第5章

单片机的并行I/O口原理及编程

51单片机有4组8位的并行输入/输出端口，这4组I/O口既可以并行输入和输出8位数据，也可以每一位均独立作为输入或输出接口。学习单片机的I/O口是单片机入门的第一步，本章通过按键、继电器、蜂鸣器、数码管、点阵屏等器件介绍51单片机并行I/O口的应用与编程。

5.1 51单片机并行I/O口端口结构和工作原理

单片机中最常用的TTL电平，对于工作电压是5V的51系列单片机，通常0～2.4V代表“0”，3.6～5V代表“1”。51单片机的并行I/O口即是可以将“0”与“1”与对应电压信号进行双向转换的端口。AT89C51单片机有4组8位I/O口，分别为P0、P1、P2、P3，每组I/O口有8个引脚，都能独立地用作输入和输出，P0～P3口各有一个锁存器分别对应于80H、90H、A0H、B0H的地址。P1、P2、P3口能驱动4个LS型TTL门电路，直接驱动MOS电路，P0口在驱动TTL电路时能带动8个LS型TTL门，但驱动MOS电路时，作为通用I/O口需外接上拉电阻才能驱动。

5.1.1 P0口(P0.0～P0.7)

P0口的内部结构如图5-1所示。P0口由锁存器、输入缓冲器、切换开关、一个与非门、一个与门和场效应管驱动电路构成。P0口为三态双向口，当内部控制信号使MUX开关接通到锁存器时，P0口作为双向I/O端口使用，由于P0口内部没有上拉电阻，作为通用I/O口使用时需要外加上拉电阻。当单片机需要外部扩展存储器时，内部控制信号使MUX开关接通到内部地址/数据总线，此时P0口在ALE信号的控制下分时输出地址总线(低8位)和8位数据总线。P0口可驱动8个LS型TTL负载，当P0口作为I/O口使用时，应先向地址为80H的锁存器写入全1，此时P0口全部引脚浮空，可作为高阻

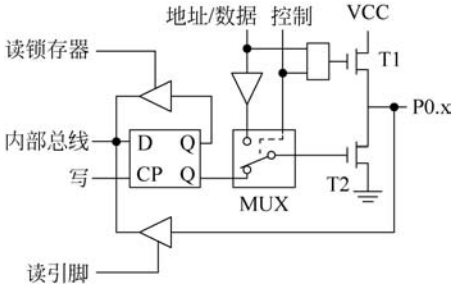

图5-1 P0口的内部结构

抗输入。

5.1.2 P1 口(P1.0～P1.7)

P1 口的内部结构如图 5-2 所示。P1 口有两个输入缓冲器,CPU 根据不同的指令发出“读锁存器”或“读引脚”信号。在读引脚时,也就是从外部数据输入数据时,为保证输入正确的外部输入电平信号,首先要向端口锁存器写 1,再进行读引脚操作,向端口锁存器写 1 后使驱动场效应管截止,引脚信号直接加到三态缓冲器,实现正确的写入。如果端口锁存器中原来的状态是 0,则加到输出驱动场效应管栅极的信号为 1,该场效应管导通,对地呈现低阻抗,此时,即使引脚上输出的信号为 1,也会因端口的低阻抗而使信号变化,使得外加的 1 信号写入时不一定是 1。P1 口作为输出口时,如果要输出 1,将 1 写入 P1 口的某一位寄存器,使输出驱动场效应管截止,该位的输出引脚由内部上拉电阻拉成高电平,即输出为 1,要输出 0,将 0 写入 P1 口的某一位寄存器,使输出驱动场效应管导通,该位对应的引脚被接到地,即输出为 0。P1 的输出缓冲级可驱动 4 个 TTL 逻辑门电路。在对 Flash ROM 编程和程序校验时,P1 口接收低 8 位地址。

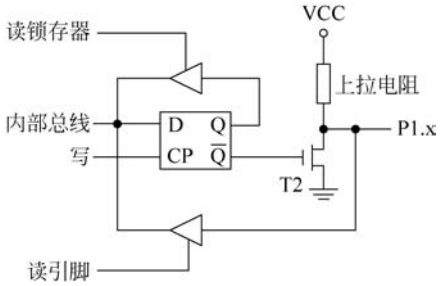

图 5-2 P1 口的内部结构

对于 AT89S51/52 单片机,P1 口的部分引脚也具有第二功能,如表 5-1 所示。

表 5-1 AT89S51/52 单片机 P1 口引脚的第二功能

引 脚	第二功能	说 明
P1.0	T2 外部输入	AT89S52 单片机有第三个定时/计数器 T2,此第二功能仅为 AT89S52 单片机所有
P1.1	T2 捕获/重载出发信号和方向控制(T2EX)	
P1.5	主机输出/从机输入数据信号(MOSI)	此第二功能是 SPI 串行总线接口的三个信号,用于对 S 系列单片机的 ISP 下载
P1.6	主机输入/从机输出数据信号(MISO)	
P1.7	串行时钟信号(SCK)	

5.1.3 P2 口(P2.0～P2.7)

P2 口的内部结构如图 5-3 所示。P2 端口在片内既有上拉电阻,又有切换开关 MUX,当切换开关向下接通时,从内部总线输出的一位数据经反相器和场效应管反相后,输出在端口引脚线上,当多路开关向上时,输出的一位地址信号也经反相器和场效应管反相后,输出在端口引脚线上。P2 的输出缓冲级可驱动 4 个 TTL 逻辑门电路。在访问外部程序存储器或 16 位地址的外部数据存储器时,P2 口送出高 8 位地址数据。在访问 8 位地址的外部数据存储器时,P2 口线上的内容(特殊功能寄存器中 R2 寄存器的内

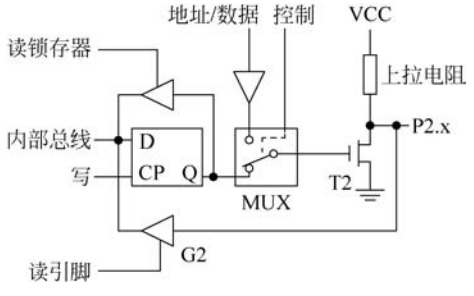

图 5-3 P2 口的内部结构

容)在整个访问期间不改变。Flash 编程或校验时,P2 也接收高位地址和其他控制信号。

5.1.4　P3 口(P3.0～P3.7)

P3 口的内部结构如图 5-4 所示。P3 口与 P1 口结构相似,区别仅在于 P3 口各端口线有两种功能选择:当处于第一功能时,第二输出功能线为 1,此时内部总线信号经锁存器和场效应管输入/输出,作为静态准双向 I/O 口线。当处于第二功能时,锁存器输出 1,通过第二输出功能线输出特定的内含信号,在输入方面可通过缓冲器读引脚信号,也可通过替代输入功能读入片内特定的第二功能信号。P3 口是一个带内部上拉电阻的 8 位双向 I/O 口,P3 的输出缓冲级可驱动 4 个 TTL 逻辑门电路。P3 口除作为一般 I/O 口线外,更重要的是它的第二功能,包括串行数据的输入/输出、外部中断 0 和 1 的输入、定时器 0 和 1 的外部计数脉冲输入、外部数据存储器的读写选通,参见表 2-2。后续章节要讲到的外部中断、定时器、串口、外部数据存储器扩展都与 P3 口的第二功能有关。

