第 5 章

数码管的显示及驱动

数码管是组成单片机系统的一个常用输出器件，常见的八段数码管由 8 个发光二极管组成，控制不同的发光二极管的亮灭就可以显示不同字符。当数码管位数较少时，每个数码管可用单片机的一个并口控制，这就是静态显示；当数码管位数较多时，为节省所占用的单片机并口数量，采用动态扫描显示。本章主要讲解数码管的工作原理和软件编程驱动方法。

5.1 数码管显示原理

LED 数码管显示数字和符号的原理与用火柴棒拼写数字非常类似，用几个发光二极管也可以拼成各种各样的数字和图形，LED 数码管就是通过控制对应的发光二极管来显示数字的。

LED 数码管的结构如图 5-1 所示。数码管实际上是由 7 个发光二极管组成一个 8 字形，还有另外一个发光二极管做成圆点形，主要作为显示数据的小数点使用，这样一共使用了 8 个发光二极管，所以称为八段 LED 数码管。这些段分别由字母 a、b、c、d、e、f、g 和 dp 来表示。当给这些数码管特定的段加上电压后，这些特定的段就会发亮，以显示出各种数字和图形。例如，当显示“1”时，应使 b、c 点亮，其他段熄灭；当显示“2”时，应使 a、b、g、e、d 点亮，其他段熄灭。

数码管内部，一般把各笔画段的发光二极管阴极或阳极连在一起，叫作数码管的公共端，相应的阴极作为公共端的数码管叫作共阴极数码管，阳极作为公共端的数码管叫作共阳极数码管。图 5-1 中引脚 A 即为数码管的公共端。图 5-2 和图 5-3 分别为共阳极和共阴极数码管的内部电路。

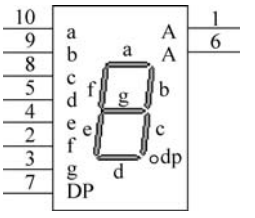

图 5-1 常见数码管结构

共阴极数码管一般把阴极接地，阳极是独立的，当我们给某一个阳极接上高电平时，对应的这个发光二极管就被点亮了。例如，想让数码管显示 8，小数点也亮时，需要 8 个阳极都送高电平。需要显示 5 时，b、e、dp 送低电平不亮，其他段送高电平点亮。总之，想让某段发光二极管亮时，就给它相应的阳极引脚送高电平，否则送低电平。每个发光二极管引脚的状

态只有高、低两种，如果把每个引脚看成一个二进制数的一位时，八位数码管的所有引脚状态组成的就是一字节。当需要显示 0～9 共 10 个数字时，每个都对应一字节的二进制数，这个二进制数叫作显示代码。为了方便显示数据，我们可以用显示代码组成一个表格，需要显示某个数字时，就把它的显示代码直接调出来，送阳极显示就行了，这样可以降低数码管显示程序实现的难度。

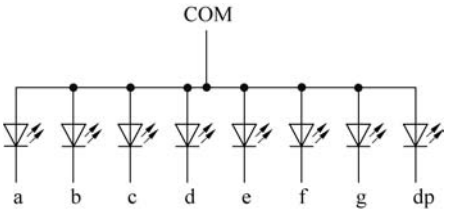

图 5-2　共阳极数码管的内部电路

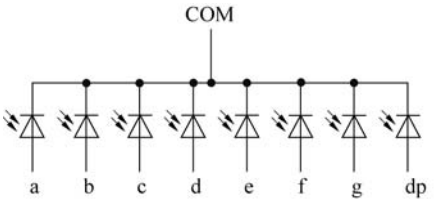

图 5-3　共阴极数码管的内部电路

共阳极数码管与共阴极的控制方式相反，公共端接高电平，需要点亮的发光二极管阴极送低电平，所以它的显示代码与共阴极的相反。共阳极和共阴极显示代码如表 5-1 所示。

表 5-1　共阳极和共阴极显示代码

代码	显示字符									
	0	1	2	3	4	5	6	7	8	9
共阳极代码	C0H	F9H	A4H	B0H	99H	92H	82H	F8H	80H	90H
共阴极代码	3FH	06H	5BH	4FH	66H	6DH	7DH	07H	7FH	6FH
代码	显示字符									
	A	b	C	d	E	F	P	U	T	y
共阳极代码	88H	83H	C6H	A1H	86H	8EH	8CH	C1H	CEH	91H
共阴极代码	77H	7CH	39H	5EH	79H	71H	73H	3EH	31H	6EH

