# 第 5 章　并行接口

并行接口应用十分普遍，许多 I/O 设备乃至功能器件都使用并行接口与 CPU 进行连接和交换数据。目前，虽然有很多 I/O 设备使用 USB 串行总线进行连接，但在它们的信息进入微机系统时，仍然使用并行方式，所以，并行接口是最基本的接口形式。

本章对并行接口的特点进行分析，从应用的角度讨论通用接口芯片 PPI 和 GPIO 的外部特性与编程模型。在此基础上设计包括步进电机、扬声器、开关、指示灯、温度传感器等器件的并行接口以及并行存储器接口。

## 5.1　并行接口的特点

所谓“并行”不是针对接口与系统总线一侧的并行数据传输而言的，而是指接口与 I/O 设备一侧的并行数据传输。并行接口有如下基本特点。

(1) 以字节、字或双字宽度在接口与 I/O 设备之间的多根数据线上传输数据，因此数据传输速率高。

(2) 并行传输时，除数据线外，还需要地址线、控制线的支持，实际上，并行接口所使用的信号线是系统三总线的延伸。正因为如此，主机与外设之间进行数据传输时，通过地址线、数据线、控制线分别进行地址、数据、控制命令的传输，而不像串行接口那样把不同类型的信息混在一起。所以，并行方式不要求采用特殊的数据格式来分辨不同类型的信息。并行传输的数据是原始数据，而不是格式化的数据。

(3) 并行传输不要求固定的传输速率，传输速率由被连接或控制的 I/O 设备的操作要求决定。

(4) 在并行数据传输过程中，一般不做差错检验。

(5) 并行接口使用的信号线比较多，适用于近距离传输。

综上所述，并行接口不像串行接口那样通过各种通信协议与标准来规范通信双方共同遵守的数据格式、传输速率以及差错检验等操作规则。

从上述特点可以得知，并行接口是一种多线连接、使用自由、应用广泛、适于近距离传输的接口，是微机接口技术的基本内容。

## 5.2　PPI8255 并行接口

本节讨论 PPI8255 的目的是通过它了解并行接口的特点，并在采用它进行外设接口设计的过程中学习接口设计方法与步骤。PPI 是一个通用型、功能强且成本低的接口芯片，可与任意一个需要并行传输数据的 I/O 设备相连接。下面先分析它的外部特性与编程模型，然后讨论几种连接 I/O 设备和器件的并行接口设计。以后简称 PPI8255 为 PPI55。

### 5.2.1 PPI 接口芯片

**1. PPI55 的外部特性**

外部特性即芯片的引脚信号，用户利用这些引脚信号线进行接口的硬件设计。PPI55 的引脚信号如表 5.1 所示。

**表 5.1 PPI55 外部信号引脚定义**

| 引 脚 名 | 方 向 | 功 能 |
|---|---|---|
| $D_0 \sim D_7$ | 双向 | 数据线 |
| $\overline{CS}$ | 入 | 片选信号 |
| $A_1$、$A_0$ | 入 | 片选寄存器 |
| $\overline{RD}$ | 入 | 读信号 |
| $\overline{WR}$ | 入 | 写信号 |
| RESET | 入 | 复位信号 |
| $PA_0 \sim PA_7$ | 双向 | A 端口的 I/O 线 |
| $PB_0 \sim PB_7$ | 双向 | B 端口的 I/O 线 |
| $PC_0 \sim PC_7$ | 双向 | C 端口的 I/O 线 |

表 5.1 中的 3 个 8 位端口 A、B、C 都可作为数据口与外设之间交换数据，其他为读写控制线和地址线。

**2. PPI55 的工作方式**

PPI55 有 3 种工作方式：0 方式、1 方式、2 方式。其中，0 方式应用最多，2 方式很少应用。

1）0 方式的特点与功能

0 方式是一种无条件的数据传输方式，也是 PPI55 的基本输入输出方式。

**特点**：一次初始化只能把某个并行端口置成输入或输出，不能置成既输入又输出；不要求固定的联络（应答）信号，无固定的工作时序和固定的工作状态字；适用于以无条件或查询方式与 CPU 交换数据的情况，不能用于以中断方式交换数据的情况。因此，0 方式使用起来不受什么限制。

