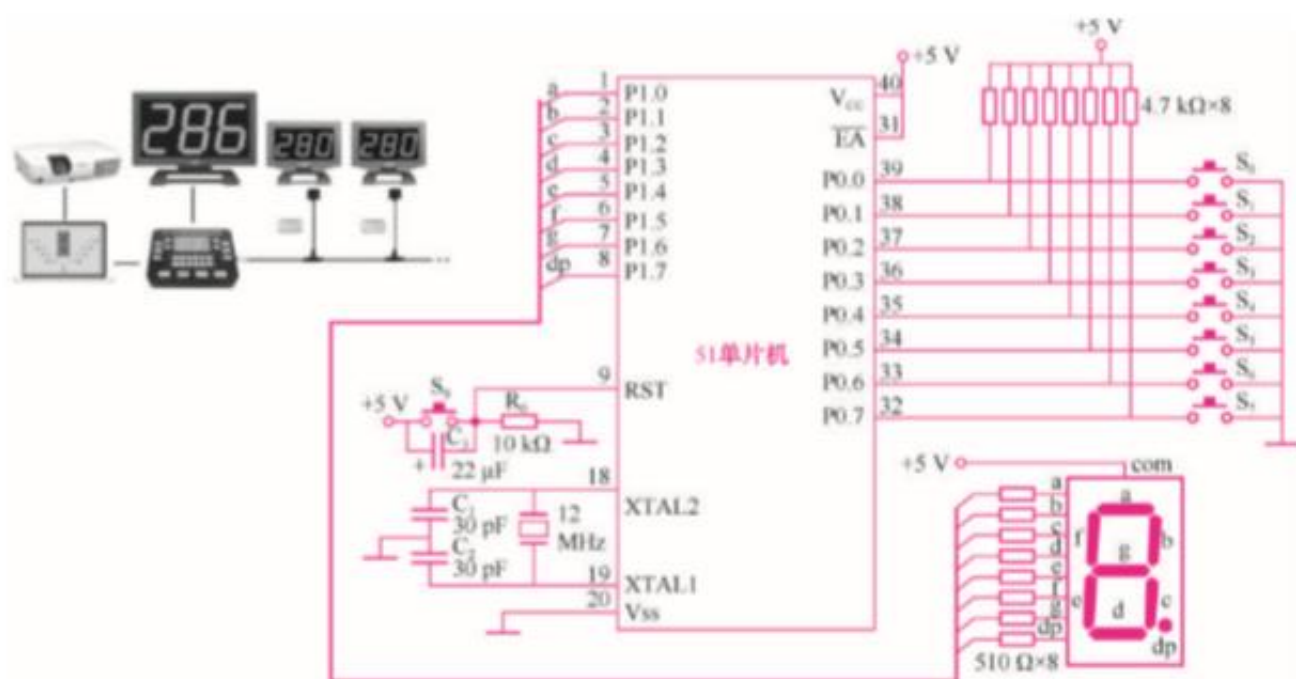


图 4.1 8 路抢答器电路



小知识 (1) 在单片机系统中, 经常采用 LED 数码管来显示单片机系统的工作状态、运算结果等各种信息, LED 数码管是单片机人机对话的一种重要输出设备。

(2) LED 数码管由 8 个发光二极管 (以下简称段) 构成, 通过不同的发光字段组合可用来显示数字 0~9、字符 A~F、H、L、P、R、U、Y、符号“-”及小数点“.”等。

(3) 因为只控制一个数码管, 选择一直点亮各段的静态显示方式。这种显示方式可在较小的电流驱动下获得较高的显示亮度, 且占用 CPU 时间少, 编程简单, 便于显示和控制。