表 5-1 中只列出了部分段码，读者可以根据实际情况选用，也可对某些显示的字符重新定义，也可选择其他字形的 LED 数码管。除了“8”字形的 LED 数码管外，市面上还有“±1”型、“米”字形和“点阵”形 LED 显示器，同时厂家也可根据用户的需要定做特殊字形的数码管。本章后面介绍的 LED 显示器均以“8”字形的 LED 数码管为例。

因为数码管是由发光二极管组成的，点亮发光二极管需要 5mA 以上的电流，单片机的端口不能提供这么大的电流，所以数码管与单片机连接时需要加驱动电路。在前面单片机端口驱动 LED 灯时用的是上拉电阻，在学习板上数码管的驱动采用的是 74HC573 锁存器，它的驱动电流较大，可以提供数码管需要的电流。

学习板上的数码管是四位一体的，即 4 个显示位“8”制成一体的，共用了两个，可以同时显示 8 位数字。其中每个四位一体的显示位公共端是独立的，除公共端之外的独立端相同名字的连在一起。例如，同一个显示块内的 4 个显示位，a 段在内部短接在一起。其他 7 个笔画段，相同名字的也都短接在一起，这个笔画段端叫作**段码端**；每位独立的公共端可以控

制多位一体中的哪一位数码管点亮，叫作**位码端**。

5.2　数码管的静态和动态显示

数码管的工作方式有两种：静态显示和动态显示。

1. 静态显示

静态显示就是指无论多少位 LED 数码管，同时处于显示状态。

LED 数码管工作于静态显示方式时，各位的共阴极（或共阳极）连接在一起并接地（或接+5V）；每位的段码线（a～dp）分别与一个八位的 I/O 口锁存器输出相连。如果送往各个 LED 数码管所显示字符的段码一经确定，则相应 I/O 口锁存器锁存的段码输出将维持不变，直到送入另一个字符的段码为止。正因为如此，静态显示方式的显示无闪烁，亮度都较高，软件控制比较容易。

四位 LED 静态显示电路如图 5-4 所示。它们的共阳极端连接在一起后接电源正极。其中各位 LED 可独立显示，只要在该位的段码线上保持段码电平，该位就能保持相应的显示字符，由于各位分别由一个八位的数字输出端口控制段码线，故在同一个时间里，每一位显示的字符可以各不相同。静态显示方式接口编程容易，但是占用口线较多。如图 5-4 所示电路中，若用 I/O 口线接口，要占用 4 个八位 I/O 口。如果显示器的数目增多，则需要增加 I/O 口的数目。因此，在显示位数较多的情况下，所需的电流比较大，对电源的要求也就随之增高，这时一般都采用动态显示方式。

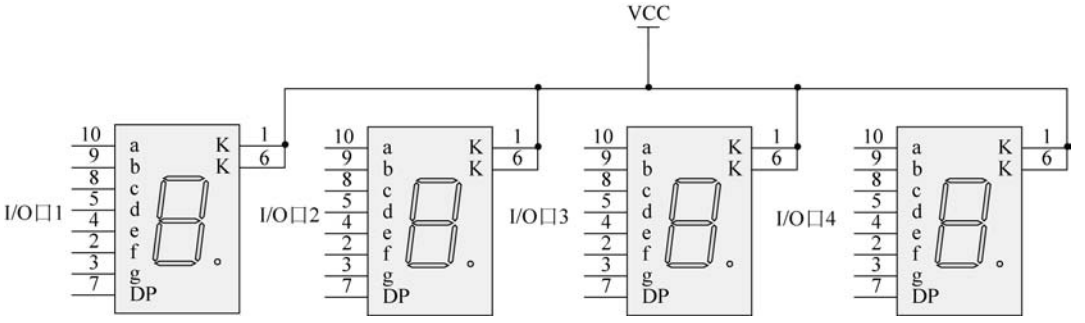

图 5-4　四位 LED 静态显示电路

2. 动态显示

动态显示是指无论在任何时刻只有一个 LED 数码管处于显示状态，即单片机采用“扫描”方式控制各个数码管轮流显示。

在多位 LED 显示时，为简化硬件电路，通常将所有显示位的段码线的相应段并联在一起，由一个八位 I/O 口控制，而各位的共阳极或共阴极分别由相应的 I/O 线控制，形成各位的分时选通。图 5-5 为一个四位八段 LED 动态显示器电路。

图中段码线占用一个八位 I/O 口，而位选线占用一个四位的 I/O 口。动态显示段码端是八位，驱动时要占用一个并口；位码端中的每个位要占用一个端口，当显示位数不大于八