**功能**：A 端口和 B 端口作为数据端口，8 位并行传送；C 端口作为数据端口，4 位并行传送，分高 4 位和低 4 位，还可作为控制线，按位输出逻辑 1 或逻辑 0。

2）1 方式的特点与功能

1 方式是一种选通方式，双方传输数据时，需要遵守握手/应答的约定。

**特点**：一次初始化只能把某个并行端口置成输入或输出；要求专用的联络（握手/应答）信号，有固定的工作时序和专用的工作状态字；适用于以查询或中断方式与 CPU 交换数据的情况，不能用于以无条件方式交换数据的情况。

**功能**：A 端口和 B 端口作为数据端口，8 位并行传送；C 端口有以下 4 种功能：

（1）作为 A 端口和 B 端口的专用联络信号线。

（2）作为数据端口，未分配作为专用联络信号线的引脚可作为数据线。

（3）作为状态端口，读取 A 端口和 B 端口的状态字。

(4) 作为控制，按位输出逻辑 1 或逻辑 0。

3) 2 方式的特点与功能

2 方式是一种双向选通方式，即双方能够同时发送和接收数据。

**特点**：一次初始化可将 A 端口置成既输入又输出，具有双向性；要求有两对专用的联络信号，有固定的工作时序和专用的工作状态字；适用于以查询和中断方式与 CPU 交换数据的情况，特别是在要求与 I/O 设备进行双向数据传输时很有用。

**功能**：A 端口作为双向数据端口，8 位并行传送；B 端口作为单向数据端口，8 位并行传送；C 端口有 4 种功能，与 1 方式类似。

### 3. PPI55 的编程模型

PPI55 的编程模型包括内部寄存器、分配给寄存器的端口地址以及装入寄存器的命令字、状态字的格式。用户通过 PPI55 的编程模型进行并行接口的程序设计。

1) PPI55 内部寄存器及端口地址

内部设置命令寄存器、状态寄存器(从 C 口读出)以及 3 个独立的输入输出数据寄存器。

命令寄存器占用一个 I/O 端口，设为 CTRL55；3 个数据寄存器各占用一个 I/O 端口，分别设为 DATD_PA、DATD_PB、DATD_PC。

2) PPI55 的命令字格式

PPI55 有两个命令字，分别为工作方式命令和按位置位/复位命令，共用一个端口。下面讨论这两个命令的功能及格式。

(1) 工作方式命令。

工作方式命令又称初始化命令，用于对 PPI55 进行初始化。

**功能**：指定 PPI55 的工作方式及其工作方式下 3 个并行端口的输入或输出功能。

**格式**：该命令字的格式与含义如图 5.1 所示。

| $D_7$=1 | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 特征位 | A组方式<br>00=0 方式<br>01=1 方式<br>10=2 方式<br>11=不用 | | A端口<br>0=出<br>1=入 | $PC_4$~$PC_7$<br>0=出<br>1=入 | B组方式<br>0=0 方式<br>1=1 方式 | B端口<br>0=出<br>1=入 | $PC_0$~$PC_3$<br>0=出<br>1=入 |

图 5.1　PPI55 工作方式命令的格式与含义

最高位 $D_7$ 是特征位，因为 PPI55 有两个命令，用特征位加以区别。$D_7=1$，表示是工作方式命令；$D_7=0$，表示是按位置位/复位命令。

从工作方式命令的格式可知，A 组有 3 种工作方式(0 方式、1 方式、2 方式)，而 B 组只有两种工作方式(0 方式、1 方式)。C 端口分成两部分，上半部分属于 A 组，下半部分属于 B 组。对 3 个并行端口的输入输出设置是：置 1 指定为输入，置 0 指定为输出。

例如，把 A 端口指定为 1 方式，输入；把 C 端口上半部分指定为输出；把 B 端口指定为 0 方式，输出；把 C 端口下半部分指定为输入。工作方式命令代码是 10110001B=B1H。

若将此工作方式命令代码写到方式命令寄存器中，就实现了对 PPI55 的初始化，即工作方式及端口功能的指定。其程序语句如下：

```
outportb(CTRL55,0x0B1);
```