(4) 按键采用独立式按键接法, 每个按键都单独占用一根 I/O 端口线, 适用于按键数目比较少的应用场合, 优点是软件结构简单。

(5) 电路中 P0 口外接的上拉电阻是保证按键断开时, I/O 端口为高电平; 按键按下时相应端口为低电平。

3. 源程序设计

程序设计思路: 系统上电时, 数码管显示 “-”, 表示开始抢答, 当记录到最先按下的按键序号后, 数码管将显示该参赛者的序号, 同时无法再接受其他按键的输入; 当系统按下复位按钮 S₁ 时, 系统显示 “-”, 表示可以接受新一轮的抢答。

```
//程序: ex4_1.c
//功能: 8 路抢答器控制程序
#include<reg51.h>          //包含头文件 reg51.h, 定义 51 单片机的专用寄存器
void delay (unsigned int i) ;//延时函数声明
void main ( )              //主函数
{
    unsigned char button;    //保存按键信息
    unsigned char code disp[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0xbf};
    //定义数组 disp, 依次存储包括 0~7 和 “-” 的共阳极数码管显示码
    P0=0xff;                 //读引脚状态, 需先置 1
    P1=disp[8];              //显示 “-”
    while (1)
    {
        button=P0;          //第一次读按键状态
        delay (1200) ;      //延时消抖
        button=P0;          //第二次读按键状态
        switch (button)     //根据按键的值进行多分支跳转
        {
            case 0xfe:P1=disp[0];delay (10000) ;while (1) ;break;
            //0 按下, 显示 0, 待机
            case 0xfd:P1=disp[1];delay (10000) ;while (1) ;break;
            //1 按下, 显示 1, 待机
            case 0xfb:P1=disp[2];delay (10000) ;while (1) ;break;
            //2 按下, 显示 2, 待机
            case 0xf7:P1=disp[3];delay (10000) ;while (1) ;break;
            //3 按下, 显示 3, 待机
            case 0xef:P1=disp[4];delay (10000) ;while (1) ;break;
            //4 按下, 显示 4, 待机
            case 0xdf:P1=disp[5];delay (10000) ;while (1) ;break;
            //5 按下, 显示 5, 待机
            case 0xbf:P1=disp[6];delay (10000) ;while (1) ;break;
            //6 按下, 显示 6, 待机
            case 0x7f:P1=disp[7];delay (10000) ;while (1) ;break;
            //7 按下, 显示 7, 待机
```



扫一扫
阅读本
程序代
码

```

        default:
    }
}

void delay (unsigned int i)           //延时函数参见任务 1-2 的 ex1_1.c 程序

```



小知识 在程序设计中，为了处理方便，把具有相同类型的若干数据项按有序的形式组织起来。这些按序排列的同类数据元素的集合称为数组。在上面的程序中，定义了数组 disp[]。

```
unsigned char code disp [] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8,
```

0xbf}; 数组中的元素有固定数目和相同类型，数组元素的数据类型就是该数组的基本类型。上面数组的类型是无符号字符型，数组名为 disp，数组元素个数为 9 个。

数组元素也是一种变量，其标志方法为数组名后跟一个下标，例如：disp[0]、disp[1]……disp[8]。

在这个数组定义语句中，关键字 code 是为了把 disp[] 数组存储在片内程序存储器 ROM 中，该数组与程序代码一起固化在程序存储器中，关于关键字 code 的具体说明参见第 3.3.2 节。

4. 任务小结

通过 8 路抢答器的设计和制作，让读者理解 C 语言中数组的应用，并初步了解单片机与 LED 数码管的接口电路设计及编程控制方法。

5. 举一反三

设计具有 4 个按键输入和 1 个数码管显示的简易密码锁。

在一些智能门控管理系统中，需要输入正确的密码才可以开锁。基于单片机控制的密码锁硬件电路包括三部分：按键输入、数码显示和电控开锁驱动电路，三者状态的对应关系如表 4.1 所示。

密码锁硬件电路设计：4 个按键，由 P0 端口的 P0.0~P0.3 控制；一个数码管，由 P1 口静态控制；一个发光二极管，由 P3.0 引脚控制，发光二极管的亮灭分别模拟开锁电路的打开和锁定。请读者自行画出简易密码锁硬件电路图。

简易密码锁的基本功能如下：(1) 4 个按键，分别代表数字 0、1、2、3；(2) 密码在程序中事先设定，为 0~3 之间的数字；(3) 数码管显示“-”，表示等待密码输入；(4) 密码输入正确时显示字符“P”约 3 s，并通过 P3.0 端口将锁打开；否则显示字符“E”约 3 s，继续保持锁定状态。