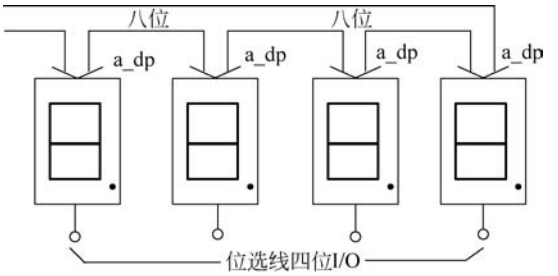

图 5-5　四位八段 LED 动态显示器电路

位时，占用的并口数量不超过一个。

由于各个数码管的段码线并联，在同一时刻，4 个数码管将显示相同的字符。因此，若要各个数码管能够同时显示出与本位相应的显示字符，就必须采用动态扫描显示方式。即在某一时刻，只让某一位的位选线处于选通状态，而其他位的位选线处于关闭状态，同时，段码线上输出相应位要有显示的字符的段码。这样，在同一时刻，四位 LED 中只有被选通的那一位显示出字符，而其他三位则是熄灭的。同样地，在下一时刻，只让其下一位的位选线处于选通状态，而其他各位的位选线处于关闭状态，在段码线上输出将要显示字符的段码，此时，只有在选通位显示出相应的字符，其他各位则是熄灭的。如此循环下去，就可使各位显示出将要显示的字符。

虽然这些字符是在不同时刻出现的，而在同一时刻，只有一位显示，其他各位熄灭，但由于 LED 数码管的余辉和人眼的“视觉暂留”作用，只要每位显示间隔足够短，则可以造成“多位同时亮”的假象，达到同时显示的效果。

LED 不同位显示的时间间隔(扫描间隔)应根据实际情况而定。发光二极管从导通到发光有一定的延时，导通时间太短，发光太弱，人眼无法看清；时间太长，要受限于临界闪烁频率，而且此时间越长，占用单片机的时间也越多。另外，显示位数增多，也将占用大量的单片机时间，因此动态显示的实质是以牺牲单片机时间来换取 I/O 端口的减少。动态显示的亮度既与导通电流有关，也与点亮时间和间隔时间的比例有关。调整电流和时间参数，可实现亮度较高较稳定地显示。动态显示的亮度比静态显示要差一些，所以在选择限流电阻时应略小于静态显示电路中的。

动态显示的优点是硬件电路简单，显示器越多，优势越明显；缺点是显示亮度不如静态显示的亮度高。如果“扫描”速率较低，会出现闪烁现象。

3. 学习板上的数码管驱动

学习板上采用的是多位一体的数码管，各位数码管的段码端是并联在一起的，给它们各位送的显示代码是相同的，如果多位同时点亮时，可以看到各位上的显示数字是相同的，所以它应该采用动态显示驱动。动态显示可以让不同位的数码管同时显示不同字符，并且占用的硬件端口又极少，想让送到段码端的字符在哪一位上显示，可以通过位码端控制。例如，共阳极的数码管某一位要点亮时，这一位的位码端就送高电平，其他不显示的位送低电

平。通过动态扫描显示，轮流给段码端送出需要显示的字形码，同时在显示的位码端送出高电平，利用发光管的余辉和人眼的视觉暂留作用，使人感觉几位数码管同时在亮，但实际是它们依次轮流点亮。单片机学习板上的数码管硬件驱动原理如图 5-6 所示。学习板上使用的是共阴极数码管。

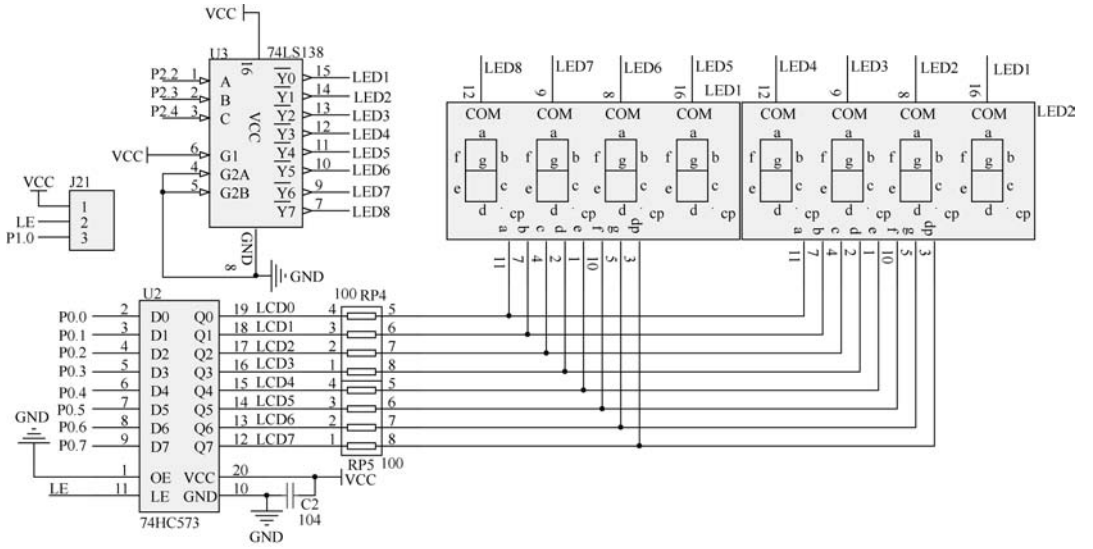

图 5-6　单片机学习板上的数码管硬件驱动原理

这里一共用到了两个四位一体的数码管，可以同时显示 8 位数字。两个显示块的段码端 a～dp 并联在一起，再通过限流电阻接到锁存器 74HC573 的数据输出端，锁存器的数据输入端接单片机的 P0 口。每个数码管的位码端接到 3-8 译码器 74LS138 的一个输出端，74LS138 的输入端由单片机 P2 口的 3 个引脚信号控制。图 5-6 中 LED1～LED8 是网络标号，网络标号相同的表示它们是连在一起的。例如，网络标号是 LED1 的位码端接到译码器的 Y0 口，它们的网络标号是相同的。

这里介绍一下 3-8 译码器的工作原理。如图 5-6 所示，74LS138 译码器选通端为 G1、G2A、G2B，当 G1 为高电平，G2A、G2B 为低电平时，可将地址端 A、B、C 输入信号组成的三位二进制编码，在输出端 Y0～Y7 译码成相应端口的低电平输出。例如，当 CBA＝110 时，对应的数值为 6，此时 Y6 引脚输出低电平，其他输出引脚保持高电平。3-8 译码器 74LS138 的真值表如表 5-2 所示。

表 5-2　74LS138 译码器真值表

G1	G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
×	H	×	×	×	H	H	H	H	H	H	H	H
L	×	×	×	×	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H

续表

G1	G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

在表 5-2 中，×表示状态任意，H 是高电平，L 是低电平，G2 包含了 G2A 和 G2B。这里我们看到，采用译码器控制的主要目的是节约单片机的端口。如果不接译码器，直接用单片机控制数码管的位码端时，需要用到 8 个单片机端口，增加了译码器则只用 3 个端口就能代替原来 8 个端口的控制功能，对于单片机硬件资源特别有限的情况下，这种方法可以大大提高硬件资源利用率。

再来看一下锁存器 74HC573 的工作原理。锁存器 74HC573 的引脚 OE 是三态允许控制端，又叫作输出使能端、输出允许端；D0～D7 是数据输入端，Q0～Q7 是数据输出端；LE 是锁存允许端，或叫锁存控制端。锁存器的真值表如表 5-3 所示。

表 5-3　74HC573 真值表

OE	LE	D	Q
L	H	H	H
L	H	L	L
L	L	×	Q0
H	×	×	Z

表 5-3 中，Z 表示高阻状态，既不是高电平也不是低电平，它的电平状态由与它相连接的其他电气状态决定；Q0 表示上次的电平状态。由表可见，当 OE 端为高电平时，输出端保持高阻状态，因此让芯片工作必须将该引脚接为低电平，由图 5-6 可见，锁存器的 OE 端是直接接地的。在 OE 端为低电平的前提下，当 LE 为高电平时，D 端和 Q 端的状态同为低电平或高电平；而当此时 LE 变为低电平时，不论 D 端的状态如何，Q 端总是保持上次的电平状态。也就是 OE 端为低电平的前提下，LE 为高电平时，Q 端跟随 D 端的状态变化；LE 为低电平时，Q 端的状态将保持，不随 D 端而变化。实现锁存功能，就需要对 LE 引脚的状态进行控制，这里将 LE 通过一个端子接到高电平或单片机的 P1.0 口，当不需要锁存数码管显示时，将它接到高电平，需要数码管显示锁存或更新显示时，将它接到 P1.0 控制，端子上可以通过短接片把该引脚接到高电平或 P1.0。在以后的程序编写过程中，默认 LE 端是受 P1.0 控制的。硬件上锁存器输入端 D0～D7 接到单片机的 P0 口，锁存器的输出端 Q0～Q7 接到数码管的段码端，这样 P0 口就可以通过锁存器向数码管送出显示代码了。

学习板上接锁存器的主要目的是：单片机的端口数量有限，P0 口不仅要控制数码管，还要控制其他硬件设备，加了锁存器就可以实现信号的隔离，不用的硬件通过锁存器的锁存端关闭，这样就不会造成设备之间的相互干扰。

5.3　数码管显示的编程实现

【例 5-1】　用数码管显示数字 3。

学习板上的八位数码管段码端都是并联在一起的，让其中一个位显示 3，要用位码端控制该位点亮，不显示的位码端控制该位熄灭。因为学习板采用的是共阴极数码管，当显示代码送到段码端的同时，显示位位码端要送低电平，其他位送高电平。由图 5-6 可见，如果我们想让最低位的数码管点亮，3-8 译码器的 Y0 应该输出低电平，根据译码器的真值表，它的输入端 A、B、C 都应该输入低电平。同时再使锁存器的 LE 端为高电平，即相连的 P1.0 输出高电平，再把数字 3 的显示代码 0x4F(共阴极显示代码见表 5-1)通过锁存器送到数码管的段码端，此时要注意显示代码的顺序，小数点在最高位，a 在最低位。

```
#include<reg51.h>
sbit le=P1^0;                    //锁存控制端的定义
void delay()
{
unsigned int i,j;
for(i=1;i>0;i--)
for(j=110;j>0;j--);
}
void main()
{
while(1)
{
    le=1;                        //LE 端为高电平,锁存器输出随输入变化
    P2=0xe3;                     //位码端控制,点亮最低位数码管
    P0=0x4f;                     //数字 3 的段码送 P0 口
    le=0;                        //LE 端低电平,锁存器输出锁存
    delay();
}
}
```

本程序可以实现八位数码管中最低位显示 3，程序中“P2＝0xe3;”指令给 P2 口送的二进制数为 1110 0011B，即 P2.2、P2.3、P2.4 端口为 0，它们是 3-8 译码器的输入。根据译码原理，使译码器的 Y0 端输出 0，其他端口输出为 1，即使最低位数码器位码为 0，点亮该数码管。指令“P0＝0x4f;”给 P0 口送的是数字 3 的共阴显示代码，P0 口接到锁存器的输入，通过锁存器接到数码器的段码端。两条指令组合，使最低位数码管显示 3。锁存器的控制端接到 P1.0 端口，程序开始用指令“sbit le＝P1^0;”定义了一个特殊功能位，定义后程序中再需要对 P1.0 操作时，都用 le 来代替了。程序编写上一定要注意，给锁存器输入端送数据的


例 5-2 详解

指令前后,一定要有 LE 端置 1 和清 0 的指令。

【例 5-2】 在一个数码管上循环显示 0～9 十位数字,每隔 1s 更新一次显示。

如果还是在最低位数码管上显示,同例 5-1,要使译码器 Y0 端输出 0,还是需要给 P2 送 0xe3。本例要显示的数据每隔 1s 要更新一次,所以送 P0 口的显示代码不是固定的。为了便于显示,可以先把这些显示代码存入一个数组,存的时候一定是按顺序存放的。例如,想显示 i 时,从数组里取第 i 个显示代码就是数字 i 的显示代码,这样可以使显示程序更容易实现。

```
#include<reg51.h>
sbit le=P1^0;
void delay()
{
 unsigned int i,j;
 for(i=1000;i>0;i--)
 for(j=110;j>0;j--);
}
void main()
{
unsigned char code tab[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
unsigned char k;
P2=0xe3;                    //位码送 P2 口,使最低位数码管亮
while(1)                    //无限循环
{
  for(k=0;k<10;k++)         //循环 10 次送 10 个数
     {
      le=1;                 //锁存控制端高电平,输出随输入变化
      P0=tab[k];            //数字 k 的显示代码送 P0 口
      Le=0;                 //锁存控制端低电平,输出锁存
      delay();
     }
  }
}
```

本例中用了一个 for 循环,循环十次顺序取数组 tab[]中的显示代码。因为数组 tab[]中的显示代码是按照显示数字 0～9 的顺序存放的,如果送到 P0 口的显示代码是 tab[k]时,显示的数字就是 k,取显示代码后再加上 1s 延时,这样就实现了顺序间隔 1s 显示 0～9 十位数字的功能。

【例 5-3】 在 4 个数码管上轮流显示数字 1234。

单片机学习板上一共有 8 个数码管,我们取其中的低四位来做实验。当让最低位数码管点亮时,译码器的 Y0 端应输出 0,此时 P2.4、P2.3、P2.2 的值应为 000,此时把数字 4 的显示代码送 P0;当倒数第二位数码管亮时,同理 P2.4、P2.3、P2.2 的值应为 001,此时把数字 3 的显示代码送 P0 口;当倒数第三位数码管亮时,同理 P2.4、P2.3、P2.2 的值应为 010,