(2) 按位置位/复位命令。

按位置位/复位命令用于对 I/O 设备进行按位控制。

**功能**：指定 C 端口 8 个引脚中的任意一个引脚(只能一次指定一个引脚)输出高电平或低电平。

**格式**：按位置位/复位命令字的格式与含义如图 5.2 所示。

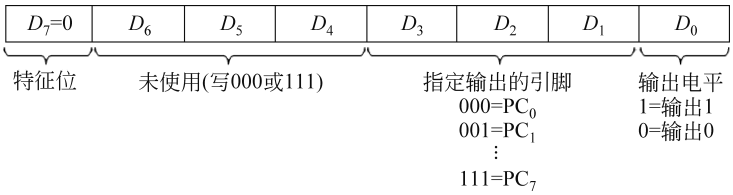


图 5.2　PPI55 按位置位/复位命令的格式与含义

利用按位置位/复位命令可以将 C 端口的 8 根线中的任意一根置成高电平输出或低电平输出，用于控制开关的通/断、继电器的吸合/释放、马达的启/停等操作。

例如，若命令 C 端口的 $PC_2$ 引脚输出高电平，去启动步进电机，则命令字应该为 00000101B＝05H。其程序语句如下：

```
Outputb(CTRL55,0x05);
```

如果要使 $PC_2$ 引脚输出低电平，去停止步进电机，则程序语句如下：

```
Outputb(CTRL55,0x04);
```

利用按位输出高、低电平的特性还可以产生正、负脉冲或方波输出，对 I/O 设备进行控制。

例如，利用 $PC_7$ 产生负脉冲，作为打印机接口电路的数据选通信号，其程序如下：

```
outportb(CTRL55,0x0e);
delay(10);
outportb(CTRL55,0x0f);
```

又如，利用 PPI82C55A 的 $PC_6$ 产生方波，送到喇叭，使其产生不同频率的声音，其程序段如下：

```
outportb(CTRL55,0x0d);              //写命令，置 PC6=1
delay(100);                         //调用延时程序，延时 100ms
outportb(CTRL55,0x0c);              //写命令，置 PC6=0
delay(100);
```

**注意**：按位置位/复位命令虽然是对 C 端口进行按位输出操作，但它不能写入作为数据口的 C 端口，只能写入命令口。原因是，它不是数据，而是命令，要按命令的格式来解释和执行。这一点初学者往往容易弄错，要特别留意。

### 5.2.2　PPI 接口实例：步进电机控制接口设计

**1. 要求**

设计一个四相六线式步进电机接口电路，要求按四相双八拍方式运行。当接通开关 $SW_2$ 时，步进电机开始运行；当接通开关 $SW_1$ 时，步进电机停止。

**2. 分析**

步进电机是将电脉冲信号转换成角位移的一种机电式 D/A 转换器。步进电机旋转的角位移与输入脉冲的个数成正比，步进电机的转速与输入脉冲的频率成正比，步进电机的转动方向与输入脉冲对绕组加电的顺序有关，因此，步进电机旋转的角位移、转速及方向均受输入脉冲的控制。

1）运行方式与运行方向控制

步进电机的运行方式是指各绕组循环轮流通电的方式。例如，四相步进电机有单四拍、双四拍、单八拍、双八拍 4 种运行方式，如图 5.3 所示。

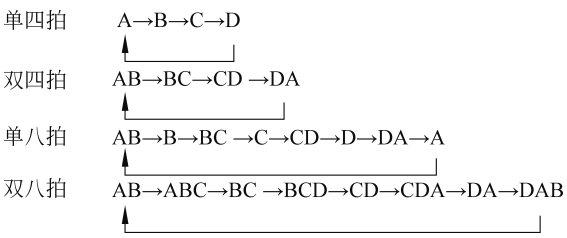


图 5.3　四相步进电机运行方式

步进电机的运行方向是指正转（顺时针）或反转（逆时针）。

为了实现对各绕组按一定方式轮流加电，需要一个脉冲循环分配器。脉冲循环分配器可用硬件实现，也可以软件实现，本例采用循环查表法来实现对运行方式与运行方向的控制。

循环查表法是将各绕组加电顺序的控制代码制成一张步进电机相序表，存放在内存区，再设置一个地址指针。当地址指针依次加 1（或减 1）时，即可从相序表中取出加电代码，然后输出到步进电机，使之产生一定运行方式的旋转。若改变相序表内的加电代码和地址指针的指向，则可改变步进电机的运行方式与运行方向。