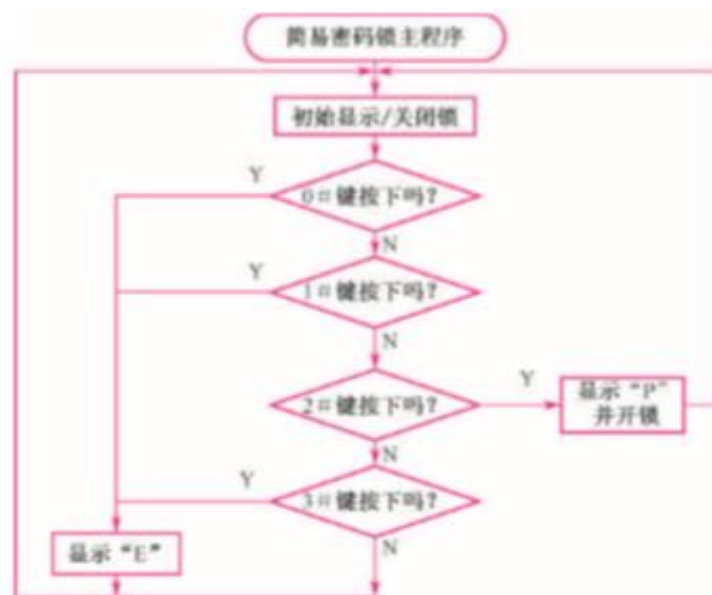
程序设计时，设初始密码锁关闭，显示符号为“-”。当按下数字键后，若与预先设定的密码相同则显示“P”，打开锁，过 3 s 后恢复锁定状态，等待下一次密码输入；否则显示“E”持续 3 s，保持锁定状态并等待下一次密码输入。采用共阳极 LED 数码管静态显示方式，密码设定为“2”。简易密码锁控制程序的流程如图 4.2 (b) 所示，其参考源程序如下。

表 4.1 简易密码锁状态

按键输入状态	数码管 显示信息	锁驱动 状态
无密码输入	-	锁定
输入与设定密码相同	P	打开
输入与设定密码不同	E	锁定



(a) 常见密码锁



(b) 控制程序流程

图 4.2 密码锁示例和简易密码锁控制程序流程

```

//程序: ex4_2.c
//功能: 简易密码锁控制程序
#include<reg51.h> //包含头文件 reg51.h, 定义 51 单片机的专用寄存器
sbit P3_0=P3^0; //控制开锁, 用发光二极管代替
void delay (unsigned char i) //延时函数声明
void main ( ) //主函数
{
    unsigned char button; //保存按键信息
    unsigned char code tab[7]={0xc0,0xf9,0xa4,0xb0,0xbf,0x86,0x8c};
    //定义显示段码表, 分别对应显示字符: 0、1、2、3、-、E、P
    P0=0xff; //读 P0 口引脚状态, 需先置全 1
    while (1)
    {
        P1=tab[4]; //密码锁的初始显示状态 "-"
        P3_0=1; //设置密码锁初始状态为 "锁定", 发光二极管熄灭
        button=P0; //读取 P0 口上的按键状态并赋值到变量 button
        delay (1200); //延时去抖
        button=P0; //再次读入按键状态
        button &= 0x0f; //采用与操作保留低 4 位的按键状态, 其他位清零
        switch (button) //判断按键的键值
        {
            case 0x0e: P1=tab[0]; delay (10000); P1=tab[5]; delay (50000); break;
            //0#键按下, 密码输入错误, 显示 "E"
            case 0x0d: P1=tab[1]; delay (10000); P1=tab[5]; delay (50000); break;
            //1#键按下, 密码输入错误, 显示 "E"
            case 0x0b: P1=tab[2]; delay (10000); P1=tab[6]; P3_0=0; delay (50000);
            break;
            //2#键按下, 密码正确, 开锁并显示 "P"

```



扫一扫
阅读本
程序代
码

```

        case 0x07:P1=tab[3];delay(10000);P1=tab[5];delay(50000);break;
        //3#键按下,密码输入错误,显示“E”
    }
}
}
void delay(unsigned char i) //延时函数参见任务1-2的 ex1_1.c 程序

```

小问答

问：在程序中用4个独立式按键分别代表0、1、2、3，只有4种密码选择，显然不够安全，是否可以用4个按键组成更多的密码选择呢？

答：能。若用4个按键的输入状态分别代表4位二进制数，可组成16种密码，范围为0 000~1 111；若再增加对按键输入顺序的判断还能组成更多种密码选择，但需要修改密码的设置与识别程序才能实现。

4.1 认识LED数码管

知识分布网络



4.1.1 LED 数码管的结构

在单片机应用系统中，LED 数码管是单片机人机对话的一种重要输出设备，经常用来显示单片机应用系统的工作状态、运算结果等各种信息。

单个 LED 数码管的外形和外部引脚如图 4.3 所示。LED 数码管由 8 个发光二极管（以下简称段）构成，通过不同的发光段组合可用来显示数字 0~9、字符 A~F、H、L、P、R、U、Y、符号“-”及小数点“.”等信息。

1. LED 数码管的工作原理

LED 数码管可分为共阳极和共阴极两种结构。



扫一扫看
LED数码
管结构演
示文稿



扫一扫看
LED数码
管结构教
学视频

(1) 共阳极数码管的内部结构如图 4.4 (a) 所示，8 个发光二极管的阳极连接在一起，作为公共控制端（com），接高电平。阴极作为“段”控制端，当某段控制端为低电平时，该段对应的发光二极管导通并点亮。通过点亮不同的段，显示出不同的字符。如显示数字 1 时，b、c 两端接低电平，其他各端接高电平。

(2) 共阴极数码管的内部结构如图 4.4 (b) 所示。8 个发光二极管的阴极连接在一起，作为公共控制端（com），接低电平。阳极作为“段”控制端，当某段控制端为高电平时，该段对应的发光二极管导通并点亮。

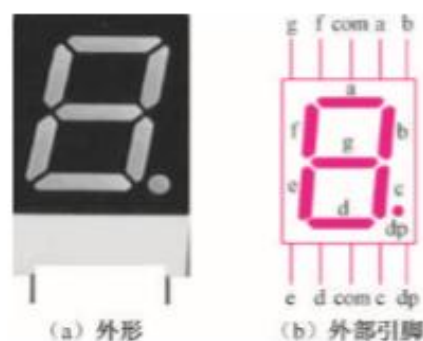


图 4.3 LED 数码管

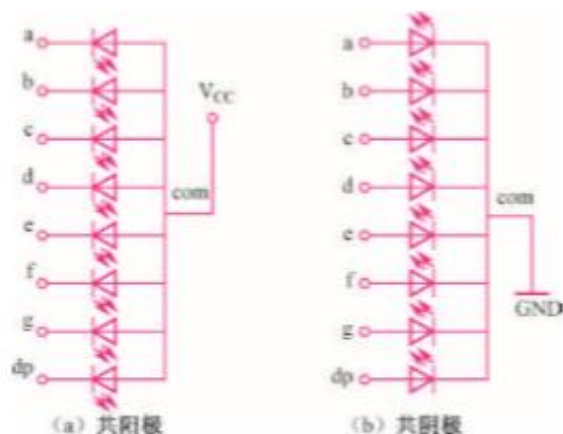


图 4.4 数码管内部结构

小问答

问：如何判断数码管的结构是共阳极还是共阴极？如何用指针式万用表测试数码管的极性及好坏？

答：根据图 4.4，通过判断任意段与公共端连接的二极管的极性就可以判断出是共阳极还是共阴极数码管。

首先将指针式万用表放置在电阻测量方式上，假设数码管是共阳极的，那么将指针式万用表的表内电源正极（黑表笔）与数码管的 com 端相接，然后用指针式万用表的表内电源负极（红表笔）逐个接触数码管的各段，数码管的各段将逐个点亮，则数码管是共阳极的；如果数码管的段均不亮，则说明数码管是共阴极的。也可将指针式万用表的红黑表笔交换连接后测试。如果数码管只有部分段点亮，而另一部分不亮，说明数码管已经损坏。

2. LED 数码管字型编码

从任务 4-1 中我们知道，若将数值 0 送至单片机的 P1 口，数码管上不会显示数字“0”。要使数码管显示出数字或字符，直接将相应的数字或字符送至数码管的段控制端是不行的，必须使段控制端输出相应的字型编码。

将单片机 P1 口的 P1.0、P1.1、…、P1.7 八个引脚依次与数码管的 a、b、…、g、dp 八个段控制引脚相连接。如果使用的是共阳极数码管，com 端接 +5 V。要在共阳极数码管上显示数字“0”，则需要将数码管的 a、b、c、d、e、f 六个段点亮，其他段熄灭，需要向 P1 口传送数据 11000000B (0xC0)，该数据就是与字符 0 相对应的共阳极字型编码。如果使用的是共阴极数码管，com 端接地。要用共阴极数码管显示数字 1，就要把数码管的 b、c 两段点亮，其他段熄灭，因此，要向 P1 口传送数据 00000110B (0x06)，这就是字符 1 的共阴极字型编码了。

表 4.2 中分别列出共阳、共阴极数码管的显示字型编码。

表 4.2 数码管字型编码

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	0xC0	0	0	1	1	1	1	1	1	0x3F
1	1	1	1	1	1	0	0	1	0xF9	0	0	0	0	0	1	1	0	0x06

显示 字符	共阳极数码管									共阴极数码管								
	d _p	g	f	e	d	c	b	a	字型码	d _p	g	f	e	d	c	b	a	字型码
2	1	0	1	0	0	1	0	0	0xA4	0	1	0	1	1	0	1	1	0x5B
3	1	0	1	1	0	0	0	0	0xB0	0	1	0	0	1	1	1	1	0x4F
4	1	0	0	1	1	0	0	1	0x99	0	1	1	0	0	1	1	0	0x66
5	1	0	0	1	0	0	1	0	0x92	0	1	1	0	1	1	0	1	0x6D
6	1	0	0	0	0	0	1	0	0x82	0	1	1	1	1	1	0	1	0x7D
7	1	1	1	1	1	0	0	0	0xF8	0	0	0	0	0	1	1	1	0x07
8	1	0	0	0	0	0	0	0	0x80	0	1	1	1	1	1	1	1	0x7F
9	1	0	0	1	0	0	0	0	0x90	0	1	1	0	1	1	1	1	0x6F
A	1	0	0	0	1	0	0	0	0x88	0	1	1	1	0	1	1	1	0x77
B	1	0	0	0	0	0	1	1	0x83	0	1	1	1	1	1	0	0	0x7C
C	1	1	0	0	0	1	1	0	0xC6	0	0	1	1	1	0	0	1	0x39
D	1	0	1	0	0	0	0	1	0xA1	0	1	0	1	1	1	1	0	0x5E
E	1	0	0	0	0	1	1	0	0x86	0	1	1	1	1	0	0	1	0x79
F	1	0	0	0	1	1	1	0	0x8E	0	1	1	1	0	0	0	1	0x71
H	1	0	0	0	1	0	0	1	0x89	0	1	1	1	0	1	1	0	0x76
L	1	1	0	0	0	1	1	1	0xC7	0	0	1	1	1	0	0	0	0x38
P	1	0	0	0	1	1	0	0	0x8C	0	1	1	1	0	0	1	1	0x73
R	1	1	0	0	1	1	1	0	0xCE	0	0	1	1	0	0	0	1	0x31
U	1	1	0	0	0	0	0	1	0xC1	0	0	1	1	1	1	1	0	0x3E
Y	1	0	0	1	0	0	0	1	0x91	0	1	1	0	1	1	1	0	0x6E
-	1	0	1	1	1	1	1	1	0xBF	0	1	0	0	0	0	0	0	0x40
-	0	1	1	1	1	1	1	1	0x7F	1	0	0	0	0	0	0	0	0x80
熄灭	1	1	1	1	1	1	1	1	0xFF	0	0	0	0	0	0	0	0	0x00

小问答

问：对于同一个字符，共阳极和共阴极的字型编码之间有什么关系？

答：从表 4.2 中可以看出，当显示字符“1”时，共阳极的字型码为 0xF9，而共阴极的字型码为 0x06，所以对于同一个字符，共阴和共阳码的关系为取反。

4.1.2 LED 数码管静态显示



扫一扫看
LED数码管
静态显示
演示文稿



扫一扫看
LED数码管
静态显示
教学视频

图 4.5 给出了两位共阳极数码管静态显示的接口电路，两个共阳极数码管的段码分别由 P1、P2 口来控制，com 端都接在 +5 V 电源上。

静态显示是指使用数码管显示字符时，数码管的公共端恒定接地（共阴极）或+5 V 电源（共阳极）。将每个数码管的 8 个段控制引脚分别与单片机的一个 8 位 I/O 端口相连接。只要 I/O 端口有显示字型码输出，数码管就显示给定字符，并保持不变，直到 I/O 端口输出新的段码。任务 4-1 采用的就是一位数码管的静态显示方法。



小经验 采用静态显示方式，较小的电流就可获得较高的亮度，且占用 CPU 时间少，编程简单，便于监测和控制。但占用单片机的 I/O 端口线多， n 位数码管的静态显示需占用 $8 \times n$ 个 I/O 端口，所以限制了单片机连接数码管的个数。同时，硬件电路复杂，成本高，因此，数码管静态显示方式适合显示位数较少的场合。

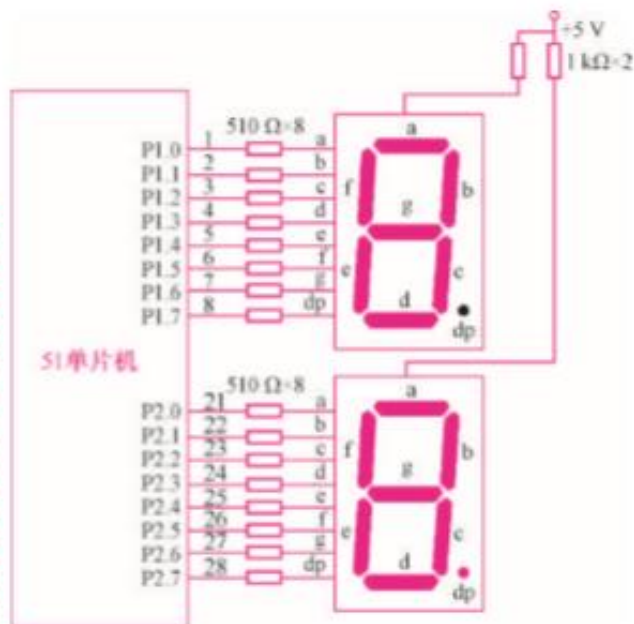


图 4.5 两位数码管静态显示接口电路

4.2 数组的概念

知识分布网络



在程序设计中，为了处理方便，把具有相同类型的若干数据项按有序的形式组织起来。这些按序排列的同类数据元素的集合称为数组。组成数组的各个数据分项称为数组元素。

数组属于常用的数据类型，数组中的元素有固定数目和相同类型，数组元素的数据类型就是该数组的数据类型。例如，整型数据的有序集合称为整型数组，字符型数据的有序集合称为字符型数组。

数组还分为一维、二维、三维和多维数组等，常用的是一维、二维和字符数组。

4.2.1 一维数组

1. 一维数组的定义

在 C 语言中，数组必须先定义、后使用。一维数组的定义格式如下：

类型说明符 数组名 [常量表达式];

类型说明符是指数组中的各个数组元素的数据类型；数组名是用户定义的数组标识符；



扫一扫看
一维数组
定义演示
文档



扫一扫看
一维数组
定义教学
视频

方括号中的常量表达式表示数组元素的个数，也称为数组的长度。

例如：

```
int a[10];           //定义整型数组 a，有 10 个元素，a[0]、a[1]、...、a[9]
float b[10],c[20];   //定义实型数组 b，有 10 个元素，实型数组 c，有 20 个元素
char ch[20];         //定义字符数组 ch，有 20 个元素
```

定义数组时，应注意以下几点：

(1) 数组的类型实际上是指数组元素的取值类型。对于同一个数组，所有元素的数据类型都是相同的。

(2) 数组名的书写规则应符合标识符的书写规定。

(3) 数组名不能与其他变量名相同。

例如，在下面的程序段中，因为变量 `num` 和数组 `num` 同名，程序编译时会出现错误，无法通过：

```
void main ( )
{
    int num;
    float num[100];
    .....
}
```

(4) 方括号中常量表达式表示数组元素的个数，如 `a[5]` 表示数组 `a` 有 5 个元素。数组元素的下标从 0 开始计算，5 个元素分别为 `a[0]`、`a[1]`、`a[2]`、`a[3]`、`a[4]`。

(5) 方括号中的常量表达式不可以是变量，但可以是符号常数或常量表达式。

例如，下面的数组定义是合法的：

```
#define NUM 5        //定义符号常数
main ( )
{
    int a[NUM],b[7+8];
    ...
}
```

但是，下述定义方式是错误的：

```
main ( )
{
    int num=10;       //定义变量 num
    int a[num];
    ...
}
```

(6) 允许在同一个类型说明中，说明多个数组和多个变量，例如：

```
int a,b,c,d,k1[10],k2[20];
```

2. 数组元素

数组元素也是一种变量，其标志方法为数组名后跟一个下标。下标表示该数组元素在数

组中的顺序号，只能为整型常量或整型表达式。如为小数时，C 编译器将自动取整。定义数组元素的一般形式为：

数组名[下标]

例如：tab[5]、num[i+j]、a[i++]都是合法的数组元素。

在程序中不能一次引用整个数组，只能逐个使用数组元素。例如，数组 a 包括 10 个数组元素，累加 10 个数组元素之和，必须使用下面的循环语句逐个累加各数组元素：

```
int a[10],sum,i;
sum=0;
for ( i=0;i<10;i++) sum=sum+a[i];
```

不能用一个语句累加整个数组，下面的写法是错误的：

```
sum=sum+a;
```

3. 数组赋值

给数组赋值的方法有赋值语句和初始化赋值两种。

在程序执行过程中，可以用赋值语句对数组元素逐个赋值，例如：

```
for ( i=0;i<10;i++)
    num[i]=i;
```

数组初始化赋值是指在数组定义时给数组元素赋予初值，这种赋值方法是在编译阶段进行的，可以减少程序运行时间，提高程序执行效率。初始化赋值的一般形式为：

类型说明符 数组名[常量表达式]={值，值，…，值};

其中在{}中的各数据值即为相应数组元素的初值，各值之间用逗号间隔，例如：

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

相当于：

```
num[0]=0;num[1]=1;...;num[9]=9;
```

在简易密码锁程序 ex4_2.c 中对数组 tab 的说明也可以修改如下：

```
unsigned char code tab[]={0xc0,0xf9,0xa4,0xb0,0xbf,0x8b,0x8c};
```

这里没有指定数组的元素个数，在{}中说明的各数据值的个数就是数组中的元素个数，因此数组 tab 的元素个数是 7 个。



小提示 数组长度和数组元素下标在形式上有些相似，但这两者具有完全不同的含义。数组说明的方括号中给出的是长度，即可取下标的最大值加 1；而数组元素的下标是该元素在数组中的位置标识。前者只能是常量，后者可以是常量、变量或表达式。

采用数组实现任务 3-1 中的流水灯控制程序如下。

```
//程序：ex4_3.c
//功能：采用数组实现的流水灯控制程序
#include<reg51.h>           //包含头文件 reg51.h，定义 51 单片机的专用寄存器
void delay ( unsigned int i ); //延时函数声明
```



```

void main ( )                //主函数
{
    unsigned char i;
    unsigned char display[] = {0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
    while (1)
    { for (i=0;i<8;i++)
      { P1=display[i];        //显示字送 P1口
        delay (10000);        //延时
      }
    }
}

void delay ( unsigned int i ) //延时函数参见任务1-2的 ex1_1.c 程序

```



扫一扫
阅读本
程序代
码

4.2.2 二维数组

定义二维数组的一般形式是：



扫一扫
看二维
数组演
示文稿



扫一扫
看二维
数组教
学视频

类型说明符 数组名 [常量表达式1] [常量表达式2];

其中“常量表达式1”表示第一维下标的长度，“常量表达式2”表示第二维下标的长度，例如：

```
int num[3][4];
```

说明了一个3行4列的数组，数组名为 num，该数组共包括 3×4 个数组元素，即：

```

num[0][0],num[0][1],num[0][2],num[0][3]
num[1][0],num[1][1],num[1][2],num[1][3]
num[2][0],num[2][1],num[2][2],num[2][3]

```

二维数组的存放方式是按行排列，放完一行后顺次放入第二行。对于上面定义的二维数组，先存放 num[0] 行，再存放 num[1] 行，最后存放 num[2] 行；每行中的 4 个元素也是依次存放的。由于数组 num 说明为 int 类型，该类型数据占 2 字节的内存空间，所以每个元素均占有 2 字节。

二维数组的初始化赋值可按行分段赋值，也可按行连续赋值。

例如，对数组 a[3][4] 可按下列方式进行赋值。

(1) 按行分段赋值可写为：

```
int a[3][4]={{80,75,92,61},{65,71,59,63},{70,85,87,90}};
```

(2) 按行连续赋值可写为：

```
int a[3][4]={80,75,92,61,65,71,59,63,70,85,87,90};
```

以上两种赋初值的结果是完全相同的。

二维数组的应用参见任务 4-2 中的 ex4_5.c 及任务 4-3 中的 ex4_8.c。

4.2.3 字符数组

用来存放字符量的数组称为字符数组，每一个数组元素就是一个字符。

字符数组的使用说明与整型数组相同，例如“char ch[10];”语句，说明 ch 为字符数

组，包含 10 个字符元素。

字符数组的初始化赋值是直接各字符赋给数组中的各个元素。例如：

```
char ch[10]={'c','h','i','n','e','s','e','\0'};
```

以上定义说明了一个包含 10 个数组元素的字符数组 ch，并且将 8 个字符分别赋值到 ch[0]~ch[7]，而 ch[8]和 ch[9]系统将自动赋予空格字符。

当对全体数组元素赋初值时也可以省去长度说明，例如：

```
char ch[]={'c','h','i','n','e','s','e','\0'};
```

这时 ch 数组的长度自动定义为 8。

通常用字符数组来存放一个字符串，字符串总是以 '\0' 来作为串的结束符。因此，当把一个字符串存入一个数组时，也要把结束符 '\0' 存入数组，并以此作为字符串的结束标志。

C 语言允许用字符串的方式对数组做初始化赋值，例如：

```
char ch[]={'c','h','i','n','e','s','e','\0'};
```

可写为：

```
char ch[]={"chinese"};
```

或去掉 {}，写为：

```
char ch[]="chinese";
```



扫一扫看
字符和字
符串演示
文稿



扫一扫看
字符和字
符串教学
视频

一个字符串可以用一维数组来装入，但数组的元素数目一定要比字符多一个，即字符串结束符 '\0'，由 C 编译器自动加上。

字符串数组的应用参见任务 4-4 中的程序 ex4_10.c。

任务 4-2 小型 LED 数码管字符显示屏控制

1. 目的和要求

利用单片机控制 6 个共阳极数码管，采用动态显示方式稳定显示小王的生日 1990 年 12 月 25 日，显示效果为“901225”，让读者理解 LED 数码管动态显示程序的设计方法。

2. 电路设计

采用静态显示方式控制 6 个数码管，则需要单片机提供 6 组 8 位并行 I/O 端口，必须对单片机并行 I/O 端口进行扩展，这将大大增加硬件电路的复杂性及成本。本任务采用动态显示方式控制 6 个共阳极数码管，电路连接方式如图 4.6 所示。将各位共阳极数码管相应的段选控制端并联在一起，仅用一个 P1 口控制，用八同相三态缓冲器/线驱动器 74LS245 驱动。将各位数码管的公共端也称为“位选端”，由 P2 口控制，用六反相驱动器 74LS04 驱动。

3. 源程序设计

动态显示方式就是按位顺序地轮流点亮各位数码管，即在某一时段，只让其中一位数码管的“位选端”有效，并送出相应的字型显示编码。例如，首先让左边第一个数码管显示字符 9，单片机的 P2 口送出位选码，即语句“P2=0xfe;”，经反相驱动器后，控制 P2.0 连接的数码管位选端为高电平，点亮该数码管，同时单片机的 P1 口送出“9”的字型编码，即