此时把数字 2 的显示代码送 P0 口；当倒数第四位数码管亮时，同理 P2.4、P2.3、P2.2 的值应为 011，此时把数字 1 的显示代码送 P0 口。每位数码管显示时，都加入一定延时，让该位点亮一段时间再切换到下一个数码管显示。此时我们就会看到 1234 这 4 个数字，轮流在 4 个数码管上显示了。

```
#include<reg51.h>
sbit le=P1^0;
void delay()
{
  unsigned int i,j;
  for(i=1;i>0;i--)
  for(j=110;j>0;j--);
}
void main()
{
unsigned char code tab[]={0x06,0x5b,0x4f,0x66}; //1～4 显示代码存入数组
unsigned char i;
while(1)
 {
     i=1;
     for(i=1;i<5;i++)                          //循环 4 次显示 4 个数
     {
     le=1;                                     //锁存控制端置高电平
     P2=0xef-4*(i-1);                          //位码送 P2 口
     P0=tab[i-1];                              //显示代码送 P0 口
     le=0;                                     //显示状态锁存
     delay();
     }
 }
}
```

本例显示的字符不仅固定在一个数码管上，要 4 个数码管轮流显示，所以除了要送不同的显示代码，还要对应点亮相应的数码管。程序中指令“P2=0xef-4*(i-1);”用于给 P2 口送出位码，其中数字 0xef 写成二进制数是 1110 1111B，如果 i=1，则 P2=0xef，此时 P2.4～P2.2 的状态为 011，译码器的 Y3 将输出低电平，点亮倒数第 4 个数码管；同时当 i=1 时，指令“P0=tab[i-1];”使送到 P0 口的显示代码为数字 1 的代码，P0 和 P2 的状态配合，使倒数第 4 个数码管显示 1。同理，当 i=2 时，P2=0xeb，译码器 3 个输入口状态为 010，此时它的 Y2 输出低电平，点亮倒数第三个数码管；同时 P0=0x5b，给 P0 送出数字 2 的显示代码。同理，i=3 时，倒数第二个数码管被点亮，并显示 3；i=4 时，倒数第一个数码管被点亮，并显示 4。当 for 循环执行一次时，4 个数码管被循环点亮一次。

这个程序实现的是 4 个数码管轮流显示不同的数字，如果我们想让它们同时显示 1234

4 个数字时,实际上把循环显示中的延时时间调得足够短就可以了。采用循环高速扫描的方式,分时轮流选通各数码管的位码端,使数码管轮流导通显示,当扫描速度达到一定程度时,人眼就分辨不出来了,这就是利用了人眼的视觉暂留原理。尽管实际上各位数码管并非同时点亮,但只要扫描的速度足够快,给人的印象就是一组稳定的显示数据,认为各数码管是同时发光的。所以编程的关键是显示数字后的延时时间要足够短。我们可以试试把程序中的延时子程序延时时间调到 1ms,再把程序下载到学习板,就可以看到 4 个数码管同时显示数字 1234 了。

将本例程序下载到学习板上的显示效果如图 5-7 所示。

图 5-7 实验效果图

数码管也常用于一些计时显示方面,如电子秒表、电子时钟、电子万年历等,这种功能较前面的几个显示例子实现相对复杂一些。

学习板中多位数码管采用动态扫描显示方式工作,主程序需要循环执行动态扫描显示子程序,才能保证数码管显示效果的稳定。当时间到需要更新显示时,再偶尔打断动态扫描显示子程序的运行。也就是说,要保证显示效果,更新显示的定时功能要在后台运行,并且程序以动态显示为主。定时更新显示的功能可以由定时中断程序来实现。

例 5-4 详解

【例 5-4】 用数码管实现秒表功能,循环显示 0～59s。

本例显示使用学习板上八位数码管的最低两位,采用动态扫描显示,取十位的显示代码时,控制十位的数码管亮;取个位的显示代码时,控制个位的数码管亮。显示数字从 0 开始,直到 59,再循环回到 0。定时中断设为每隔 1s 中断一次,每当 1s 时间到时,打断动态显示程序更新显示。

实现方法:秒信号的产生可用定时器来实现,即用定时器 T0 实现 50ms 定时,然后用中断计数器记录中断次数,当中断 20 次时,刚好 1s。用一个变量存储秒,每计满 1s,该变量的值加 1,计满 60s 时清 0。将 0～9 的段码按数字从小到大的顺序存在数组中,需要显示时再按顺序把段码取出来送端口显示。用如下方法获得两位显示数字:

- 显示数字除以 10 取余数就是个位上的数字;
- 显示数字除以 10 取整就是十位上的数字。

```
#include<reg51.h>
unsigned char code tab[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
unsigned char i,s;                    //定义中断次数变量和秒存储变量
sbit le=P1^0;                         //定义锁存器锁存控制端
void delay()                          //延时 1ms 子程序
{
unsigned int m,n;
for(m=1;m>0;m--)
```

```
for(n = 110;n > 0;n -- );
}
void xianshi(unsigned char k)                    //显示子程序
{
 le = 1;                                         //锁存控制端置高电平
 P2 = 0xe3;                                      //位码送 P2 口,使最低位 LED 点亮
 P0 = tab[k % 10];                               //取秒的个位显示代码送 P0 口显示
 le = 0;                                         //锁存控制端置低电平
 delay();
 le = 1;                                         //锁存控制端置高电平
 P2 = 0xe7;                                      //位码送 P2 口,使十位 LED 点亮
 P0 = tab[k/10];                                 //取秒的十位显示代码送 P0 口显示
 le = 0;                                         //锁存控制端置低电平
 delay();
 P0 = 0;                                         //使 LED 熄灭
}
void main()
{
 EA = 1;                                         //开中断
 ET0 = 1;
 TMOD = 0x01;                                    //设置 T0 为定时方式 1
 TH0 = (65536 - 50000)/256;                      //设置 T0 初值,定时 50ms
 TL0 = (65536 - 50000) % 256;
 TR0 = 1;                                        //开启定时器
 i = 0;                                          //中断次数计数器初值为 0
 s = 0;                                          //秒计数初值为 0
 while(1)                                        //循环执行显示子程序
 {
  xianshi(s);                                    //调用显示子程序,将秒计数值 s 送 LED 显示
 }
}
void timer0()interrupt 1                         //T0 溢出中断子程序
{
 i++;                                            //进入中断,中断计数值加 1
 if(i == 20)                                     //中断 20 次,时间为 1s
 {
  i = 0;                                         //1s 时间到,清 0 中断计数值
  s++;                                           //1s 时间到,秒计数值加 1
  if(s == 60)s = 0;                              //如果 s 加 1 为 60,则清 0
 }
TH0 = (65536 - 50000)/256;                       //T0 重赋初值
TL0 = (65536 - 50000) % 256;
}
```

秒表程序主要由四部分组成,分别是主程序、延时子程序、显示子程序、中断子程序。主程序负责定时器的初始化、中断的初始化、设置变量初值、调用显示子程序等,主程序的大部分时间都是在调用显示子程序。显示子程序负责把定时得到的秒计数值送 LED 显示,程序中取个位的显示代码用的是指令“P0＝tab[k％10];”,其中 k 对应的是秒计数值,秒计数值

是两位数，k%10 将计数值除以 10 后取余数，即秒值的个位，再取显示代码数组 tab 中的第 k%10+1 个数组值，即秒值个位的显示代码送 P0 口；相应的取秒值的十位显示代码用的指令是“P0=tab[k/10];”，因为 k/10 的功能是两位数 k 除以 10 后取整，即 k 的十位。中断子程序在 T0 启动后每隔 50ms 中断一次，负责检查定时时间是否到了 1s，到了就更新秒计数值，当秒计数值等于 60 时，还要清 0，使显示值重新循环。T0 溢出中断子程序中一定要有 T0 重赋初值指令，才能保证每隔 50ms 中断一次。

【例 5-5】 用数码管显示随机检测结果。

显示实现方法：实际应用中，常常需要用显示器显示一些动态变化的参量，如显示温度、压力等。这些检测值特点是不断动态变化的，如何获得动态参量的段码就是本例显示的关键。实际上，无论检测结果如何变化，要显示的结果总是由 0～9 这 10 个数字构成的，所以可将它们的显示代码按显示数据大小顺序存在一个数组中，单片机把要显示的数据分成个位、十位、百位、千位，再分别到数组中取显示代码送显示器对应位显示。例如，要显示的随机数是“8765”，可以按以下方法将每位要显示的数字分离出来：

- 8765 除以 10 所得余数是 5，就是要显示的个位数；
- 8765 除以 100 所得余数是 65，再除以 10 商是 6，就是要显示的十位数；
- 8765 除以 1000 所得余数是 765，再除以 100 商是 7，就是要显示的百位数；
- 8765 除以 1000 商是 8，就是要显示的千位数。

因为存放显示代码的数组中的元素是按顺序存储的，第一个是数字 0 的显示代码，第二个是数字 1 的显示代码，依次类推，所以只要获得了某位的显示数据，取显示代码就非常方便了。例如，要显示数字 5 时，它的显示代码就是 tab 中的第六个元素，可以写成 tab[5]，将这个代码送到 LED 的段码端即可显示。

另外，单片机对检测数据的采集和处理是需要时间的，如果把数据处理和动态显示同时放在主程序里顺序执行，就会在执行数据处理时停止动态显示，结果会造成 LED 动态显示结果不稳定、严重闪烁，所以在需要数码管 LED 动态扫描显示的场合，都需要不间断地循环执行动态扫描程序，即动态扫描不能长时间间断。在程序设计上可以考虑让动态显示在主程序中循环执行，数据采集在定时中断子程序中完成，这样动态显示只有在需要数据采集时暂时停止并且时间较短，对显示效果的影响不大；同时定时采集也能保证检测数据在固定时间内被采集到，不影响采集的实时性。到目前为止，我们还没有学习传感器的数据采集方法，所以本例程序用一个标准随机函数 rand()来模拟数据采集的结果。

```
#include <reg51.h>
#include <stdlib.h>                    //包含随机函数 rand()的头文件
unsigned char i;                       //存储中断次数
unsigned int s;                        //存储四位随机数
sbit le = P1^0;
unsigned char code tab[ ] =            //显示代码数组
{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
void delay()                           //延时 5ms
```

```
{
 unsigned int i,j;
 for(i = 5;i > 0;i -- )
 for(j = 110;j > 0;j -- );
}
void display(unsigned int x)                    //显示子程序
{
 le = 1;
 P2 = 0xe3;                                     //显示个位
 P0 = tab[x % 10];
 le = 0;
 delay();
 le = 1;
 P2 = 0xe7;                                     //显示十位
 P0 = tab[(x % 100)/10];
 le = 0;
 delay();
 le = 1;
 P2 = 0xeb;                                     //显示百位
 P0 = tab[(x % 1000)/100];
 le = 0;
 delay();
 le = 1;
 P2 = 0xef;                                     //显示千位
 P0 = tab[x/1000];
 le = 0;
 delay();
 P0 = 0;                                        //关显示
 delay();
}
void main()
{
 TMOD = 0x01;                                   //T0 设置为方式 1 定时
 TH0 = (65536 - 50000)/256;                     //设置 T0 初值
 TL0 = (65536 - 50000) % 256;
 EA = 1;                                        //开 T0 溢出中断
 ET0 = 1;
 TR0 = 1;                                       //启动 T0
 s = 0;
 while(1)display(s);                            //循环显示
}
void timer0()interrupt 1                        //T0 中断处理子程序
{
 TR0 = 0;                                       //关 T0
 i++;                                           //中断次数加一
 if(i == 20)                                    //中断 20 次定时 1 秒
```

```
{
    s = rand()/10;                                    //取随机数中的高四位
    i = 0;                                            //清 0 重新统计中断次数
 }
 TH0 = (65536 - 50000)/256;                           //T0 重赋初值
 TL0 = (65536 - 50000) % 256;
 TR0 = 1;                                             //启动 T0
}
```

本例用到了随机函数 rand()来表示动态检测结果,该随机函数包含在头文件 stdlib.h 中,所以程序一开始用指令"#include < stdlib.h >"来声明这个头文件。

主程序主要是在初始化时设置了定时器 T0 的初值、工作方式、开放了 T0 的溢出中断,然后循环显示动态检测结果。显示子程序用 LED 数码管的低四位显示动态检测结果,分别将位码值送 P2 口,段码值送 P0 口,轮流使 4 位 LED 点亮。在进入 T0 的溢出中断子程序后,先关闭 T0 停止定时,再判断定时时间是否到 1s,如果到了 1s,就取随机数的高四位送显示,否则重置 T0 初值,并启动 T0 开始计时。

对于需要 LED 动态扫描显示的场合,同时程序功能又相对复杂的应用,如何合理地进行程序设计,这是一个很好的实例。

习题

(1) 数码管的共阴或共阳显示代码是如何确定的?

(2) 简述数码管的静态显示原理。

(3) 简述数码管的动态显示原理。

(4) 编程:用八位数码管显示数字 0～7,并间隔 1s 向高位移位,直到 F 出现在最低位上,下 1s 到来时让 0 再出现在最低位上,如此循环。

本章小结

本章主要介绍单片机对数码管的显示驱动方法。首先介绍了数码管的结构原理,又介绍了数码管的静态和动态显示的驱动方法,讲解了数码管显示一位或多位数字的软件编程实现方法,及数码管作为计时显示器的软件实现方法。本章软件程序完整,有详细说明,并附有程序在学习板上的运行结果。