表 5.2 是步进电机四相双八拍运行方式的一种相序表。若运行方式发生改变，则加电代码也会改变。

在表 5.2 所示的相序表中，若把地址指针从 400H 单元开始，依次加 1，取出加电代码去控制步进电机的运行方向定为正方向，那么，地址指针从 407H 单元开始，依次减 1 的方向就是反方向。表 5.2 中的地址指针是随机给定的，在程序中要定义一个变量（数组）以指出相序表的首址。

可见，对步进电机运行方式的控制是通过改变相序表中的加电代码实现的，而运行方向的控制是通过设置相序表的指针实现的。

表 5.2　步进电机四相双八拍运行方式相序表

| 绕组与数据线的连接 | | | | | | | | 运行方式 | 相　序 | |
|---|---|---|---|---|---|---|---|---|---|---|
| D | | C | | B | | A | | 四相双八拍 | 加电代码 | 地址指针 |
| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | | | |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | AB | 05H | 400H |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | ABC | 15H | 401H |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | BC | 14H | 402H |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | BCD | 54H | 403H |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | CD | 50H | 404H |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | CDA | 51H | 405H |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | DA | 41H | 406H |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | DAB | 45H | 407H |

2）运行速度的控制

控制步进电机运行速度有两种途径：一是利用硬件改变输入脉冲的频率，即对定时器（如 PPI82C54A）定时常数进行设定；二是软件延时，即调用延时子程序。

采用软件延时方法改变步进电机速度，虽然简便易行，但延时受 CPU 主频的影响，在主频较低的微机上开发的步进电机控制程序在主频较高的微机上就不能正常运行，甚至由于实际主频太高，使步进电机干脆不动了。

3）步进电机的驱动

步进电机在系统中是一种执行元件，都要带负载，因此需要功率驱动。在电子仪器和设备中，一般所需功率较小，常采用达林顿管，如采用 TIP122 作为功率驱动级，其驱动原理如图 5.4 所示。

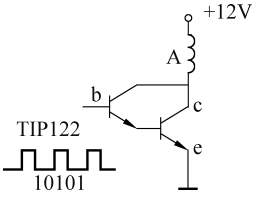

图 5.4　步进电机驱动原理

在图 5.4 中，在 TIP122 的基极上所加电脉冲为高。即，加电代码为 1 时，达林顿管导通，使绕组 A 通电；加电代码为 0 时，绕组 A 断电。

4）步进电机的启/停控制

为了控制步进电机的启/停，通常采用设置硬开关或软开关的方法。所谓硬开关，一般是在外部设置按键开关 SW，并且约定当开关 SW 按下时启动或停止步进电机运行；所谓软开关，就是在系统的键盘上定义某个键，当该键按下时启动或停止步进电机运行。

**3. 设计**

1）硬件设计

根据设计要求，需要使用 3 个端口。

- A 端口为输出，向步进电机的 4 个绕组发送加电代码（相序码），以控制步进电机的运行方式。
- C 端口的高 4 位（$PC_4$）为输出，控制 74LS373 的开关，起隔离作用。当步进电机不工