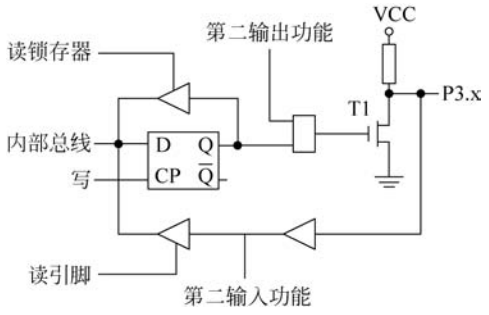

图 5-4　P3 口的内部结构

5.1.5　P0～P3 口功能总结

P0 口有三个功能:①外部扩展存储器时,当作数据(Data)总线;②外部扩展存储器时,当作地址(Address)总线;③不扩展时,可作一般的 I/O 使用,但内部无上拉电阻,作为输入或输出时应在外部接上拉电阻。

P1 口只作 I/O 口使用:其内部有上拉电阻。

P2 口有两个功能:①扩展外部存储器时,作地址总线使用;②作一般 I/O 口使用,其内部有上拉电阻。

P3 口有两个功能:除了作为 I/O 使用外(其内部有上拉电阻),还有一些特殊功能,由特殊寄存器来设置。

P0 口是双向口,双向指的是它被用作地址/数据端口时,P0 口才处于两个开关管推挽状态,当两个开关管都关闭时,才会出现高阻状态。当 P0 口用于一般 I/O 口时,内部接 VCC 的开关管是与引脚(端口)脱离联系的,只有拉地的开关管起作用,P0 口作为输出,必须外接上拉电阻,不然就无法输出高电平。

P1、P2、P3 口是准双向口,准双向口是指 P1、P2、P3 有固定的内部上拉电阻,当用作输入时被拉高,当外部拉低时会有拉电流,即电流从单片机 I/O 口流出到外部。

5.2　AT89C51 单片机 I/O 口驱动能力

单片机的引脚可以用程序来控制,输出高、低电平,这些是单片机的输出电压。但是,程序控制不了单片机的输出电流,单片机的输出电流很大程度上取决于引脚上的外接器件。

单片机输出低电平时，允许外部器件向单片机引脚内灌入电流，这个电流称为“灌电流”，外部电路称为“灌电流负载”。单片机输出高电平时，允许外部器件从单片机的引脚拉出电流，这个电流称为“拉电流”，外部电路称为“拉电流负载”。这些电流一般是多少？最大限度是多少？这就是常见的单片机输出驱动能力的问题。

每个引脚输出低电平时，允许外部电路向引脚灌入的最大电流为 10 mA；每个 8 位的接口（P1、P2、P3），允许向引脚灌入的总电流最大为 15 mA，而 P0 允许向引脚灌入的最大总电流为 26 mA，全部的四个接口所允许的灌电流之和最大为 71 mA。而当这些引脚输出高电平时，单片机的“拉电流”能力不到 1 mA。结论是：单片机输出低电平时，驱动能力尚可；输出高电平时，输出电流的能力很弱。综上所述，灌电流负载是合理的，而“拉电流负载”和“上拉电阻”会产生很大的无效电流并且功耗大。

5.3 并行 I/O 口应用举例

并行 I/O 口是 51 单片机最基础的功能模块，I/O 控制虽简单却能实现很多功能。以下通过键盘设计、继电器和蜂鸣器控制、数码管动态显示几个应用来介绍 I/O 口的使用与编程。

5.3.1 独立键盘设计

按键是单片机系统与操作人员之间交互重要组件，用于完成操作人员对单片机系统的输入控制。常见开关及符号如图 5-5 所示。

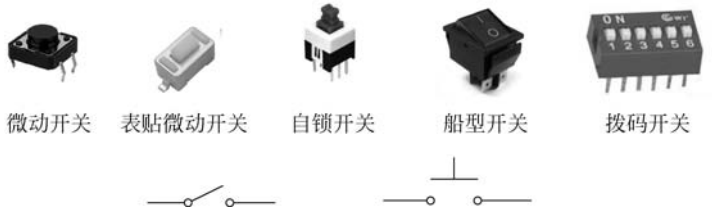

图 5-5 常见开关及符号

通常用到的是机械弹性开关，按下时闭合，松开后断开。自锁式开关按下时闭合且会自动锁住，再次按下时才弹起。单片机检测按键的原理：进行按键检测时用到 I/O 口的输入功能，把按键一端接地，另一端接 I/O 口，单片机初始状态 I/O 口为高电平，如果读到的 I/O 口电平为高电平，说明没有按键按下，当按键按下时将和地接通，程序一旦检测到 I/O 口变为低电平则说明按键按下，则执行相应的指令。在独立式键盘设计时要考虑如下问题：

（1）键的连击问题。连击会在一次按键中产生多次击键的效果，如图 5-6 所示，人为按键按下时间通常为几十到数百毫秒，若是程序在循环中不断检测按键是否按下，在按键按下的时间中，循环检测程序已经执行多次，会造成一次按下执行多次操作。对于这类问题，在键盘编程中常采用等待按键释放的处理方法来消除连击，使得每次按键仅产生一次键的处理效果。这样就可以避免当某个按键还未松开时，键扫描程序和处理程序已执行多遍。

（2）按键消抖问题。图 5-7 为理想按键的过程，当松开和按下按键时立即更换通断状

态，即按下时为低电平(0)断开时为高电平(1)。在实际过程中，当用手按下一个按键时，按键在闭合位置和断开位置之间往往弹跳若干下才会稳定到闭合状态，在释放一个按键时，也会出现类似的情况，这就是按键抖动。通常，按键所用触点为机械弹性触点，按键抖动是由按键机械结构的固有特性决定的，不可避免。按键抖动的持续时间一般为 5～10ms。

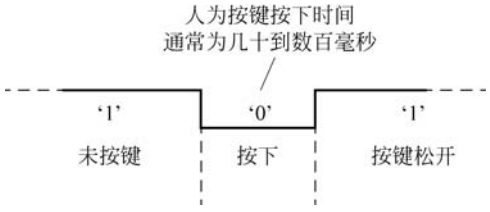

图 5-6　键的连击

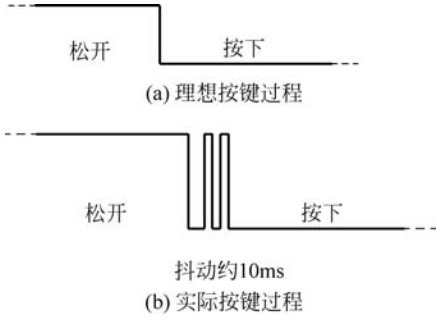

图 5-7　按键抖动

消除按键抖动有软件方法和硬件方法。软件方法常用延时 10ms 后等待按键稳定再次判断是否有键按下。硬件方法采用按键消抖电路，按键消抖电路常用 RS 触发器法或利用电容放电延时的并联电容法，专用键盘接口芯片常含有按键去抖电路。除特殊要求外，不建议自行加入硬件去抖电路，增加硬件开销，降低系统可靠性。建议用软件去抖。

(3) 多键同时闭合问题。当有两个或多个键同时闭合时，可以采用条件判断的方式来确认哪个按键有效。可采用以先按下的键为有效键，以按下时间最长的键为有效键，将最后释放的键视为有效键。有复合按键的设计按照复合按键的功能进行编程，无复合按键的设计中通常采用单键按下有效、多键按下无效的原则进行处理。

以下通过两个例子来讲解单片机 I/O 口查询法来实现按键功能。

例 5-1　用两个按键分组控制如图 5-8 所示的流水灯的左移和右移。

编程分析：在这个例子中用到了两个按键，K1 接 P3.6，K2 接 P3.7，程序中用特殊功能位定义 sbit 将两个按键所接的 I/O 口用 K1 和 K2 定义。按键编程采用查询的方式，在主程序循环中查询 K1 和 K2 是否为 0，如果为 0，则执行相应的操作。当检测到键按下后，延时 10ms，消除按键抖动后再次检测按键，为避免键的连击，即一次按下执行多次操作，等待按键释放以后，再执行对应的功能。

主程序如下：

```
void main( )
{
    P1 = 0xFE;                          //初值,从左到右第一个 LED 点亮
    while(1)
    {
        if(K1 == 0)
        {
            delay10ms(1);               //延时消抖
            if(K1 == 0)
            {
                while(K1 == 0);         //等待按键弹起
```

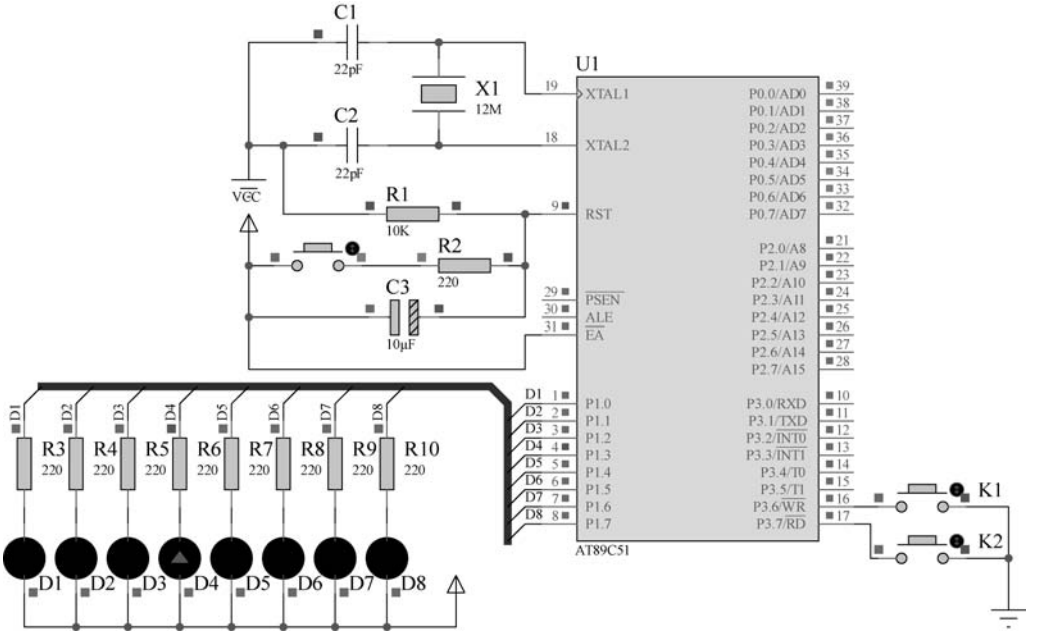

图 5-8　按键控制流水灯

```
                P1 = _crol_(P1,1);
            }
        }
        else if (K2 == 0)
        {
            delay10ms(1);           //延时消抖
            if(K2 == 0)
            {
                while(K2 == 0);     //等待按键弹起
                P1 = _cror_(P1,1);
            }
        }
    }
}
```

如果该程序中不加入等待按键弹起的语句会出现什么现象？在程序中将等待按键弹起这两条语句用/* */注释掉，再次编译。运行仿真程序，按下 K1 或 K2，LED 能左移或右移，但每次移动的位数是一个不确定的状态，这就是键的连击，按键一次按下执行了多次操作。

下面用调用函数的方式来实现上述功能，编制一子程序 Move_Led()来实现按键按下的功能。避免键的连击用更新并比较一个变量 Recent_Key 的方式来实现，和等待按键弹起原理一样，读者可自行分析。

```
void Move_Led()
{
    if((P3 & 0x40) == 0) P1 = _crol_(P1,1); //K1 按下
    if((P3 & 0x80) == 0) P1 = _cror_(P1,1); //K2 按下
```

```
}
void main( )
{
    UCHAR Recent_Key = 0xC0;          //最近按键,K1、K2 均未按下,屏蔽掉无关位后,值为初值
    P1 = 0xFE;                        //初值,从左到右第一个 LED 点亮
    while(1)
    {
        if((P3 &0xC0) != Recent_Key)
        {
            delay10ms(1); //延时消抖
            Recent_Key = (P3 &0xC0);
            Move_Led();
        }
    }
}
```

例 5-2　电路如图 5-9 所示,编程实现将 P1.4～P1.7 所接拨码开关的状态显示到 P1.0～P1.3 所接的 LED 上。

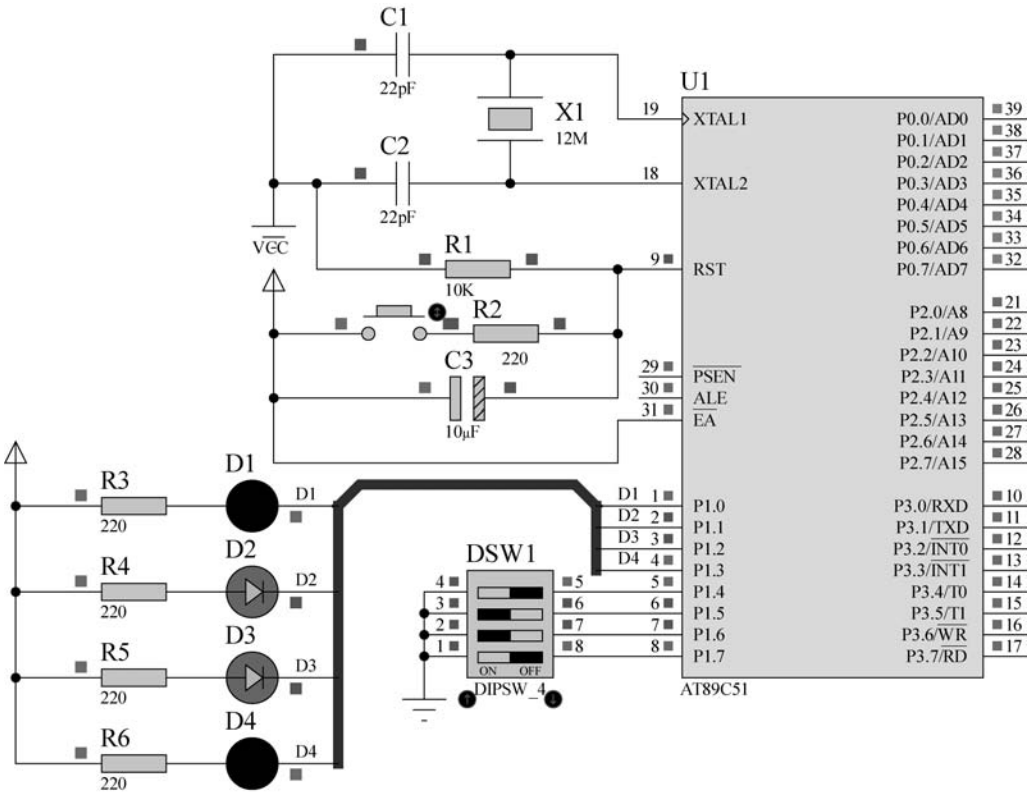

图 5-9　拨码开关状态显示

主程序如下：

```
void main()
{
    UCHAR temp;
```

```
    while(1)
    {
        temp = P1;                          //读拨码开关的状态
        temp >>= 4;                         //将 temp 的内容右移 4 位
        P1 = temp;                          //将拨码开关的状态送 LED 显示
    }
}
```

本程序 4 个拨码开关接 P1 的高 4 位，将 P1 高 4 位的状态读入就获得 4 个开关的状态，4 个 LED 接按键的低 4 位，右移 4 位送 P1 低 4 位就将开关的状态送 LED 显示。在 Keil 输入上面程序并进行编译，仿真，仿真电路的现象是无论拨码开关怎么拨动，4 个 LED 灯全亮，程序主要问题在哪里？

分析：I/O 口之所以能检测开关的状态，是因为单片机上电复位后 I/O 口的初始状态为高电平“1”，当开关与地接通，此时 I/O 口就读到了“0”，也就检测到开关接通。当开关断开时，由于 I/O 口初始状态是 1，此时读到的也是“1”，就检测到开关断开。回到以上程序，首先读拨码开关的状态，这句程序没有问题，然后将 temp 右移 4 位，将高 4 位读到的拨码开关的状态送到了低 4 位，这句程序也没有问题，下一句代码将拨码开关的状态送显示，问题就出在这一句代码。由于上一句代码采用右移运算符，是低位移出，高位补 0，此时高 4 位被补 0，然后 P1 送显，就将拨码开关的状态送给 P1.0～P1.3，同时将高位补的 4 个 0 送给 P1.4～P1.7，程序将 P1.4～P1.7 的给清 0，而检测程序是循环运行的，那么以后的无论拨码开关状态如何，读到的状态都是“0”，因此出现了拨码开关无效，4 个 LED 全亮的现象。在实际程序运行过程中，4 个 LED 是显示过 1 次高 4 位拨码开关状态的，由于程序循环运行时间极短，运行到下一次就是 LED 全亮，所以看不到这极短时间的显示。要解决这个问题，需要在将 4 位拨码开光状态移位到 P1 低 4 位送显的同时，不要改变 P1 高 4 位的状态，即是送显的同时将 P1 口高 4 位置 1。

针对上述分析，只需要将 temp 移位后将 P1.4～P1.7 置 1 即可。即将“P1＝temp;”这条程序修改为“P1＝temp | 0xf0;”，修改代码后程序编译，仿真实现功能。

5.3.2 继电器和蜂鸣器

继电器是一种电气控制器件，是当输入量的变化达到规定要求时，在电气输出电路中使被控量发生预定的阶跃变化的一种电器。它具有控制系统（又称输入回路）和被控制系统（又称输出回路）之间的互动关系。通常应用于自动化的控制电路中，当输入量（如电压、电流、温度、湿度等）达到设定值时，使被控输出电路导通或断开，实际上，它是用小电流去控制大电流运作的一种“自动开关”，故在电路中起着自动调节、安全保护、转换电路等作用。常用的电磁继电器的基本原理是通过电磁线圈实现低电压控制和高电压通断。主要技术指标有线圈额定电压、触点最大电压、触点最大电流等。图 5-10 为普通单刀双掷继电器实物。

图 5-10 继电器实物

以下通过一个简单的例子来讲解单片机 I/O 口对继电器的控制。

例 5-3　用按键通过继电器控制白炽灯的亮灭。

编程分析：电路如图 5-11 所示，当单片机 I/O 口给 PNP 三极管 Q1 的基极送低电平时，三极管导通，继电器电圈有电流流过，继电器吸合，控制白炽灯点亮。当单片机 I/O 口给 Q1 的基极送高电平时，三极管截止，继电器线圈无电流流过，继电器断开。在继电器线圈两端反向接了一个二极管 D1，(这个二极管叫续流二极管，由于在电路中起到续流的作用而得名)，一般选择快速恢复二极管或者肖特基二极管，以并联的方式接到继电器线圈两端，并与其形成回路。当继电器断电的瞬间会产生一个很强的反向电动势，续流二极管在回路将此反向电动势以续电流方式消耗，从而保护电路中的三极管不被损坏。程序只需要通过 I/O 口送 0 和送 1 即可打开和关闭白炽灯。

主程序如下：

```
void main()
{
    while(1)
    {
        if(key == 0)
        {
            delay10ms(1);                    //延时消抖
            if(key == 0)
            {
                while(key == 0);             //等待按键弹起
                lamp = ~lamp;
            }
        }
    }
}
```

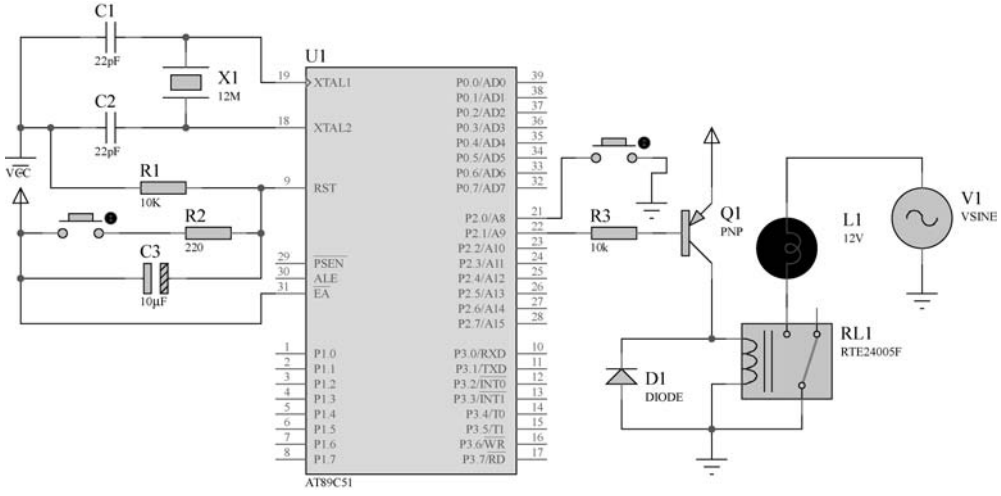

图 5-11　继电器控制白炽灯仿真电路

蜂鸣器是一种一体化结构的电子讯响器，采用直流电压供电，蜂鸣器实物如图 5-12 所

示。其广泛应用于计算机、打印机、复印机、报警器、电子玩具、电话机、定时器等电子产品中，作为发声器件。根据蜂鸣器结构主要分为压电式蜂鸣器和电磁式蜂鸣器两类。这两类蜂鸣器又各分为有源蜂鸣器和无源蜂鸣器两种，这里的“源”不是指电源而是指振荡源。有源蜂鸣器内部带振荡电路，加上电流电压即可发出鸣叫声，消耗电流 20mA 左右；无源蜂鸣器内部不带振荡源，所以如果用直流信号无法令其鸣叫，必须用 2～5kHz 频率去驱动它。51 单片机的 I/O 口提供的驱动电流远小于蜂鸣器的驱动电流，需采用三极管扩流或采用数字芯片驱动（如 74HC573）。

图 5-12　蜂鸣器实物图

例 5-4　无源蜂鸣器驱动电路图如图 5-13 所示。编程实现以下功能：①K1、K2 分别按下时蜂鸣器按不同的频率发声；②循环播放一段生日快乐歌。

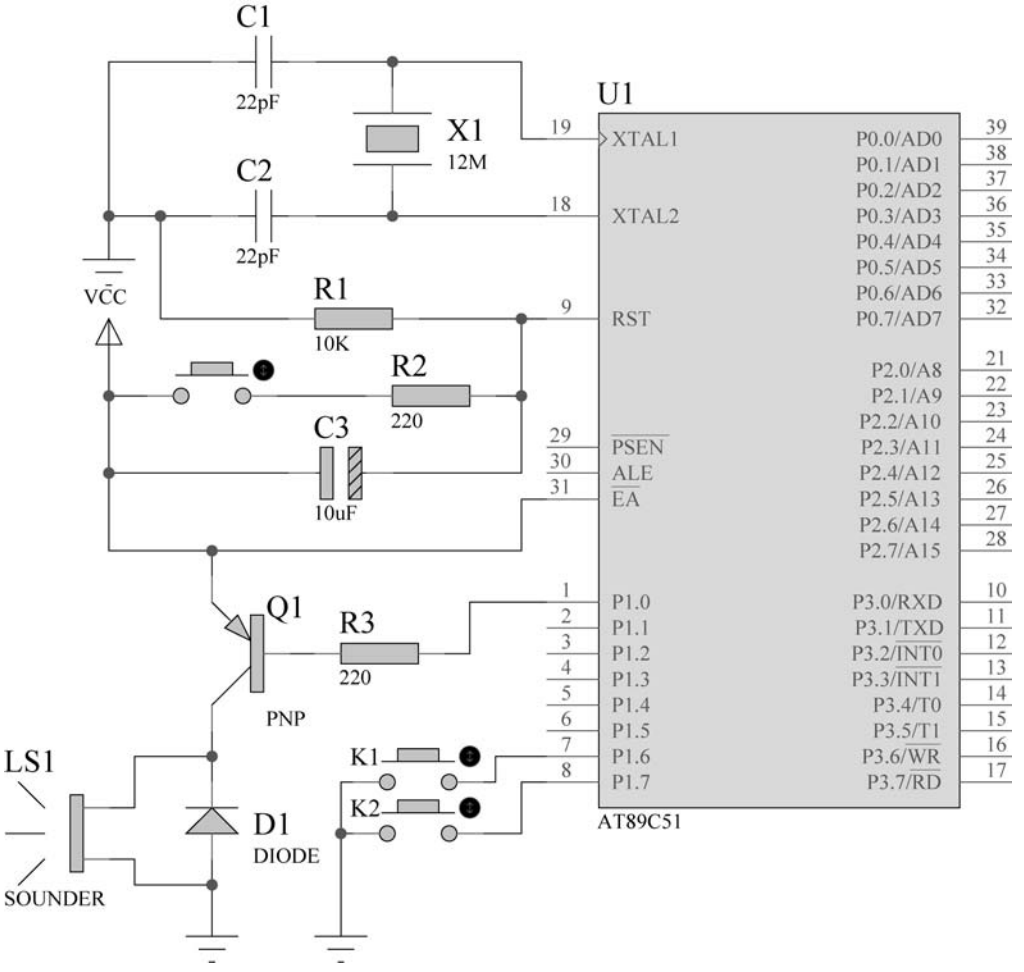

图 5-13　蜂鸣器仿真电路

编程分析：在本例中所用蜂鸣器为无源蜂鸣器，需要用方波信号去驱动它发声。要产生方波信号，I/O 口不断取反，在取反后加上一段延时程序就可以方便地实现。要求按照不

同频率进行发声，编制一带参数的子程序，在调用的时候输入不同的参数即可实现不同频率发声。

程序如下：

```
#include<reg51.h>
#define UCHAR unsigned char
#define UINT unsigned int
sbit BUZ = P1^0;
sbit K1 = P1^6;
sbit K2 = P1^7;
void delay(UINT x)
{
    UCHAR t;
    while(x--) for(t = 0; t<120; t++);
}
void play(UCHAR t)                          //按周期 t 发声
{
    UCHAR i;
    for(i = 0; i<100; i++)
    {
        BUZ = ~BUZ; delay(t);
    }
    BUZ = 0;
}
void main()
{
    P1 = 0xff;
    while(1)
    {
        if (K1 == 0) play(1);
        if (K2 == 0) play(2);
    }
}
```

通过蜂鸣器演奏乐曲，由乐谱的基本知识可知，音调和音调的时长是音符的主要特征，通过产生不同的音调和音调的时长可以奏出不同的音符，然后一个个音符串联在一起就可以产生美妙的音乐。

演奏一段生日快乐歌的程序如下：

```
#include<reg51.h>
#define UCHAR unsigned char
#define UINT unsigned int
sbit BUZ = P1^0;
UCHAR code music_tone[] = {212,212,190,212,159,169,212,212,190,212,142,159,
212,212,106,126,159,169,190,119,119,126,159,142,159,0};
//以上为生日快乐歌音符频率表，频率表最后为 0 表示播放结束
UCHAR code music_long[] = {9,3,12,12,12,24,9,3,12,12,12,24,9,3,12,12,12,
12,12,9,3,12,12,12,24,0};
//以上为生日快乐歌节拍表(每个音符演奏长短)，节拍表最后为 0 表示播放结束
void delay(UINT x) //延时子程序
```

```
{
    UCHAR t;
    while(x--) for(t = 0; t < 120;t++);
}
void playmusic()                    //播放歌曲子程序
{
    UINT i = 0,j,k;
    while(music_long[i] != 0 || music_tone[i] != 0) //如果播放没有结束,连续查表播放
    {
        for(j = 0; j < music_long[i] * 20;j++)
        {
            BUZ = ~BUZ;
            for(k = 0; k < music_tone[i]/3 ; k++);
        }
        delay(10);
        i++; //下一个音符
    }
}
void main()                         //主程序
{
    while(1)
    {
        playmusic();
        delay(500);                 //播放完一遍后,暂停,然后继续播放
    }
}
```

在程序中定义了两个数组：第一个数组 music_tone[]为生日快乐歌音符频率表，通过查表由蜂鸣器发出每个节拍的音调；第二个数组 music_long[]为生日快乐歌音符节拍表，通过查表控制每个音符播放的时长（节拍）。在这两个表中的最后一个数组都为 0，用于查表过程中查询是否播放完成。在程序中编制一个 playmusic()函数用于播放歌曲，当两个表均查到为 0 时，该曲播放完毕，停止查表，此子程序结束。在主程序循环中连续调用该函数，每播放完一次延时后连续播放。

播放音乐最好用定时器的方式实现，由于本章还未讲到定时器，就用延时的方式来实现。音乐节拍和频率的数据如何产生，可以网上下载 51 单片机蜂鸣器谱曲软件，如单片机音乐代码转换工具（Music Encode），本例只是通过一首歌曲来学习编程，在实际应用中蜂鸣器仅做报警和提示音使用，播放音乐音质不佳，不用深究如何谱曲。实际开发中需要用到音乐也是用音乐芯片来实现。

5.3.3 数码管的动态显示

在第 4 章讲到了数码管的静态显示，静态显示的特点是每个数码管的段选必须接一个 8 位数据线来保持显示的字形码。当送入一次段码后，显示字形可一直保持，直到送入新段码为止。这种方法的优点是占用 CPU 时间少，编程简单；缺点是每个数码管需要单独用一组 I/O 口控制，如果需要显示位数较多，占用大量 I/O 口。

多位数码管的实物和原理结构如图 5-4 所示，多位数码管是将所有位数码管的段选线

并联在一起，公共端分别引出作为位选线。动态扫描显示即由位选线控制是哪一位数码管有效，轮流选中各位数码管并送出字形码，利用发光管的余辉和人眼视觉暂留作用，使人的感觉好像各位数码管同时都在显示。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的阻值时应略小于静态显示电路。那么每送一位显示后延时多少呢，对于人眼超过每秒 24 帧就可以将静态画面连贯起来，对于 8 位数码管而言，扫描一遍 1/24s(约 40ms)，选择每送一位数码管显示后延时一般不大于 5ms 即可实现清晰稳定的显示。下面通过一个例子来讲解数码管动态显示的原理及编程。

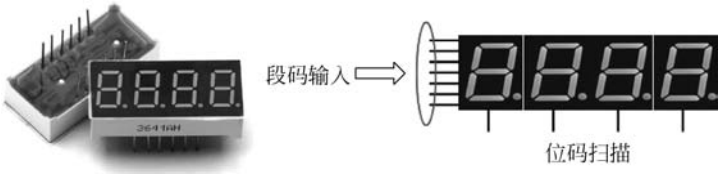

图 5-14　多位数码管

例 5-5　如图 5-15 所示，编程实现 8 位数码管动态显示。

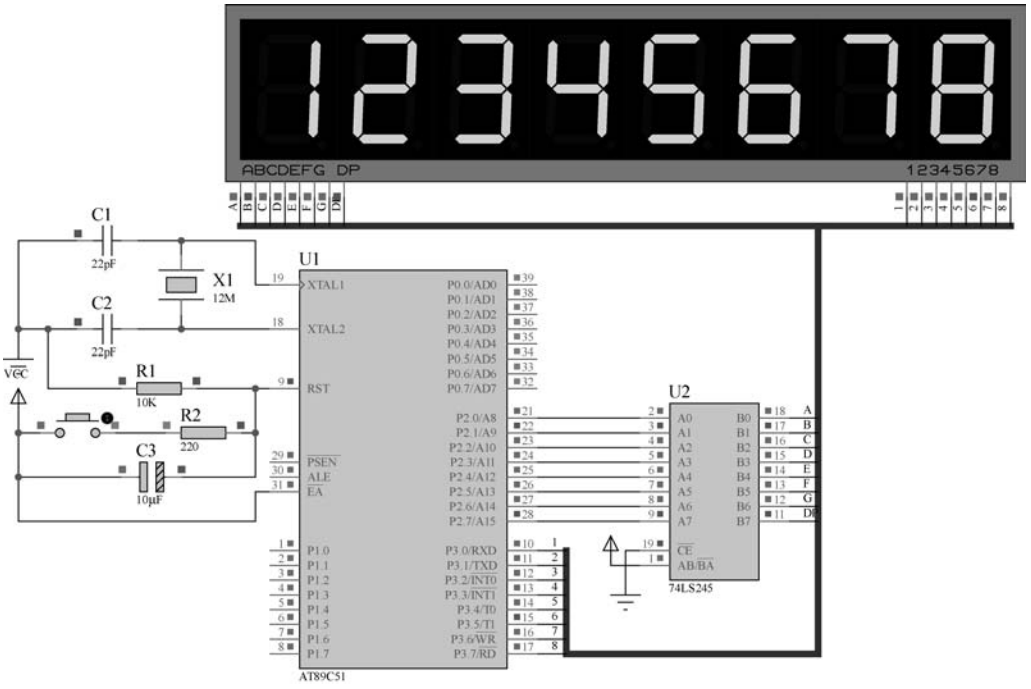

图 5-15　8 位数码管动态显示电路

程序分析：本例用的共阴极 8 位数码管，数码管位选线由 P3 口控制，数码管段选线由 P2 口控制，单片机 I/O 口拉电流能力很弱，电流不足以驱动数码管显示，通过 74LS245 来驱动。74LS245 是 8 路同相三态双向总线收发器，常用于驱动 LED 或者其他设备。从左边第一个数码管开始显示，此时 P3 口送的位选码为 0xFE，选中从左第一个数码管，然后 P2 口送对应显示的段码，延时 5ms 后，用_crol_将 0xFE 移位，选中第二个数码管，然后 P2 口送对应显示的段码，如此循环往复，即实现了数码管的动态显示。

程序如下：

```
#include <reg51.h>
#include <intrins.h>
#define UCHAR unsigned char
#define UINT unsigned int
UCHAR code smg[] =                          //0～9 的共阳数码管段码
{0X3F,0X06,0X5B,0X4F,0X66,0X6D,0X7D,0X07,0X7F,0X6F};
UCHAR Dsy_Buffer[8];
UCHAR Num[] = {1,2,3,4,5,6,7,8};
UCHAR Scan_Bit;                             //动态扫描位,选择要显示的数码管
UCHAR Dsy_Idx;                              //显示缓冲索引 0～7
void delayms(UINT ms)                       //延时子程序,实现约 n×1ms 的延时
{
    UCHAR t;
    while(ms--) for(t = 0; t < 120; t++);
}
void main()
{
    Scan_Bit = 0xFE;
    Dsy_Idx = 0x00;
    while(1)
    {
        UCHAR q;
        for(q = 0;q < 8;q++)
        {
            Dsy_Buffer[q] = smg[Num[q]];    //将带显示的数字查段码表
        }
        P3 = Scan_Bit;                      //选通相应数码管
        P2 = Dsy_Buffer[Dsy_Idx];
        delayms(5);
        Scan_Bit = _crol_(Scan_Bit,1);      //准备下次将选通的数码管
        Dsy_Idx = (Dsy_Idx + 1) % 8;        //索引在 0～7 内循环
    }
}
```

本例中用定义了一个变量 Dsy_Idx 作为显示缓冲索引，索引在 0～7 内循环，每选中相应的位选线后通过索引查表送显对应位置显示数据的显示段码。仿真后可以看到待显示的数字 12345678 清晰地显示在 8 位数码管上，双击单片机，仿真将单片机的晶振频率调整至 0.12MHz，相当于将延时增加 100 倍，此时可以清楚地展示多位数码管动态显示的原理，显示的数字是从左到右轮流依次逐位显示的。

5.3.4 点阵屏显示

LED 点阵显示屏广泛应用于汽车报站器、广告屏等。8×8 LED 点阵是最基本的点阵显示模块，理解 8×8 LED 点阵的工作原理就可以基本掌握 LED 点阵显示技术。点阵显示屏结构和实物如图 5-16 所示。

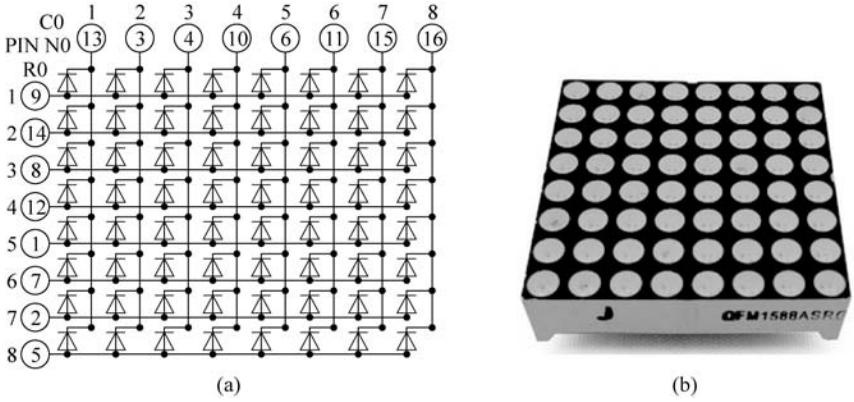

图 5-16　LED 点阵屏结构和实物

从图 5-16(a)可以看出，8×8 点阵由 64 个发光二极管组成，且每个发光二极管是放置在行线和列线的交叉点上，当对应的某一行为高电平，某一列为低电平时，对应的发光二极管就点亮。通过编程控制各显示点对应 LED 阳极和阴极端的电平，就可以有效地控制各显示点的亮灭，显示出相应的字符或图形。

普通数码管分为共阳极和共阴极数码管，市面上所售 LED 点阵对共阳还是共阴一般是根据点阵第一个引脚的极性所定义的，第一个引脚为阳极则为共阳，反之为共阴。

点阵屏的显示一般采用列扫描法，即先选定一列，给这一列送显示码并显示，再选定第二列，给第二列送显示码并显示，如此逐列选中依次送显，每列显示的时间不小于 5ms，由于人眼的视觉暂留效应，看到点阵屏上显示的图形或字符。

例 5-6　点阵屏显示驱动电路图如图 5-17 所示，在点阵屏上显示爱心的符号♥。

编程分析：8×8 点阵屏的接口电路如图 5-17 所示。将点阵屏共阳端通过 P3 口进行控制，用于逐列选择并提供显示驱动电流，由于 I/O 口驱动电流不足，通过 8 路通向三态总线收发器 74LS245 来驱动，点阵屏阴极的一端通过 P2 口送显示码。

首先将待显示的字符或图形转换成送显的字模，需要借助字模提取软件，本例用到的“字模提取 V2.1”软件下载自网络。取模的操作步骤(图 5-18)如下：

步骤一：新建图像，选择宽度为 8，高度为 8。

步骤二：在建好的模内绘制想要显示的图形，为便于绘制，可选择模拟动画下的放大格点操作后再进行绘制。

步骤三：单击修改图像下的黑白反显图像。

步骤四：单击取模方式下的 C51 格式，提取字模，生成的字模在点阵生成区内，可复制到程序里。

于是得到显示码 0xE3、0xC1、0x81、0x03、0x03、0x81、0xC1、0xE3，当 P3 口从左至右依次选中每一列显示的时候，P2 口依次送这 8 个显示码。

编程思路：点阵屏的驱动为动态扫描显示，通过 P3 口来进行列选，每次扫描显示一列，通过 P2 口送该列的字形码。由于人眼的视觉暂留效应，看到心形符号显示在点阵屏上。

显示爱心的程序如下。

```
#include<reg51.h>
```

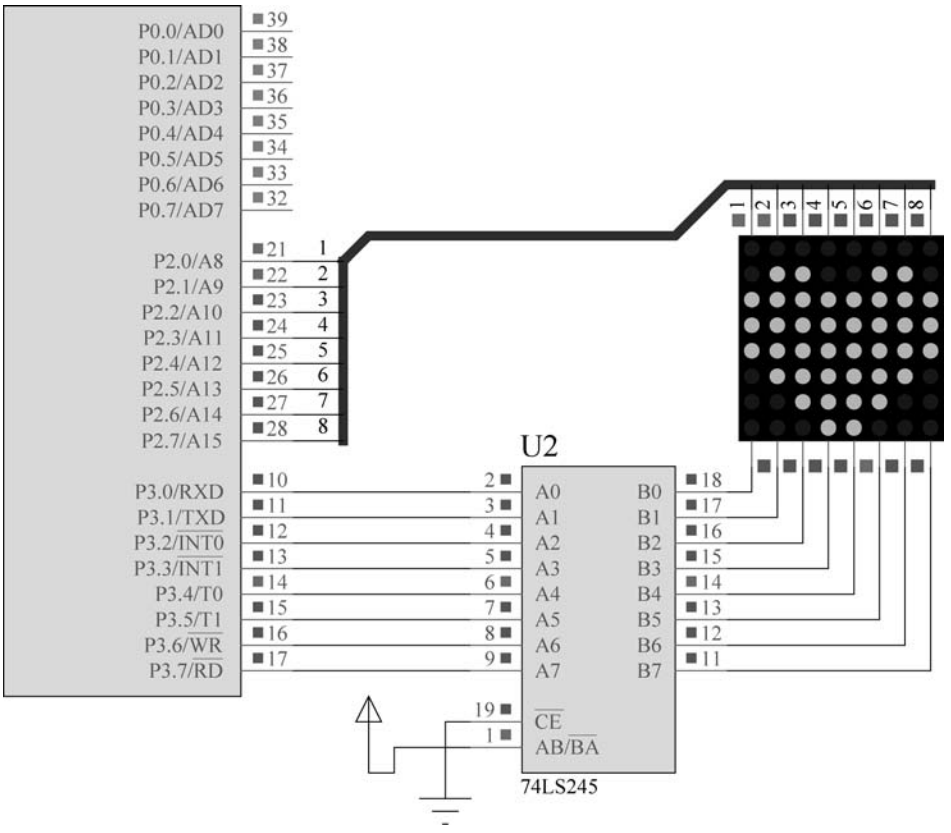

图 5-17　8×8 点阵显示仿真图

```
#include <intrins.h>
#define UCHAR unsigned char
#define UINT unsigned int
void delay(UINT x)
{
    UCHAR t;
    while(x--) for(t = 0; t < 120;t++);
}
UCHAR code table[] =
{
    0xE3,0xC1,0x81,0x03,0x03,0x81,0xC1,0xE3,
};
void main()
{
    UCHAR i = 0;
    P3 = 0x80;
    while(1)
    {
        P3 = _crol_(P3,1);
        P2 = table[i];
        delay(1);
        i = (i+1) % 8;
    }
}
```

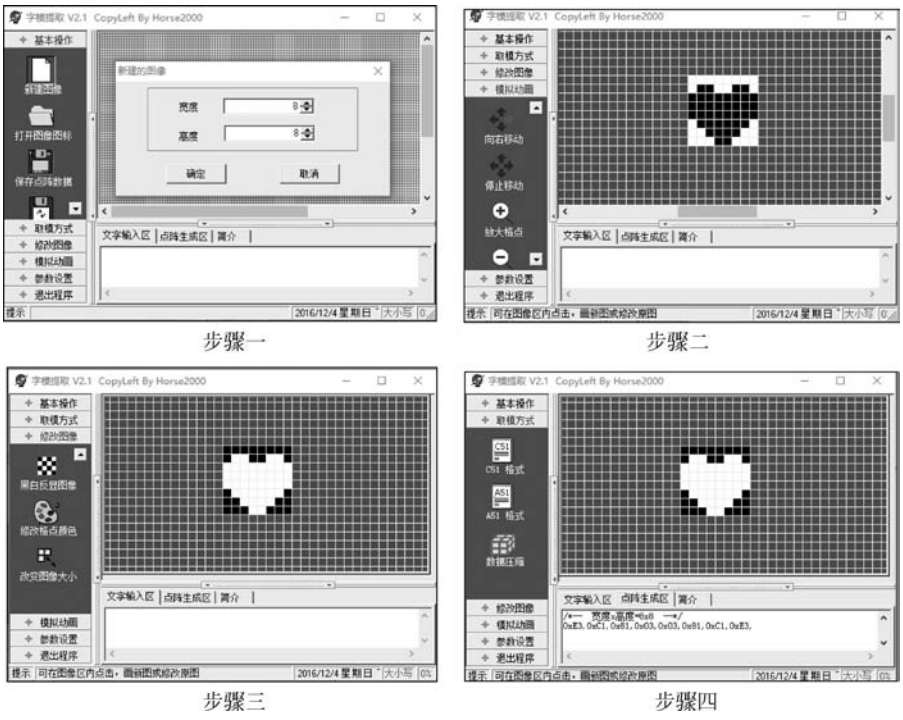

图 5-18　字幕提取软件操作步骤

习题

一、填空

1. 51 单片机_____负载大于_____负载，因此在驱动 LED 点亮时最适合选择单片机 I/O 口输出_____电平。

2. 用 C51 编程访问 51 单片机 I/O 口，可以进行_____寻址和_____寻址。

3. 51 单片机内部无上拉电阻的并行 I/O 口为_____，第二功能最多的 I/O 口为_____。

4. 8 字形 LED 数码管不包括小数点段共计是_____段，当显示的 LED 数码管位数较多时，一般采用_____显示方式。

二、单项选择

1. 下列说法错误的是_____。

A. 51 单片机 I/O 口采用的是 TTL 电平

B. 51 单片机 I/O 口可通过继电器控制高电压设备

C. 有源蜂鸣器和无源蜂鸣器的区别是是否需要供电电源

D. 在键盘编程中常采用等待按键释放的处理方法来消除连击

2. 下列说法正确的是_____。

A. 51 单片机 I/O 口内部均有上拉电阻

B. P2 口作为地址输出线使用时输出外部存储器的高 8 位地址

C. 有源蜂鸣器和无源蜂鸣器的区别是是否需要供电电源

D. P3 口有串行数据输入输出、外部中断输入输出，定时器外部脉冲计数输入等复用功能

三、问答题

1. 简述多位数码管动态显示的原理。

2. 双向 I/O 口与准双向 I/O 口有什么区别，AT89C51 单片机哪些 I/O 口是双向 I/O 口，哪些是准双向 I/O 口？

四、编程题

1. 两排 LED 的初始状态如图 5-19 所示。

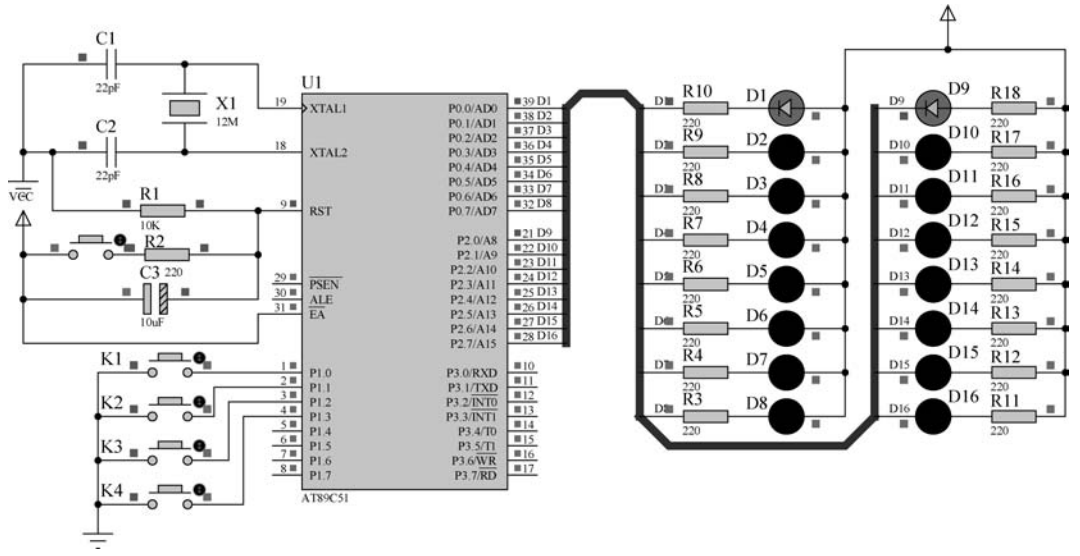

图 5-19　电路图

绘制电路并编程仿真实现如下功能：

（1）按下 K1 时，P0 端口控制的 LED 上移一位；

（2）按下 K2 时，P0 端口控制的 LED 下移一位；

（3）按下 K3 时，P2 端口控制的 LED 上移一位；

（4）按下 K4 时，P2 端口控制的 LED 下移一位。

要求用三种不同的方式编程，用 Keil 进行程序编制和编译，用 Proteus 进行仿真。

（1）用 if 语句实现。

（2）用 switch case 语句实现。（必做）

（3）编制一 P1 端口按键移动 LED 函数，并在主函数中调用来实现。

2. 设计 4 位数码管动态显示电路，并编制程序，在 4 位数码管上显示数字。

3. 在点阵屏上滚动循环显示“I♥U”。