



全国优秀教材特等奖



“十三五”职业教育国家规划教材



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>



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。

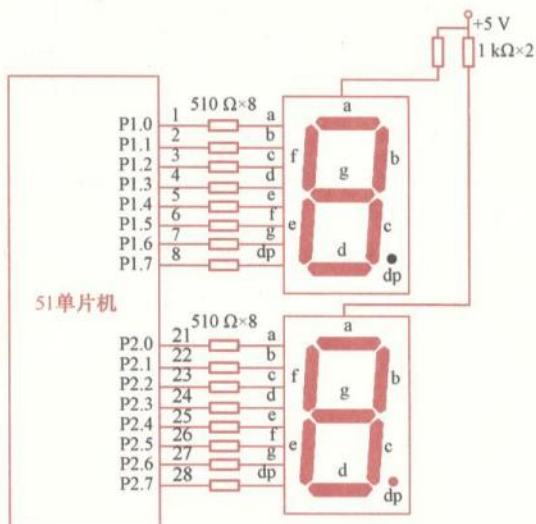


图 4.5 两位数码管静态显示接口电路

4.2 数组的概念

知识分布网络



扫一扫看数码管与单片机的静态电路连接微视频



扫一扫看数据排队微视频:什么是数组

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数组。

数组还分为一维、二维、三维和 multidimensional 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稿



扫一扫看数据排队微视频:数组定义及初始化



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];        //定义字符型数组 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
```



扫一扫看程序
简洁高效
微视频:数组应用



```

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:

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



扫一扫
看二维
数组演
示文稿



扫一扫看数
据方阵队形
微视频:二
维数组



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";
```

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。



扫一扫看
字符组成串
微视频: 字
符数组



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

任务 4-2 小型 LED 数码管字符显示屏控制



扫一扫下
载该任务
的仿真实
验文件

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



语句“P1=0x90;”, 数码管显示字符9。然后, 采用同样的方法编程, 依次顺序显示第2到第6个数码管。表4.3列出了6个显示字符在P2、P1口依次输出的数据。

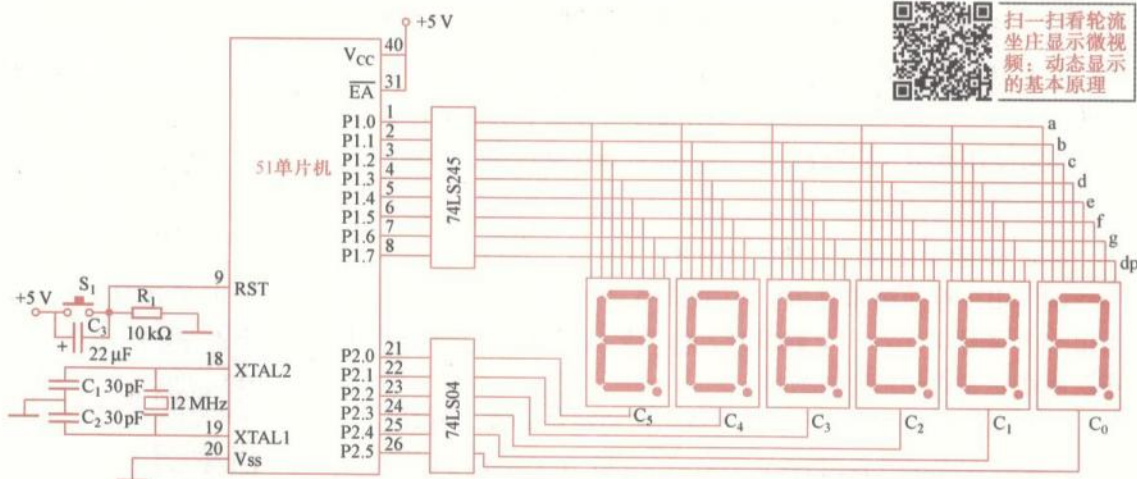


图 4.6 6 位数码管动态显示电路

表 4.3 901225 字符在 P2、P1 口依次输出的数据

| 显示字符 | 9 | 0 | 1 | 2 | 2 | 5 |
|--------------|-----------|-----------|-----------|-----------|-----------|-----------|
| P2 (位选码) | 11111110B | 11111101B | 11111011B | 11110111B | 11101111B | 11011111B |
| P1 (字型码或段选码) | 0x90 | 0xc0 | 0xf9 | 0xa4 | 0xa4 | 0x92 |



小提示 了解了数码管的动态显示过程, 可以采用顺序程序结构实现该显示程序。主要程序段如下:

```
while (1) {
    P1=0xff;          //关显示, 共阳极数码管 0xff 熄灭
    P2=0xfe;          //送位码, 选中 P2.0 连接的数码管
    P1=0x90;          //送 9 的字型码
    delay (200);      //延时
    ...
    P1=0xff;          //关显示
    P2=0xdf;          //送位码, 选中 P2.5 连接的数码管
    P1=0x92;          //送 5 的字型码
    delay (200);      //延时
}
```

由表4.3可以看出, 位选码有一定的变化规律, 依次左移一位, 字型码没有规律, 为此定义一个一维数组来存储这6个字符的字型码。6位数码管动态显示生日“901225”的程序如下。

```
//程序: ex4_4.c
//功能: 6 位数码管动态显示生日 “901225”
#include<reg51.h>          //包含头文件 reg51.h, 定义 51 单片机的专用寄存器
```



```
#include<intrins.h>           //包含头文件 intrins.h, 使用了内部函数_crol_ ( )
void delay ( unsigned int i ); //延时函数声明
void main ( )                 //主函数
{
    unsigned char led[]={0x90,0xc0,0xf9,0xa4,0xa4,0x92};
                                //设置数字901225共阳极字型码

    unsigned char i,w;
    while (1)
    {
        w=0xfe;                //位选码初值为0xfe
        for ( i=0;i<6;i++)
        {
            P1=0xff;            //关显示, 共阳极数码管 0xff 熄灭
            P2=w;                //位选码送位选端 P2口
            w=_crol_(w,1);       //位选码左移一位, 选中下一位 LED
            P1=led[i];           //显示字型码送 P1口
            delay (100);         //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

小问答

问：在 LED 数码管动态显示程序中，如果把延时函数的实际参数 100 修改为 10 000，LED 数码管显示会有什么变化？为什么？

答：6 个数码管上轮流显示“901225”，不能同时稳定显示。

由于人的眼睛存在“视觉驻留效应”，必须保证每位数码管显示间断的时间间隔小于眼睛的驻留时间，才可以给人一种稳定显示的视觉效果。如果延时时间太长，每位数码管闪动频率太慢，就不能产生稳定显示效果。

问：如果去掉关显示语句“P1=0xff;”显示效果会有什么影响？

答：会有拖影现象，影响显示效果。如果在字符交替显示时，不关掉显示的话，会将上一个字符显示在下一个字符位置上很短的时间，形成拖影，导致显示效果不美观。

4. 任务小结

本任务采用单片机并行 I/O 端口 P1 口、P2 口控制 6 个共阳极数码管显示，进一步训练应用单片机并行 I/O 端口的能力，熟练掌握数码管动态显示接口技术以及使用数组和循环程序结构进行程序设计与调试的能力。

5. 举一反三

(1) 采用 6 个数码管以多屏方式交替显示小王的生日“901225”和学号“125315”。实现分屏交替显示不同字符信息的参考程序如下。



```
//程序: ex4_5.c
//功能: 6 位数数码管交替稳定显示 “901225” 和 “125315” 两屏内容
#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
//函数名: disp3
//函数功能: 实现 6 个数码管交替显示 “901225” 和 “125315” 两屏内容
//形式参数: 无
//返回值: 无
void disp3 ( )
{ unsigned char lednum[2][6]={0x90,0xc0,0xf9,0xa4,0xa4,0x92},
                                {0xf9,0xa4,0x92,0xb0,0xf9,0x92}};
                                //二维数组存储 901225、125315 的共阳极字型码
    unsigned char com[]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf};
                                //一维数组存储位选码

    unsigned char i,j,num;
    for ( num=0;num<2;num++)    //显示两屏字符
        for ( j=0;j<100;j++)   //一屏字符扫描显示 100 遍, 达到稳定显示效果
            for ( i=0;i<6;i++)
            {
                P1=0xff;        //关显示
                P2=com[i];       //位选码送位选端 P2口
                P1=lednum[num][i]; //显示字型码送 P1口
                delay (100);     //延时
            }
}

void main ( )                  //主函数
{
    while (1) disp3 ( );
}

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

(2) 实用移动显示广告屏设计。采用单片机控制 6 个数码管以移动显示方式显示 “HELLO” 字样, 由右往左移动显示的过程如图 4.7 所示。

本任务只要能依次显示出 6 屏不同的内容, 就可以达到移动显示的效果。值得注意的是本任务每屏显示数据之间对应一定的排列顺序, 将所有在显示屏上要出现的显示字符按顺序排列为 “xxxxHELLOx”, 其中 x 表示无显示。

可见, 第 1 屏显示的 6 位数据为 “xxxxxH”, 第 2 屏显示的 6 位数据为 “xxxxHE”, 依次类推, 第 6 屏显示数据为 “HELLOx”。参考程序如下。

```
//程序: ex4_6.c
//功能: 6 个数码管移动显示 “HELLO”
#include<reg51.h>           //包含头文件 reg51.h, 定义 51 单片机的专用寄存器
```

| | | | | | | |
|---|---|---|---|---|---|-----|
| □ | □ | □ | □ | □ | H | 第1屏 |
| □ | □ | □ | □ | H | E | 第2屏 |
| □ | □ | □ | H | E | L | 第3屏 |
| □ | □ | H | E | L | L | 第4屏 |
| □ | H | E | L | L | O | 第5屏 |
| H | E | L | L | O | □ | 第6屏 |
| □ | □ | □ | □ | □ | H | 第1屏 |

图 4.7 移动显示过程



扫一扫看行云
流水显示微视
频: 可以移动
的动态显示



```

void delay ( unsigned int i ) ;           //延时函数声明
//函数名: disp3
//函数功能: 实现 6 个数码管移动显示 “HELLO”
//形式参数: 无
//返回值: 无
void disp3 ( )
{ unsigned char ledmove[] =
    { 0xff,0xff,0xff,0xff,0xff,0x89,0x86,0xc7,0xc7,0xc0,0xff };
    //存储移动字符xxxxxHELLOx的共阳极字型码

    unsigned char com[]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf};    //存储位选码
    unsigned char i,j,num;
    for ( num=0;num<6;num++)           //显示六屏字符
        for ( j=0;j<100;j++)           //一屏字符扫描显示 100 遍, 达到稳定显示效果
            for ( i=0;i<6;i++)
            {
                P1=0xff;                //关显示
                P2=com[i];              //位选码送位选端 P2口
                P1=ledmove[num+i];      //显示字型码送 P1口
                delay ( 100 ) ;          //延时
            }
}

void main ( )                           //主函数
{
    while (1) disp3 ( ) ;
}

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

4.3 LED 数码管动态显示

知识分布网络

LED数码管动态显示

动态显示的概念

动态显示接口

扫一扫看多个
数字怎么办微
视频: 动态显
示的电路连接扫一扫看LED
数码管动态
显示硬件接
口演示文稿

在单片机应用系统设计中, 往往需要采用各种显示器件来显示控制信息和处理结果。当采用数码管显示, 且位数较多时, 一般采用数码管动态显示控制方式。

动态显示是一种按位轮流点亮各位数码管, 高速交替地进行显示, 利用人的视觉暂留作用, 使人感觉看到多个数码管同时显示的控制方式。

数码管动态显示电路通常是把所有数码管的 8 个显示段分别并联起来, 仅用一个并行 I/O 端口控制, 称为“段选端”。各位数码管的公共端, 称为“位选端”, 由另一个 I/O 端口控制。6 个数码管动态显示硬件连接电路参见图 4.6。

动态显示是指按位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。此时, 其他位的数码管因“位选端”无效而处



于熄灭状态;下一时段按顺序选通另外一位数码管,并送出相应的字型显示编码,按此规律循环下去,即可使各位数码管分别间断地显示出相应的字符。这一过程称为动态扫描显示。



小经验 与静态显示方式相比,当显示位数较多时,动态显示方式可节省 I/O 端口资源,硬件电路简单,但其显示的亮度低于静态显示方式。由于 CPU 要不断地依次运行扫描显示程序,将占用 CPU 更多的时间。若显示位数较少,采用静态显示方式更加简便。

动态显示方式在实际应用中,由于需要不断地扫描数码管才能得到稳定显示效果,因此在程序中不能有比较长时间地停止数码管扫描的语句,否则会影响显示效果,甚至无法显示。

通常,在程序设计中,把数码管扫描过程编成一个相对独立的扫描函数,在程序中需要延时或等待查询的地方调用该函数,代替空操作延时,就可以保证扫描过程不会间隔时间太长。

任务 4-3 LED 点阵式电子广告牌控制



扫一扫看LED
数码管动态
显示程序演
示文稿

1. 目的和要求

利用单片机控制一块 8×8 LED 点阵式电子广告牌,将一些特定的文字或图形以特定的方式显示出来。

任务要求在 8×8 LED 点阵广告牌上稳定显示“0”。

2. 电路设计

用单片机控制一块 8×8 LED 点阵式电子广告牌的硬件电路如图 4.8 所示。每一块 8×8 LED 点阵式电子广告牌有 8 行 8 列共 16 个引脚,采用单片机的 P1 口控制 8 条行线, P0 口控制 8 条列线。



扫一扫看LED
数码管动态
显示程序操
作训练