作时，关闭 74LS373，以使电机在停止运行后不会因为 PPI82C55A 的漏电流引起发热而烧坏。

- C 端口的低 4 位（$PC_0$ 和 $PC_1$）为输入，分别与开关 $SW_2$ 和 $SW_1$ 连接，以控制步进电机的启动和停止。

步进电机控制接口原理如图 5.5 所示。

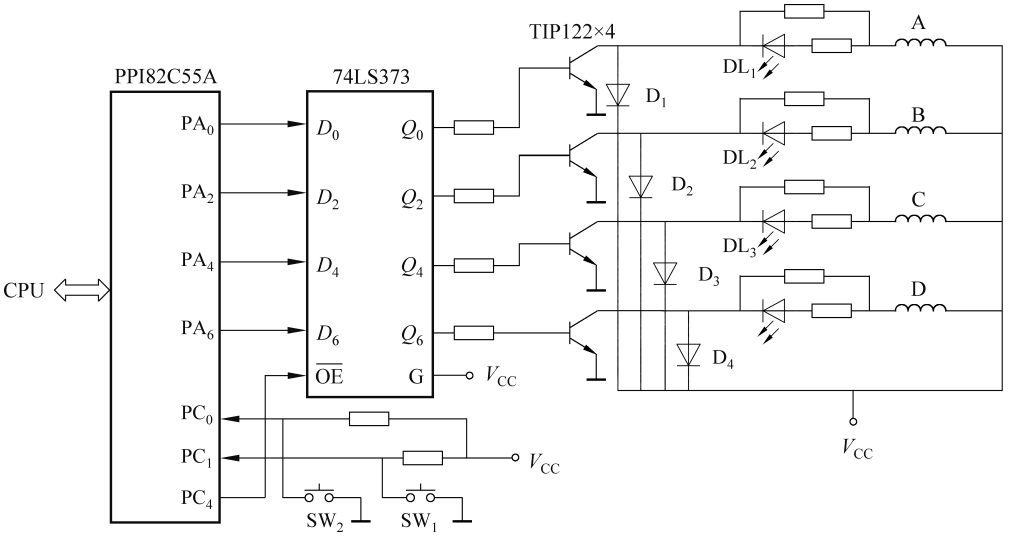


图 5.5　步进电机控制接口原理图

2）软件设计

在开环控制方式下，四相步进电机的启停操作可以随时进行，是一种无条件的并行传送方式。控制程序包括相序表和相序指针的设置、PPI55 初始化、步进电机启停控制、相序代码传送以及电机的保护措施等内容。

步进电机控制程序如下：

```
#define CTRL55 0x303
#define DATA_PA 0x300
#define DATA_PB 0x301
#define DATA_PC 0x302
#include<stdio.h>
#include<conio.h>
#include<dos.h>
void main()
{
    unsigned char PSTA[8]={0x05,0x15,0x14,0x54,0x50,0x51,0x41,0x45};
    printf("Hit SW2 to Start,Hit SW1 to Quit!\n");
    outportb(CTRL55,0x81);                    //初始化 PPI82C55A
    outportb(CTRL55,0x09);                    //关闭 74LS373(置 PC4=1),保护步进电机
    while(inportb(DATA_PC)&0x01);             //检测开关 SW2 是否按下(PC0=0?)
```

```
    outportb(CTRL55,0x08);                    //打开 74LS373(置 PC4=0),进行启动控制
    while(1)
    {
        for(i=0;i<8;i++)
        {
            outportb(DATA_PA,PSTA[i]);        //传送相序代码
            delay(100);                       //延时,进行速度控制
            if((inportb(DATA_PC)&0x02)==0)    //检测开关 SW1 是否按下(PC1=0?)
            {
                outportb(CTRL55,0x09);        //关闭 74LS373(置 PC4=1),保护步进电机
                return;                       //返回 DOS
            }
        }
        if(i==8)                              //已到 8 次,重新赋值
            i==0;
    }
}
```

**4. 讨论**

开环运行的步进电机一般需要控制以下 6 方面：

(1) 运行方式。四相步进电机有 4 种运行方式，采用构造相序表的方法实现不同运行方式的要求。

(2) 运行方向。控制步进电机的正反方向，采用把相序表的地址指针设置在表头或表尾来确定运行方向。

(3) 运行速度。控制步进电机的快慢。可以采用在延时程序中改变延时常数的方法来实现，也可以用硬件方法实现。

(4) 运行花样。有点动、先正后反、先慢后快、走走停停等花样。

(5) 启停控制。设置开关，包括设置硬开关和软开关两种方法。

(6) 保护措施。在步进电机与接口电路之间设置隔离电路，如具有三态的 74LS373。

试分析本例实现了哪几项控制，并指出实现的每一项控制的相应程序段或程序行。

### 5.2.3 PPI 接口实例：声光报警器接口设计

**1. 要求**

设计一个声光报警器，要求：按下按钮开关 SW，开始报警，喇叭 SPK 发声，LED 灯同时闪光；当拨通 DIP 拨动开关的 0 位($DIP_0$)时，结束报警，喇叭停止发声，LED 灯熄灭。

**2. 分析**

根据要求，该声光报警器包括 4 种简单的器件：喇叭、8 个 LED 灯、DIP 拨动开关及按钮开关 SW。它们都是并行接口的对象，虽然功能单一、结构简单，但都必须通过接口电路才能进入微机系统，受 CPU 的控制，发挥相应的作用。

**3. 设计**

1）硬件设计

根据设计要求，需要使用 3 个端口：

- A 端口为输出，连接 8 个 LED 灯（$LED_0$～$LED_7$）。
- B 端口为输入，连接 DIP 开关的 8 位（$DIP_0$～$DIP_7$）。
- C 端口的 $PC_6$ 为输出，连接喇叭 SPK；C 端口的 $PC_2$ 为输入，连接按钮开关 SW。

声光报警器电路原理如图 5.6 所示。

2）软件设计

声光报警器程序流程图如图 5.7 所示。

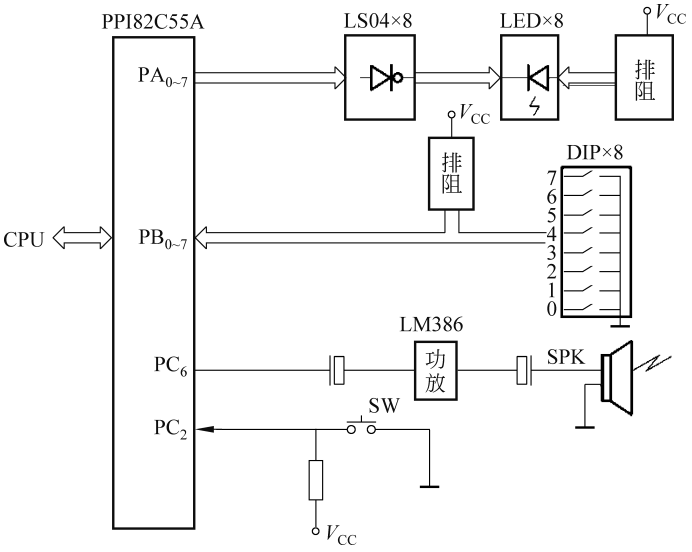


图 5.6　声光报警器电路原理

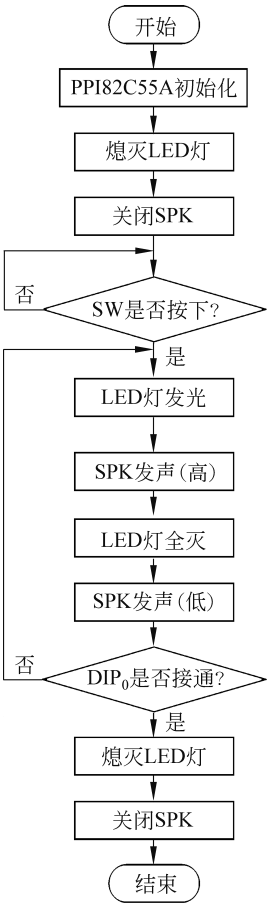


图 5.7　声光报警器程序流程图

声光报警器控制程序如下：

```
#define CTRL55 0x303
#define DATA_PA 0x300
```

```
#define DATA_PB 0x301
#define DATA_PC 0x302
#include<stdio.h>
#include<conio.h>
#include<dos.h>
void OutSpk(unsigned int time);
void main()
{
    unsigned char tmp,i=0;
    outportb(CTRL55,0x83);   //0方式,A端口和PC4~PC7为输出,B端口和PC0~PC3为输入
    outportb(DATA_PA,0x00);          //LED灯全灭(PA0~PA7全部置0)
    outportb(CTRL55,0x0c);           //关闭SPK(置PC6=0)
    while(inportb(DATA_PC)&04);      //判断SW是否按下(PC2=0?)
    while(inportb(DATA_PB)&0x01)     //判断DIP0是否按下?(PB0=0?)
    {
        outportb(DATA_PA,0x0ff);     //LED灯全亮
        for(i=0;i<200;i++)
        {
            OutSpk(30);              //调用喇叭发声(高频)子程序
        }
        outportb(DATA_PA,0x00);      //LED灯全灭
        for(i=0;i<200;i++)
        {
            OutSpk(270);             //调用喇叭发声(低频)子程序
        }
        delay(600);
    };
    outportb(DATA_PA,0x00);          //LED灯全灭
    outportb(CTRL55,0x0c);           //关闭SPK
}
void OutSpk(unsigned int time)
{
    outportb(CTRL55,0x0d);
    delay(time);
    outportb(CTRL55,0x0c);
    delay(time);
}
```

**4. 讨论**

从本例的电路还可以派生出多种应用,例如:

(1) LED走马灯花样(点亮花样)程序。利用DIP开关控制LED灯产生8种走马灯花样。例如,将DIP开关的1号开关合上时,8个LED灯从两端向中间依次点亮;将2号开关合上时,LED灯从中间向两端依次点亮。按下按钮开关SW时,LED灯熄灭。实现方法为: