项目二　LED 循环点亮控制

终极目标

能完成单片机的输入输出电路设计，能应用 C 语言程序完成单片机输入输出控制，实现对 LED 循环点亮控制的设计、运行及调试。

促成目标

1. 掌握 P0、P1、P2 和 P3 功能及应用技能;
2. 掌握内部数据存储器的地址分配及特殊功能寄存器;
3. 掌握 C 语言的数据类型、常量和变量;
4. 会利用单片机 I/O 口实现开关控制 LED 循环点亮和步进电机控制。

2.1　工作模块 3　LED 循环点亮控制

使用 AT89S52 单片机，P1 口引脚接 8 个 LED 的阴极，通过程序按一定的规律向 P1 口的引脚输出低电平和高电平，控制 8 个发光二极管循环点亮。

2.1.1　LED 循环点亮电路设计

按照工作任务要求，LED 循环点亮电路是由单片机最小应用系统和 8 个 LED 电路构成，如图 2-1 所示。

8 个 LED 采用共阳极接法，LED 的阳极通过 220Ω限流电阻后连接到 5V 电源上，P1 口接 LED 的阴极。P1 口的引脚输出低电平时对应的 LED 点亮，输出高电平时对应的 LED 熄灭。

LED 循环点亮 Proteus 仿真电路设计过程与工作模块 1 基本一样，在以后工作模块中只写出不同的设计过程，不再详细叙述具体设计过程。

（1）新建设计文件，设置图纸尺寸，设置网格，保存设计文件。文件名为“LED 循环点亮”。

（2）选取元器件。从 Proteus 元器件库中选取元器件：AT89S52（单片机）、CRYSTAL（晶振）、CAP（电容）、CAP-ELEC（电解电容）、RES（电阻）、LED-RED（红色发光二极管）。

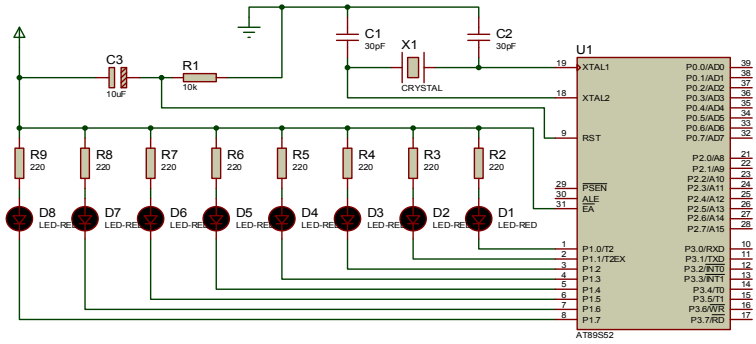


图 2-1　LED 循环点亮电路

（3）放置元器件，编辑元器件，放置终端，连线。按图 2-1 所示放置元器件并连线。

（4）属性设置。先右击后单击元器件电容 C1，在弹出的 Edit Component 对话框中将电容量改为 30pF，单击 OK 按钮完成元器件电容 C1 的属性编辑。同样的方法编辑其他元器件属性。

（5）电气规则检测。单击“工具”→“电气规则检查”命令，弹出检查结果窗口，完成电气检测。若检测出错，根据提示修改电路图并保存，直至检测成功。

2.1.2　LED 循环点亮程序设计

LED 循环点亮电路设计完成以后，我们还不能看到 LED 循环点亮的现象，还需要编写程序控制单片机引脚电平的高低变化，来控制 LED 的亮灭，实现 LED 循环点亮。

1. LED 循环点亮功能实现分析

由于 LED 循环点亮电路的 LED 是采用共阳极接法，这样我们就可以通过“0”和“1”来控制 LED 的亮和灭。例如：在 P1 口输出十六进制数 0xfe（二进制 11111110B），D1 被点亮。若 P0 口输出 0x7f（二进制 01111111B），则 D8 被点亮。LED 循环点亮功能的实现过程如下：

（1）8 个 LED 全灭，控制码为 0xff;

（2）D1 点亮，P1 口输出 0xfe，取反为 0x01（二进制 00000001B），初始控制码为 0x01；

（3）D2 点亮，P1 口输出 0xfd，取反为 0x02（二进制 00000010B），控制码为 0x02；

（4）D3 点亮，P1 口输出 0xfb，取反为 0x04（二进制 00000100B），控制码为 0x04；

……

（5）D8 点亮，P1 口输出 0x7f，取反为 0x80（二进制 10000000B），控制码为 0x80；

（6）重复第二步，这样就可以实现 LED 循环点亮。

2. LED 循环点亮控制程序设计

从以上分析可以看出，首先使所有的 LED 都熄灭，然后将控制码取反后，从 P1 口输出，点亮相应的 LED。控制码左移一位，即可获得下一个控制码。

LED 循环点亮控制 C 语言程序如下：

```
#include <AT89X52.H>                    //包含 AT89X52.H 头文件
void  Delay()                           //延时函数
{
  unsigned char i, j;
     for (i=0;i<255;i++)
        for (j=0;j<255;j++);
}
void main()
```

```
{
   unsigned char i;
   unsigned char temp;
   P1 = 0xff;                          //十六进制全 1,熄灭所有 LED
   while(1)
   {
     temp = 0x01;                      //第一位为 1，即初始控制码为 0x01
     for (i=0;i<8;i++)
         {
           P1 = ~ temp;                //temp 值取反送 P1 口
           Delay();
           temp = temp << 1 ;          //temp 值左移一位，获得下一个控制码
         }
    }
 }
```

程序开始时将初始控制码取反后从 P1 口输出，这个数本身是让 P1.0 为低电平，其他位为高电平，点亮 D1，然后延时一段时间，再让控制码左移一位，获得下一个控制码，然后再对控制码取反后输出到 P1 口，这样就实现“LED 循环点亮”效果。

在此还要再次强调，由于人眼的视觉暂留效应以及单片机执行每条指令的时间很短，在控制 LED 亮灭的时候应该延时一段时间，否则就看不到“LED 循环点亮”效果了。

2.1.3　并行 I/O 端口电路

单片机有 4 组 8 位并行 I/O 端口，称为 P0 口、P1 口、P2 口和 P3 口，每个端口都各有 8 条 I/O 口线，每条 I/O 口线都能独立地用作输入或输出。P0 口负载能力为 8 个 TTL 门电路，P1 口、P2 口和 P3 口负载能力为 4 个 TTL 门电路。实际上，它们已被归入特殊功能寄存器之列，并且具有字节寻址和位寻址功能。

1．P0 口

P0 口某位结构图如图 2-2 所示，它由一个数据输出锁存器（D 触发器）、两个三态数据输入缓冲器、一个输出控制电路和一个输出驱动电路组成。输出控制电路由一个转换开关 MUX、一个与门及一个非门组成，输出驱动电路由一对场效应管（V1 和 V2）组成，其工作状态受输出控制端的控制。

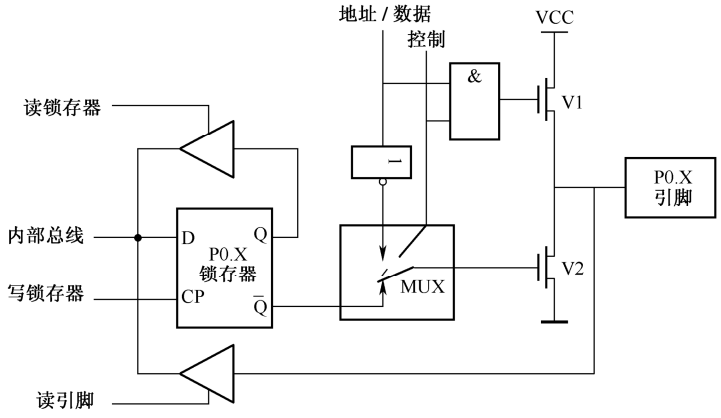


图 2-2　P0 口某位结构图

P0 口有两种功能：通用 I/O 口和地址/数据分时复用总线。

（1）P0 口作通用 I/O 口使用。

作为通用的 I/O 口使用时，内部的控制信号为低电平，封锁与门，将输出驱动电路的上拉

场效应管（V1）截止，同时使多路转接电路 MUX 接通锁存器 Q 端的输出通路。

注意

当 P0 口进行一般的 I/O 输出时，由于输出电路是漏极开路电路，因此必须外接上拉电阻才能有高电平输出；当 P0 口进行一般的 I/O 输入时，必须先向电路中的锁存器写入“1”，使场效应管（V2）截止，以避免锁存器为“0”状态时对引脚读入的干扰，因为如果 V2 管是导通的，不论 P0.X 引脚上的状态如何，输入都会是低电平，将导致输入错误。

（2）P0 口作地址/数据分时复用总线使用。

当输出地址或数据时，由内部发出控制信号，打开上面的与门，并使多路转接电路 MUX 将内部地址/数据线与驱动场效应管（V2）接通。输出驱动电路由于上、下两个驱动场效应管处于反相，形成推拉式电路结构，使负载能力大为提高。

若地址/数据线为 1，则 V1 导通，V2 截止，P0 口输出为 1；反之 V1 截止，V2 导通，P0 口输出为 0。当输入数据时，读引脚使三态数据输入缓冲器打开，数据信号则直接从引脚通过数据输入缓冲器进入内部总线。

2. P1 口

P1 口某位结构图如图 2-3 所示，P1 口是一个双向 8 位 I/O 口，每一位均可单独定义为输入口或输出口。

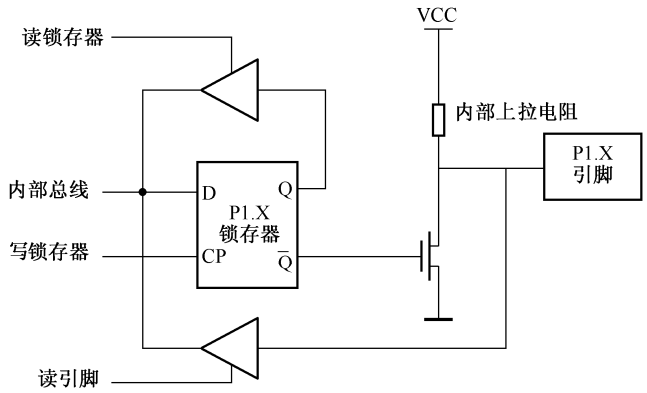


图 2-3 P1 口某位结构图

P1 口通常作为通用 I/O 口使用，在电路结构上与 P0 口有一些不同之处：首先它不再需要多路转接电路 MUX；其次是电路的内部有上拉电阻，与场效应管共同组成输出驱动电路。为此，P1 口作为输出口使用时，已经能向外提供推拉电流负载，无需再外接上拉电阻。

当作为输出口时，1 写入锁存器，$\overline{Q}$=0，场效应管截止，内部上拉电阻将电位拉至“1”，此时该口输出为 1，当 0 写入锁存器，$\overline{Q}$=1，场效应管导通，输出则为 0。当作为输入口时，必须先向锁存器写 1，$\overline{Q}$=0，场效应管截止，此时该位既可以把外部电路拉成低电平，也可由内部上拉电阻拉成高电平。

3. P2 口

P2 口某位结构图如图 2-4 所示，它由一个数据输出锁存器（D 触发器）、两个三态数据输入缓冲器、一个转换开关 MUX、一个数据输出驱动电路和控制电路组成。

P2 口电路比 P1 口电路多了一个多路转接电路 MUX，这又正好与 P0 口一样。P2 口可以作为通用 I/O 口使用，这时多路转接电路开关倒向锁存器 Q 端。在实际应用中，P2 口通常作

为高 8 位地址线使用，此时多路转接电路开关应倒向相反方向。

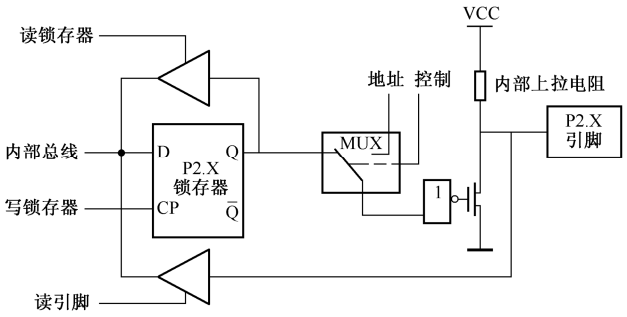


图 2-4　P2 口某位结构图

在无外部扩展存储器的系统中，4 个 I/O 口都可以作为通用 I/O 口使用。

在有外部扩展存储器的系统中，P2 口送出高 8 位地址 AB8～AB15，P0 口分时送出低 8 位地址 AB0～AB7 和 8 位数据 D0～D7。由于有 16 位地址，MCS-51 单片机最大可外接 64KB 的程序存储器和数据存储器。

4. P3 口

P3 口某位结构图如图 2-5 所示，P3 口除了作为通用 I/O 口使用外，每一根线还具有第二种功能。

P3 口的第一功能和 P1 口一样可作为输入输出端口，同样具有字节操作和位操作两种方式，每一位均可定义为输入或输出。

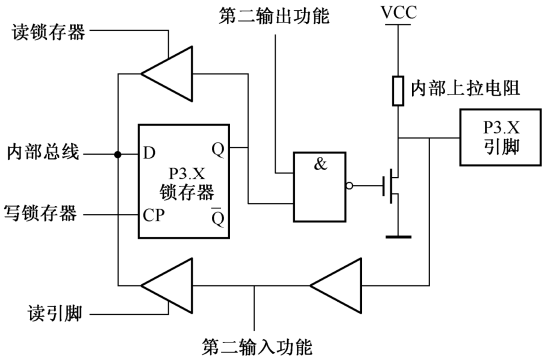


图 2-5　P3 口某位结构图

P3 口的特点在于，为适应引脚的第二功能的需要，增加了第二功能控制逻辑，在真正的应用电路中，第二功能显得更为重要。由于第二功能信号有输入和输出两种情况，因此我们分两种情况加以说明。

对于第二功能为输出的信号引脚，当作为 I/O 使用时，第二功能信号引线应保持高电平，与非门开通，以维持从锁存器到输出端数据输出通路的畅通。当输出第二功能信号时，该位的锁存器应置“1”，使与非门对第二功能信号的输出是畅通的，从而实现第二功能信号的输出。

对于第二功能为输入的信号引脚，在口线的输入通路上增加了一个缓冲器，输入的第二功能信号就从这个缓冲器的输出端取得。而作为 I/O 使用的数据输入，仍取自三态缓冲器的输出端。不管是作为输入口使用还是作为第二功能信号输入，输出电路中的锁存器输出和第二功能输出信号线都应保持高电平。

P3 口的第二功能定义如表 2-1 所示。

表 2-1　P3 口的第二功能定义

引脚	第二功能
P3.0	RXD：串行口输入
P3.1	TXD：串行口输出
P3.2	$\overline{\text{INT0}}$：外部中断 0 输入
P3.3	$\overline{\text{INT1}}$：外部中断 1 输入
P3.4	T0：定时器/计数器 0 计数输入
P3.5	T1：定时器/计数器 1 计数输入
P3.6	$\overline{\text{WR}}$：外部数据存储器写选通（输出）
P3.7	$\overline{\text{RD}}$：外部数据存储器读选通（输出）

【技能训练 2-1】P0 口外接上拉电阻

工作模块 3 是通过程序按一定规律向 P1 口的引脚输出低电平和高电平来实现 LED 循环点亮的。在这里如果通过 P0 口完成 LED 循环点亮，那么如何实现呢？

1. P0 口和 P1 口对比分析

（1）由于 P0 口的输出电路是漏极开路电路，所以在进行输出时，必须外接上拉电阻才能有高电平输出；

（2）由于电路的内部有上拉电阻，所以 P1 口在作为输出口使用时，无需再外接上拉电阻。

2. P0 口 LED 电路设计

本电路设计和模块 3 中 LED 循环点亮电路基本一样，差别是：使用了排阻、P0 口接 LED 的阳极以及在 P0 口和 LED 阳极之间外接了上拉电阻，如图 2-6 所示

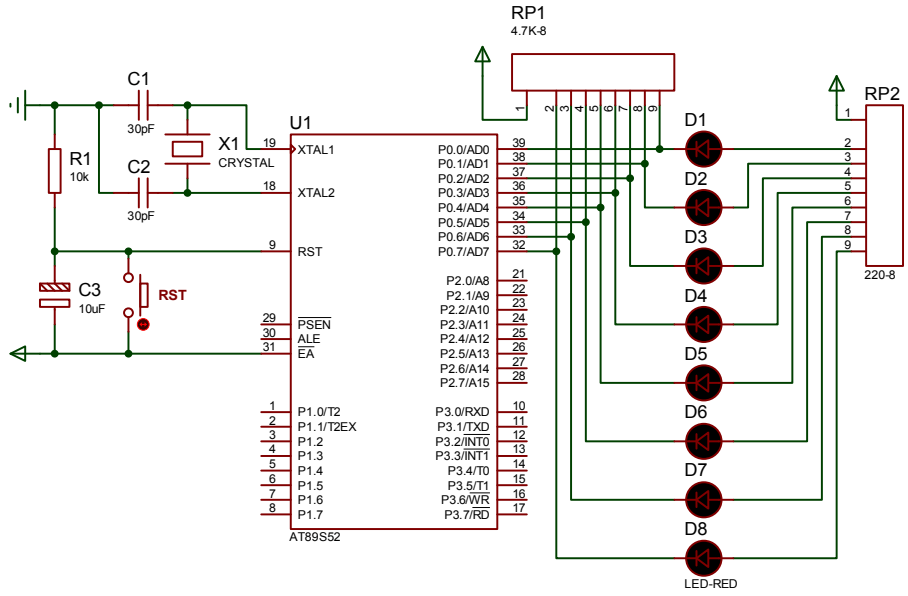


图 2-6　P0 口 LED 循环点亮电路

8 个电阻的功能是完全一样的，加工到一个器件里面，这个器件通常称为排阻。为了在电路板上占很小的地方，方便安装和生产，在电路设计时常常选择排阻。

PR1 和 PR2 都是排阻，阻值分别为 4.7kΩ*8 和 220Ω*8。PR1 排阻是上拉电阻，其功能是在这个引脚没有信号的时候，起到电位上拉的作用。PR2 和普通的电阻用途没有任何不同，在这里面起到限流作用，使通过 LED 的电流被限制在十几毫安左右。

3. P0 口 LED 程序设计

本程序设计和模块 3 中 LED 循环点亮程序基本一样，差别如下：

```
void main()
{
   ……
   P0 = 0x0;                    //熄灭所有 LED
   ……
   P0 = temp;                   //temp 值送 P0 口
   ……
 }
```

2.2　MCS-51 单片机内存空间

微型计算机通常只有一个逻辑空间，程序存储器 ROM 和数据存储器 RAM 都要统一编址，即一个存储器地址对应一个唯一存储单元。单片机是将程序存储器和数据存储器分开，它们有各自的寻址系统、控制信号和功能。

MCS-51 单片机内部集成了一定容量的程序存储器（8031/8032 等除外）和数据存储器，同时还具有强大的外部存储器扩展能力，MCS-51 单片机存储器的配置图如图 2-7 所示。

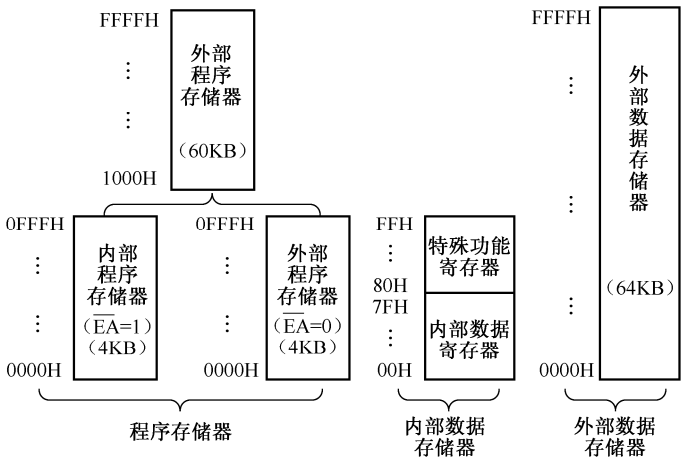


图 2-7　MCS-51 单片机存储器配置图

MCS-51 单片机存储器在物理结构上可分为 4 个存储空间：内部数据存储器、内部程序存储器、外部数据存储器和外部程序存储器。从逻辑上分，即从用户使用的角度看，MCS-51 单片机存储器分为 3 个逻辑空间：片内外统一编址的 64KB 程序存储器地址空间、256B 或 384B 的内部数据存储器地址空间和 64KB 外部数据存储器地址空间。

2.2.1 数据存储器

数据存储器也称为随机存取数据存储器，用于存放执行的中间结果和过程数据。MCS-51 的数据存储器均可读写，部分单元还可以位寻址。

数据存储器分为内部数据存储和外部数据存储两种，无论在物理上还是逻辑上，其地址空间是彼此独立的。内部数据存储地址范围为 00H～FFH，外部数据存储最多可扩展 64KB，其地址范围为 0000H～FFFFH。

内部数据存储器在物理上和逻辑上都分为两个地址空间：00H～7FH 单元组成的低 128 字节数据存储器空间和 80H～FFH 单元组成的高 128 字节特殊功能寄存器空间。

在 51 子系列内部真正可作数据存储器用的只有低 128 字节数据存储器空间，地址为 00H～7FH，如图 2-8 所示。它们划分为 3 个区域：通用寄存器区、位寻址区和用户 RAM 区，这两个空间是相连的。

在 52 子系列中，有 256 字节数据存储器空间，高 128 字节数据存储器空间与特殊功能寄存器空间的地址是重叠的，都是 80H～FFH，如图 2-8 所示。

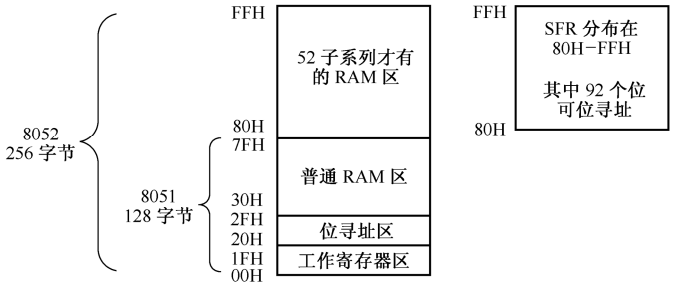


图 2-8　MCS-51 内部 RAM 功能配置图

1. 通用寄存器区（00H ~ 1FH）

由 32 字节的 RAM 单元组成，地址为 00H～1FH，分成 4 个工作区，每个区由 8 个 8 位通用寄存器组成，8 个通用寄存器 R0、R1、R2、R3、R4、R5、R6、R7 与 8 个存储单元一一对应。

这 4 组通用寄存器都称为 R0～R7，那么在程序中怎么辨别和使用它们呢？选择使用的工作区是由程序状态字 PSW 中的第三位 RS0 和第四位 RS1 确定的，RS1、RS0 可通过程序置 1 或清 0，以达到选择不同工作区的目的，其余的可用作一般的数据缓冲器，如表 2-2 所示。

表 2-2　通用寄存器选择

工作寄存器区	RS1　RS0	地址
0 区	0　0	00H～07H
1 区	0　1	08H～0FH
2 区	1　0	10H～17H
3 区	1　1	18H～1FH

4 组通用寄存器工作区给软件设计带来极大方便，在实现中断嵌套时可以灵活选择不同的通用寄存器工作区，实现现场保护。CPU 在复位后，默认选择第 0 组通用寄存器。

2. 位寻址区（20H ~ 2FH）

内部 RAM 的 20H～2FH 单元为位寻址区，既可作为一般单元用字节寻址，也可对它们的位进行寻址。位寻址区共有 16 字节，128 位，位地址为 00H～7FH，如图 2-9 所示。

直接地址									
2FH	7F	7E	7D	7C	7B	7A	79	78	
2EH	77	76	75	74	73	72	71	70	
2DH	6F	6E	6D	6C	6B	6A	69	68	
2CH	67	66	65	64	63	62	61	60	
2BH	5F	5E	5D	5C	5B	5A	59	58	
2AH	57	56	55	54	53	52	51	50	
29H	4F	4E	4D	4C	4B	4A	49	48	
28H	47	46	45	44	43	42	41	40	位寻址区
27H	3F	3E	3D	3C	3B	3A	39	38	
26H	37	36	35	34	33	32	31	30	
25H	2F	2E	2D	2C	2B	2A	29	28	
24H	27	26	25	24	23	22	21	20	
23H	1F	1E	1D	1C	1B	1A	19	18	
22H	17	16	15	14	13	12	11	10	
21H	0F	0E	0D	0C	0B	0A	9	8	
20H	7	6	5	4	3	2	1	0	

图 2-9　内部 RAM 中位地址

CPU 能直接寻址这些位，执行如置“1”、清“0”、求“反”、转移、传送和逻辑等操作。我们常称 MCS-51 具有布尔处理功能，布尔处理的存储空间指的就是这些位寻址区。

3. 用户 RAM 区（30H ~ 7FH）

在内部 RAM 低 128 个单元中，通用寄存器占去 32 个单元，位寻址区占去 16 个单元，剩下的 80 个单元就是供用户使用的一般 RAM 区了，地址单元为 30H～7FH，这些 RAM 单元只能按字节寻址。对这部分区域的使用没有任何规定或限制，一般应用中常把堆栈开辟在此区中。单片机在复位时，SP 的初值为 07H，可在初始化程序时设定 SP 的值，确定堆栈区的范围，通常情况下将堆栈区设在 30H～7FH 范围之内。

2.2.2　特殊功能寄存器

高 128 个单元是供特殊功能寄存器使用的，也称为专用寄存器，单元地址为 80H～FFH。MCS-51 单片机把 CPU 中的专用寄存器、并行端口锁存器、串行口与定时器/计数器内的控制寄存器集中安排到一个区域，离散地分布在地址 80H～FFH 范围内，这个区域称为特殊功能寄存器（SFR）区。特殊功能寄存器字节地址分配如图 2-10 所示。

同时某些 SFR 寄存器还可以位寻址，即对这些 SFR 寄存器 8 位中的任何一位进行单独的位操作，这一点与 20H～2FH 中的位操作是完全相同的。在 SFR 中有 12 个特殊功能寄存器的字节地址能被 8 整除，这 12 个特殊功能寄存器的 93 位具有位寻址功能，有 3 个未定义位。

特殊功能寄存器最低位的位地址与特殊功能寄存器的字节地址相同，次低位的位地址等于特殊功能寄存器的字节地址加 1，依此类推，最高位的位地址等于特殊功能寄存器的字节地址加 7。特殊功能寄存器的位地址分配如图 2-11 所示。

复位后内部各寄存器的数据值，如图 2-12 所示。

寄存器符号	名称		字节地址
ACC	累加器		E0H
B	B 寄存器		F0H
PSW	程序状态字		D0H
SP	堆栈指针		81H
DPTR	数据指针	DPH	83H
		DPL	82H
P0	P0 口锁存器		80H
P1	P1 口锁存器		90H
P2	P2 口锁存器		A0H
P3	P3 口锁存器		B0H
IP	中断优先级控制寄存器		B8H
IE	中断允许控制寄存器		A8H
TMOD	定时器 / 计数器方式控制寄存器		C8H
TCON	定时器 / 计数器控制寄存器		88H
TH0	定时器 / 计数器 0（高字节）		8CH
TL0	定时器 / 计数器 0（低字节）		8AH
TH1	定时器 / 计数器 1（高字节）		8DH
TL1	定时器 / 计数器 1（低字节）		8BH
SCON	串行控制寄存器		98H
SBUF	串行数据缓冲器		99H
PCON	电源控制寄存器		97H

图 2-10　特殊功能寄存器

SFR	MSB	位地址 / 位定义						LSB	字节地址
B	F7	F6	F5	F4	F3	F2	F1	F0	F0H
ACC	E7	E6	E5	E4	E3	E2	E1	E0	E0H
PSW	D7	D6	D5	D4	D3	D2	D1	D0	D0H
	CY	AC	F0	RS1	RS0	OV	–	P	
IP	BF	BE	BD	BC	BB	BA	B9	B8	B8H
	–	–	–	PS	PT1	PX1	PT0	PX0	
P3	B7	B6	B5	B4	B3	B2	B1	B0	B0H
	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	
IE	AF	AE	AD	AC	AB	AA	A9	A8	A8H
	EA	–	–	ES	ET1	EX1	ET0	EX0	
P2	A7	A6	A5	A4	A3	A2	A1	A0	A0H
	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
SCON	9F	9E	9D	9C	9B	9A	99	98	98H
	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	
P1	97	96	95	94	93	92	91	90	90H
	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	
TCON	8F	8E	8D	8C	8B	8A	89	88	88H
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
P0	87	86	85	84	83	82	81	80	80H
	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	

图 2-11　SFR 中的位地址

寄存器	复位状态	寄存器	复位状态
PC	0000H	ACC	00H
B	00H	PSW	00H
SP	07H	DPTR	0000H
P0～P3	0FFH	IP	×××00000B
IE	0××00000B	TMOD	00H
TCON	00H	TL0，TL1	00H
TH0，TH1	00H	SCON	00H
SBUF	不定	PCON	0×××00000B

图 2-12　复位后 SFR 的值

1. ACC 累加器

累加器是一个最常用的特殊功能寄存器，是实现各种寻址及运算的寄存器，而不是一个仅做加法的寄存器，在 MCS-51 指令系统中所有算术运算、逻辑运算几乎都要使用它。对程序存储器和外部数据存储器的访问只能通过它进行。

2. B 寄存器

B 是一个 8 位特殊功能寄存器，在做乘除运算时要使用它，它也是一个有特殊功能的寄存器。

3. PSW 程序状态字

PSW 是一个 8 位特殊功能寄存器，用于存放程序运行中的各种状态信息，如图 2-13 所示。

位地址	D7	D6	D5	D4	D3	D2	D1	D0
符号	CY	AC	F0	RS1	RS0	OV	—	P

图 2-13　PSW 程序状态字

（1）CY（PSW.7）：高位进位标志位。当执行算术运算时，最高位向前进位或借位时，CY 被置位。在执行逻辑运算时，可以被硬件或软件置位或清零。在布尔处理机中，它被认为是位累加器。

（2）AC（PSW.6）：辅助进位标志位。进行加法或减法运算时，当低 4 位向高 4 位进位或借位时，AC 被置位，否则就被清零。

（3）F0（PSW.5）：用户标志位。用户定义的一个状态标志，可以用软件来使它置位或清零，也可以用软件测试 F0 以控制程序的流向。

（4）RS1（PSW.4）、RS0（PSW.3）：寄存器组选择位。可以用软件来置位或清零，以确定工作寄存器组，RSl、RS0 与工作寄存器组的对应关系见表 2-2。

（5）OV（PSW.2）：溢出标志位。当执行算术运算时，对带符号数作加、减运算时，OV＝1，表示加、减运算的结果超出 8 位带符号数的范围（+127～-128）。OV 标志常用 C6 和 C7 的关系来表示：

$$OV=C6 \oplus C7$$

当进行加、减运算时，C6 表示 D6 位向 D7 位有进位或借位，C7 表示 D7 位向进位位有进位或借位。

（6）—（PSW.1）：保留位，无定义。

（7）P（PSW.0）：奇偶校验位。若累加器（ACC）中“1”的位个数是奇数个则 P＝1，偶数个则 P＝0。

此标志位对串行通信中的数据传输有重要意义。在串行通信中常用奇偶校验来检验数据传输的误码率。在发送端可根据 P 的值对数据的奇偶位置位或清零，若通信协议中规定采用奇校验的办法，则 P＝1，当数据传输到接收端，若 P＝1，则表示传输过程中奇偶无错误，可以接收，否则奇偶有错，不能接收。

PSW 是编程时需要特别关注的一个寄存器。如当前 ACC 累加器中数据的奇偶性（P）、做加减法时的进位与借位（CY）、4 个工作寄存器组的选择（RS1、RS0）以及辅助进位（AC）和溢出标志位（OV）等。

4. DPTR 数据指针（DPL 和 DPH）

DPTR 是一个 16 位特殊功能寄存器，用来存放外部数据存储器的 16 位地址。DPTR 可以分成两个 8 位寄存器：高位字节寄存器 DPH 和低位字节寄存器 DPL，既可作一个 16 位寄存器用，也可作两个 8 位寄存器用。

5. SP 堆栈指针

SP 是一个 8 位特殊功能寄存器，MCS-51 单片机堆栈是在内部 RAM 中，复位初始化后，SP=07H，从 08H 开始存放。因为 08H～1FH 是 1～3 工作寄存器组，如要用到这些区，可重新设置 SP。

6. P0、P1、P2 和 P3 口

前面已经介绍了 MCS-51 单片机有 4 个双向 I/O 口 P0、P1、P2、P3。如果需要从指定端口输出一个数据，只需将数据写入指定 I/O 口即可。如果需要从指定 I/O 口输入一个数据，只需先将数据 0FFH（全部为 1）写入指定 I/O 口，然后再读指定 I/O 口即可。如果不先写入 0FFH（全部为 1），读入的数据有可能不正确。

2.2.3 “头文件包含”处理

所谓“头文件”是指一个文件将另外一个文件的内容全部包含进来。

头文件一般在 C:\KELL\C51\INC 下，INC 文件夹中有不少头文件，并且其中还有很多以公司分类的文件夹，里面也都是相关产品的头文件。如果我们要使用自己写的头文件，只需把自己写的头文件放在 INC 文件夹里就可以了。

在单片机中用 C 语言编程时，往往第一行就是头文件或者其他的自定义头文件。以 AT89X52.H 头文件为例，根据前面介绍过的特殊功能寄存器知识，下面对 AT89X52.H 头文件进行初步解析。

1. 特殊功能寄存器在 AT89X52.H 中的定义

打开 AT89X52.H 头文件可以看到有关特殊功能寄存器的一些内容：

```
/*-----------------------------------------------
Byte Registers
-----------------------------------------------*/
sfr P0     = 0x80;
sfr SP     = 0x81;
sfr DPL    = 0x82;
sfr DPH    = 0x83;
sfr PCON   = 0x87;
sfr TCON   = 0x88;
sfr TMOD   = 0x89;
sfr TL0    = 0x8A;
sfr TL1    = 0x8B;
sfr TH0    = 0x8C;
sfr TH1    = 0x8D;
sfr P1     = 0x90;
sfr SCON   = 0x98;
sfr SBUF   = 0x99;
sfr P2     = 0xA0;
sfr IE     = 0xA8;
sfr P3     = 0xB0;
sfr IP     = 0xB8;
```

```
sfr T2CON   = 0xC8;
sfr T2MOD   = 0xC9;
sfr RCAP2L  = 0xCA;
sfr RCAP2H  = 0xCB;
sfr TL2     = 0xCC;
sfr TH2     = 0xCD;
sfr PSW     = 0xD0;
sfr ACC     = 0xE0;
sfr B       = 0xF0;
```

根据图 2-10 不难看出，这里都是一些有关特殊功能寄存器符号的定义，即规定符号名与地址的对应关系。例如：

```
sfr P1 = 0x90;
```

这条语句是定义 P1 与地址 0x90 对应，其目的是为了要使用 P1 这个符号，即通知 C 编译器，程序中所用的 P1 是指单片机的 P1 端口，而不是其他变量。P1 口的地址就是 0x90，0x90 是 C 语言中十六进制数的写法，相当于汇编语言中写 90H。

2. 符号 P1_0 表示 P1.0 引脚

打开 AT89X52.H 头文件可以看到有关 P1 口位符号定义的一些内容：

```
/*-----------------------------------------------
P1 Bit Registers
-----------------------------------------------*/
sbit P1_0 = 0x90;
sbit P1_1 = 0x91;
sbit P1_2 = 0x92;
sbit P1_3 = 0x93;
sbit P1_4 = 0x94;
sbit P1_5 = 0x95;
sbit P1_6 = 0x96;
sbit P1_7 = 0x97;
```

根据图 2-11 不难看出，这里都是一些有关 P1 口每位符号的定义，即规定符号名与位地址的对应关系。例如：

```
sbit P1_0 = 0x90;
```

这条语句是定义 P1_0 与位地址 0x90 对应，其目的是为了要使用 P1_0 这个符号，即通知 C 编译器，程序中所用的 P1_0 是指单片机 P1 口的第 0 位，而不是其他位变量。在 C 语言里，如果直接写 P1.0，C 编译器并不能识别，而且 P1.0 也不是一个合法的 C 语言变量名，所以要给它另起一个名字，这里起的名为 P1_0。AT89X52.H 头文件的其他部分可以参考以上说明，以后再详细介绍。

2.2.4　程序存储器

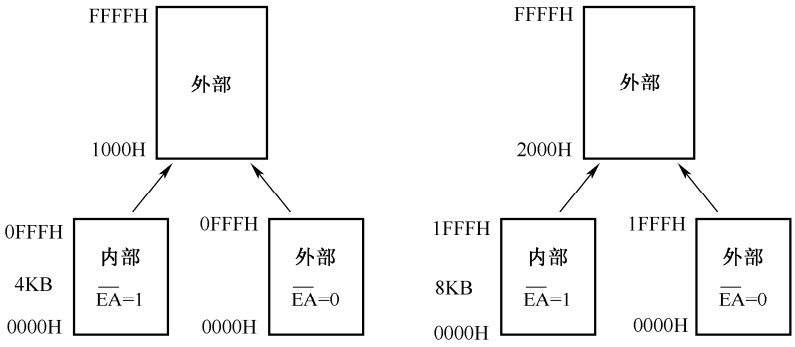
程序存储器用于存放用户程序、数据和表格等。它是以程序计数器 PC 作为地址指针，MCS-51 的程序计数器 PC 是 16 位的，所以 MCS-51 具有 64KB 程序存储器寻址空间。

1. 程序存储器的配置

对于内部无 ROM 的 8031 单片机，它的程序存储器必须外接，空间地址为 64KB，此时单片机的 EA 端必须接地，强制 CPU 从外部程序存储器读取程序。

对于内部有 ROM 的单片机，正常运行时，则需接高电平，使 CPU 先从内部的程序存储器中读取程序，当 PC 值超过内部 ROM 的容量时，才会转向外部的程序存储器读取程序。51

系列程序存储器内部有 4KB 的程序存储单元，其地址为 0000H～0FFFH，如图 2-14（a）所示；52 系列程序存储器内部有 8KB 的程序存储单元，其地址为 0000H～1FFFH，如图 2-14（b）所示。



（a）51 系列 ROM 配置图　　（b）52 系列 ROM 配置图

图 2-14　MCS-51ROM 配置图

当 $\overline{EA}$ =1 时，程序从内部 ROM 开始执行，当 PC 值超过内部 ROM 容量时会自动转向外部 ROM 空间。当 $\overline{EA}$ =0 时，程序从外部存储器开始执行，例如前面提到的内部无 ROM 的 8031 单片机，在实际应用中就要把 8031 的引脚接为低电平。

2. 具有特殊功能的地址

在程序存储器中具有一些特殊功能的地址，这在使用中应加以注重。

（1）启动地址。单片机启动复位后，程序计数器的内容为 0000H，所以系统必须从 0000H 单元开始执行程序。因而 0000H 是启动地址，也称为系统程序的复位入口地址。一般在 0000H～0002H 这 3 个单元中存放一条无条件转移指令，从转移地址开始存放初始化程序及主程序，让 CPU 直接去执行用户指定的程序。

（2）中断服务程序入口地址。其他特殊功能地址分别为各种中断源的中断服务程序入口地址，如表 2-3 所示。

表 2-3　各种中断服务程序入口地址

中断源	入口地址
外部中断 0	0003H
定时/计数器 0	000BH
外部中断 1	0013H
定时/计数器 1	001BH
串行中断	0023H
*定时器 2 溢出或 T2EX（P1.1）端负跳	002BH

*表中第 6 个中断源为 52 系列芯片特有。

以上是专门用于存放中断服务程序的地址单元，中断响应后，按中断的类型，自动转到各自的入口地址去执行程序。

2.3　工作模块 4　开关控制 LED 循环点亮

用 P3.0 作输入接开关 SW，P1 口作输出接 8 个 LED，通过开关 SW 控制 LED 循环点亮。开关 SW 合上，LED 循环点亮，开关 SW 打开，LED 停止循环点亮。

2.3.1　开关控制 LED 循环点亮电路设计

开关控制 LED 循环点亮电路比 LED 循环点亮控制电路（见图 2-1）多一个开关电路部分，其他都一样。开关 SW 一端接到单片机的 P3.0 引脚上，另一端接地，当开关 SW 合上时，P3.0 引脚就接到低电平。开关控制 LED 循环点亮电路设计如图 2-15 所示。

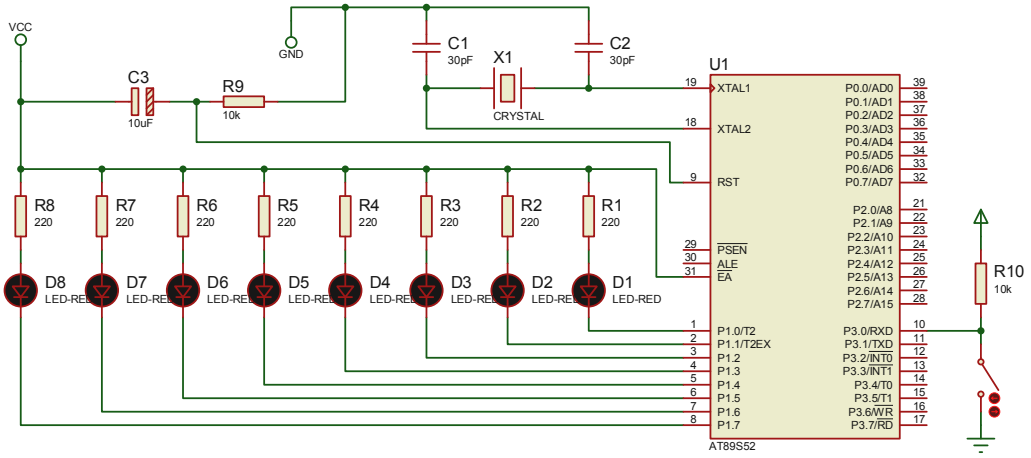


图 2-15　开关控制 LED 循环点亮电路

运行 Proteus 软件，新建“开关控制 LED 循环点亮”设计文件。按图 2-15 所示，放置并编辑 AT89S52、CRYSTAL、CAP、CAP-ELEC、RES、LED-RED 和 SWITCH 等元器件。设计完成开关控制 LED 循环点亮电路后，进行电气规则检测。

2.3.2　开关控制 LED 循环点亮程序设计

与 LED 循环点亮控制程序相比，关键是如何用开关控制 LED 循环点亮。根据工作任务要求，用 P3.0 作输入接开关 SW，通过开关 SW 控制 LED 循环点亮。开关 SW 合上，P3.0 为低电平，LED 循环点亮；开关 SW 打开，P3.0 为高电平，LED 停止循环点亮。

开关控制 LED 循环点亮程序如下：

```
#include <AT89X52.H>                    //包含 AT89X52.H 头文件
sbit SW=P3^0;                           //符号 SW 表示 P3.0 引脚
void  Delay()                           //延时函数
{
  unsigned char i, j;
     for (i=0;i<255;i++)
```

```
        for (j=0;j<255;j++);
}
void main()
{
   unsigned char i;
   unsigned char temp;
   P1 = 0xff;                    //十六进制全 1,熄灭所有 LED
   while(1)
   {
     temp = 0x01;                //第一位为 1，即初始控制码为 0x01
     for (i=0;i<8;i++)
     {
       if(SW==0)                 // SW 若合上，P3.0 为低电平，LED 循环点亮
       {
         P1 = ~ temp;            //temp 值取反送 P1 口
         Delay();
         temp = temp << 1 ;      //temp 值左移一位，获得下一个控制码
       }
     }
   }
 }
```

开关控制 LED 循环点亮程序设计好以后，运行 Keil μVision2 软件，生成“开关控制 LED 循环点亮.hex”文件。然后打开“开关控制 LED 循环点亮”Proteus 电路，加载“开关控制 LED 循环点亮.hex”文件。最后进行仿真运行，观察开关控制 LED 循环点亮是否与设计要求相符。

2.3.3 C51 数据类型

C51 定义了标准 C 语言的所有数据类型，同时对标准 C 语言进行了扩展，更加注意对系统资源的合理利用，如表 2-4 所示。

表 2-4 C51 基本数据类型

数据类型	长度	数值范围
unsigned char	1 字节	0～255
char	1 字节	-128～+127
unsigned int	2 字节	0～65535
int	2 字节	-32768～+32767
unsigned long	4 字节	0～4294967295
long	4 字节	-2147483648～+2147483647
float	4 字节	±1.175494E-38～±3.402823E+38
*	1～3 字节	对象的地址
bit	位	0 或 1
sfr	1 字节	0～255
sfr16	2 字节	0～65535
sbit	位	0 或 1

1. C51 基本数据类型

标准 C 语言中的基本数据类型为 char、int、short、long、float 和 double，而在 C51 编译器中 int 和 short 相同，float 和 double 相同，等等。

（1）char 字符类型。char 类型的长度是一个字节（8 位），通常用于定义处理字符数据的变量或常量。这很适合 MCS-51 单片机，因为 MCS-51 单片机每次可处理 8 位数据。char 类型分无符号字符类型 unsigned char 和有符号字符类型 signed char，默认值为 signed char 类型。

unsigned char 类型，用字节中所有的位来表示数值，数值范围是 0～255。常用于处理 ASCII 字符或用于处理小于或等于 255 的整型数。signed char 类型，最具有重要意义的位是最高位上的符号标志位，“0”表示正数，“1”表示负数，负数用补码表示，数值范围是-128～+127。正数的补码与原码相同，负数的补码等于它的原码按位取反后加 1（符号位不变）。

例如：

```
unsigned char a;        //定义变量 a 为无符号字符类型 unsigned char
char b;                 //定义变量 b 为有符号字符类型 signed char
```

（2）int 整型。int 整型长度为两个字节（16 位），用于存放一个 2 字节数据。MCS-51 系列单片机将 int 型变量的高位字节数存放在低地址字节中，低位字节数存放在高地址字节中。int 整型分有符号整型数 signed int 和无符号整型数 unsigned int，默认值为 signed int 类型。

unsigned int 表示的数值范围是 0～65535。signed int 表示的数值范围是-32768～+32767，字节中最高位表示数据的符号，“0”表示正数，“1”表示负数。

例如：

```
unsigned int x;         //定义变量 x 为无符号整型数 unsigned int
int y;                  //定义变量 y 为有符号整型数 signed int
```

（3）long 长整型。long 长整型长度为 4 个字节（32 位），用于存放一个 4 字节数据。分有符号 long 长整型 signed long 和无符号长整型 unsigned long，默认值为 signed long 类型。

unsigned long 表示的数值范围是 0～4294967295。signed int 表示的数值范围是-2147483648～+2147483647，字节中最高位表示数据的符号，“0”表示正数，“1”表示负数。

（4）float 浮点型。float 浮点型长度为 4 个字节（32 位），占 4 个字节。在十进制中具有 7 位有效数字，许多复杂的数学表达式都采用浮点变量数据类型。

（5）* 指针型。指针型本身就是一个变量，在这个变量中存放的是指向另一个数据的地址。这个指针变量要占据一定的内存单元，在 C51 中它的长度一般为 1～3 个字节。

2. C51 扩展的数据类型

为了更加有效地利用单片机的硬件资源，对标准 C 语言进行了扩展，增加了如下几个特殊的数据类型。

（1）bit 位变量。bit 位变量可以将与 MCS-51 硬件特性操作有关的定义成位变量。位变量必须定位在 MCS-51 单片机内部 RAM 的位寻址空间中。但不能定义位指针，也不能定义位数组。bit 位变量的值就是一个二进制位，不是 0 就是 1，类似 True 和 False。

例如：

```
bit flag;                //flag 为 bit 位变量，其值是 0 或 1
```

（2）sfr 特殊功能寄存器。为了能直接访问这些特殊功能寄存器 SFR，C51 提供了一种自主形式的定义方法，这种定义方法与标准 C 语言不兼容，只适用于对 MCS-51 系列单片机进行 C 语言编程。sfr 占用一个字节，数值范围为 0～255。利用它可以访问 51 单片机内部的所有特殊功能寄存器。特殊功能寄存器 C51 定义的一般语法格式如下：

sfr 特殊功能寄存器名=特殊功能寄存器的字节地址；

例如：

```
sfr P1 = 0x90;
```

这一句定义了 P1 为 P1 端口在内部的寄存器，在后面的语句中我们可以用 P1 =0xff（对 P1 端口的所有引脚置高电平）之类的语句来操作特殊功能寄存器。

又如：

```
sfr  SCON=0x98;          //串口控制寄存器，地址为 0x98
sfr  TMOD=0x89;          //定时器/计数器方式控制寄存器，地址为 0x89
```

“sfr”是定义语句的关键字，其后必须跟一个 MSC-51 单片机真实存在的特殊功能寄存器名，“=”后面必须是一个整型常数，不允许带有运算符的表达式，是特殊功能寄存器的字节地址，这个常数值的范围必须在 SFR 地址范围内，位于 0x80～0xFF。

（3）sfr16 16 位特殊功能寄存器。sfr16 占用两个字节，数值范围为 0～65535。sfr16 和 sfr 一样用于操作特殊功能寄存器，所不同的是它用于操作占用两个字节的寄存器，如 52 子系列的定时器/计数器 2。

在许多新的 MCS-51 系列单片机中，有时会使用两个连续地址的特殊功能寄存器来指定一个 16 位的值。为了有效地访问这类 SFR，可使用关键字“sfr16”来定义，16 位 SFR 定义语句的语法格式与 8 位 SFR 相同，只是“=”后面的地址必须用 16 位 SFR 的低字节地址，即低字节地址作为 sfr16 的定义地址。

例如：

```
sfr16 T2 = 0xCC          //定时器/计数器 2，T2 低 8 位地址为 0xCC，T2 高 8 位地址为 0xCD
```

这种定义适用于所有新的 16 位 SFR，但不能用于定时器/计数器 0 和 1。

（4）sbit 可寻址位。C51 的扩充功能支持特殊位的定义，与 SFR 定义一样，关键字 sbit 用于定义某些特殊位，利用它可以访问芯片内部的 RAM 中的可寻址位或特殊功能寄存器中的可寻址位。如先前定义：

```
sfr P1 = 0x90;
```

因为 P1 端口的寄存器是可位寻址的，所以可以定义：

```
sbit P1_1 = P1^1;        //P1_1 为 P1 中的 P1.1 引脚
```

这样以后的程序语句中就可以用 P1_1 来对 P1.1 引脚进行读写操作了。在 C 语言里，由于 P1.1 不是一个合法的 C 语言变量名，要给它另起一个名字，这里起的名为 P1_1，所以必须给它们建立联系，这里使用了 C51 的关键字 sbit 来定义，sbit 的用法有三种格式：

第一种格式：

```
sbit bit-name = sfr-name^int constant;
```

其中 bit-name 是一个寻址位符号名，该位符号名必须是 MCS-51 单片机中规定的位名称，sfr-name 必须是已定义过的 SFR 的名字，“^”后的整常数是寻址位在特殊功能寄存器 sfr-name 中的位号，必须是 0～7 范围中的数。

例如：

```
sfr   PSW=0xD0;          //定义 PSW 寄存器地址为 0xD0
```

```
sbit  OV=PSW^2;          //定义 OV 位为 PSW.2，地址为 0xD2
sbit  CY=PSW^7;          //定义 CY 位为 PSW.7，地址为 0xD7
```

第二种格式：

```
sbit  bit-name = int constant^int constant;
```

其中“=”后的 int constant 为寻址地址位所在的特殊功能寄存器的字节地址，“^”符号后的 int constant 为寻址位在特殊功能寄存器中的位号。

例如：

```
sbit  OV=0xD0^2;         //定义 OV 位地址是 0xD0 字节中的第 2 位
sbit  CY=0xD0^7;         //定义 CY 位地址是 0xD0 字节中的第 7 位
```

第三种格式：

```
sbit  bit-name = int constant;
```

其中“=”后的 int constant 为寻址位的绝对位地址。

例如：

```
sbit  OV=0xD2;           //定义 OV 位地址为 0xD2
sbit  CY=0xD7;           //定义 CY 位地址为 0xD7
```

特殊功能位代表了一个独立的定义类，不能与其他位定义和位域互换。

MCS-51 系列单片机的特殊功能寄存器的数量与类型不尽相同，因此建议将所有特殊的 sfr 定义放入一个头文件中，该文件应包括 MCS-51 单片机系列机型中的 SFR 定义。

说明：在 C51 存储器类型中提供一个 bdata 的存储器类型，这是指可位寻址的数据存储器，位于单片机的可位寻址区中，可以将要求可位寻址的数据定义为 bdata，如：

```
unsigned char bdata xb;      //在可位寻址区定义 unsigned char 类型的变量 xb
int bdata yb[2];             //在可位寻址区定义数组 yb[2]，这些也称为可寻址位对象
sbit xb7=xb^7                //用关键字 sbit 定义位变量来独立访问可寻址位对象的其中一位
sbit yb12=yb[1]^12;
```

操作符“ ^ ”后面的位置的最大值取决于指定的数据类型，char 0～7，int 0～15，long 0～31。

2.3.4　C 语言常量与变量

常量在程序运行过程中是不能改变的，变量在程序运行过程中是可以不断变化的。变量的定义可以使用所有 C51 编译器支持的数据类型，而常量的数据类型只有整型、浮点型、字符型、字符串型和位变量。

1. 常量

常量可用在不必改变值的场合，如固定的数据表、字库等。

（1）整型常量可以表示为十进制，如 123，0，-89 等；十六进制以 0x 开头，如 0x34，-0x3B 等。长整型在数字后面加字母 L，如 104L，034L，0xF340 等。

（2）浮点型常量可分为十进制和指数表示形式。十进制由数字和小数点组成，如 0.888，3345.345，0.0 等，整数或小数部分为 0，可以省略，但必须有小数点。指数表示形式为[±]数字[.数字]e[±]数字，[]中的内容为可选项，其中内容根据具体情况可有可无，但其余部分必须有，如 125e3，7e9，-3.0e-3。

（3）字符型常量是单引号内的字符，如'a'，'d'等，不可以显示的控制字符，可以在该字符前面加一个反斜杠"\"组成专用转义字符。常用转义字符表见表 2-5。

表 2-5 常用转义字符表

转义字符	含义	ASCII 码（十六/十进制）
\o	空字符（NULL）	00H/0
\n	换行符（LF）	0AH/10
\r	回车符（CR）	0DH/13
\t	水平制表符（HT）	09H/9
\b	退格符（BS）	08H/8
\f	换页符（FF）	0CH/12
\'	单引号	27H/39
\"	双引号	22H/34
\\	反斜杠	5CH/92

（4）字符串型常量由双引号内的字符组成，如"test"，"OK"等。当引号内没有字符时，为空字符串。在使用特殊字符时同样要使用转义字符，如双引号。在 C 语言中字符串常量是作为字符类型数组来处理的，在存储字符串时系统会在字符串尾部加上\o 转义字符，以作为该字符串的结束符。字符串常量"A"和字符常量'A'是不同的，前者在存储时多占用一个字节的字间。

（5）位标量，它的值是一个二进制值。

常量的定义方式有几种，下面加以说明。

```
#difine False 0x0;              //用预定义语句可以定义常量
#difine True 0x1;               //这里定义 False 为 0,True 为 1
```

程序中用到 False 和 True 时，在编译时，False 替换为 0，True 替换为 1。

```
unsigned int code a=100;        //这一句用 code 把 a 定义在程序存储器中并赋值
const unsigned int c=100;       //用 const 定义 c 为无符号 int 常量并赋值
```

以上两句的值都保存在程序存储器中，而程序存储器在运行中是不允许修改的，所以如果在这两句后面用了类似 a=110，a++这样的赋值语句，编译时将会出错。

2. 变量

变量在程序执行过程中其值能不断变化。要在程序中使用变量必须先用标识符作为变量名，并指出所用的数据类型和存储模式，这样编译系统才能为变量分配相应的存储空间。定义一个变量的格式如下：

```
[存储种类]  数据类型  [存储器类型]  变量名表
```

在定义格式中除了数据类型和变量名表是必要的，其他都是可选项。

（1）存储种类。存储种类有 4 种：自动（auto）、外部（extern）、静态（static）和寄存器（register），默认类型为自动（auto）。

（2）数据类型。这里的数据类型和前面各种数据类型的定义是一样的。说明一个变量的数据类型后，还可选择说明该变量的存储器类型。

（3）存储器类型。存储器类型的说明就是指定该变量在 C51 硬件系统中所使用的存储区域，并在编译时准确定位。表 2-6 中是 Keil μVision2 能认别的存储器类型。

如果省略存储器类型，系统则会按编译模式 SMALL、COMPACT 或 LARGE 规定的默认存储器类型去指定变量的存储区域。无论什么存储模式都可以声明变量在任何的 8051 存储区范围，然而把最常用的命令如循环计数器和队列索引放在内部数据区可以显著地提高系统性能。还有要指出的就是变量的存储种类与存储器类型是完全无关的。

表 2-6　存储器类型

存储器类型	说明
data	直接访问内部数据存储器（128B），访问速度最快
bdata	可位寻址内部数据存储器（16B），允许位与字节混合访问
idata	间接访问内部数据存储器（256B），允许访问全部内部地址
pdata	分页访问外部数据存储器（256B），用 MOVX @Ri 指令访问
xdata	外部数据存储器（64KB），用 MOVX @DPTR 指令访问
code	程序存储器（64KB），用 MOVC @A+DPTR 指令访问

（4）存储模式。SMALL 存储模式把所有函数变量和局部数据段放在 8051 系统的内部数据存储区，这使访问数据非常快，但 SMALL 存储模式的地址空间受限。在写小型的应用程序时，变量和数据放在 data 内部数据存储器中是很好的，因为访问速度快，但在较大的应用程序中 data 区最好只存放小的变量、数据或常用的变量（如循环计数、数据索引），而大的数据放置在其他存储区域。

COMPACT 存储模式中所有的函数和程序变量和局部数据段定位在 8051 系统的外部数据存储区。外部数据存储区最多可以有 256 字节（一页），在本模式中外部数据存储区的短地址用@R0/R1。

LARGE 存储模式中所有函数和过程的变量和局部数据段都定位在 8051 系统的外部数据区。外部数据区最多可以有 64KB，这要求用 DPTR 数据指针访问数据。

2.4　工作模块 5　步进电机控制

使用 AT89S52 单片机，由 P1 口的 P1.0、P1.1、P1.2 和 P1.3 四个引脚通过步进电机驱动电路分别接在四相步进电机的四相绕组，步进电机的励磁方式采用四相双四拍，通过程序控制步进电机正转。

2.4.1　认识步进电机

步进电机是利用输入数字信号转换成机械能量的电气设备。步进电机的应用范围已很广，除数控、工业控制和计算机外部设备中大量使用外，在工业自动线、印刷机、遥控指示装置、航空系统中都已成功地应用了步进电机。

1. 步进电机的结构

以内部线圈绕线来区分步进电机，有四相和五相两种，使用 5V 及 12V 电源控制。一般来说，四相步进电机又称为二相双绕组步进电机，是最常用的一种电机，其内部接线图如图 2-16 所示。

线圈被分为 A、B、C 和 D 四相。由于 A 相和 C 相（或 B 相和 D 相）线圈都绕在相同的磁极上，而此两组线圈缠绕的方向相反，只需对其中的一组线圈励磁，便可改变定子磁场的极性。因此不可将 A 相和 C 相（或 B 相和 D 相）线圈同时励磁。

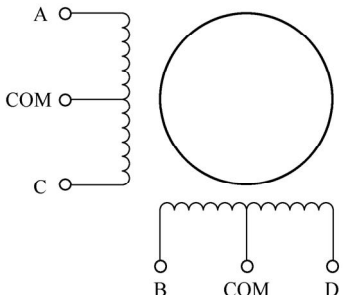


图 2-16　四相步进电机内部接线图

步进电机是“一步一步”转动的一种电机，每输入一个脉冲信号（Pulse），步进电机固定旋转一个步进角。步进角由步进电机规范而定，一般为 1.8°～9°，市面上以 1.8°步进角较普遍。例如，步进角为 1.8°的步进电机，如果输入 200 个脉冲信号，步进电机就会旋转 200 个步进角，且刚好转一圈（200×1.8°=360°）。由于步进电机的旋转角度与输入脉冲数目成正比，只要控制输入的脉冲数目便可控制步进电机的旋转角度。因此，常用于精确定位和精确定速。

2. 步进电机线圈的励磁方式

直流电流通过定子线圈建立磁场方式，称为励磁。如果要控制步进电机进行正确的定位和控制，必须按照一定的顺序对各相线圈进行励磁。四相式步进电机线圈励磁的方式，可分为一相励磁、二相励磁和一-二相励磁三种。本工作模块的步进电机励磁方式采用的是二相励磁，即四相双四拍。

（1）四相：表示电动机有四相绕组，分别为 A、B、C、D 绕组。

（2）二相励磁（双）：表示每一种励磁状态都有两相绕组励磁。

（3）拍：从一种励磁状态转换到另一种励磁状态，叫一拍。

（4）二相励磁顺序（四拍）：四种励磁状态为一个循环。只要改变励磁顺序，就可以改变步进电机旋转方向。

正转时二相励磁顺序：

（A，B）→（B，C）→（C，D）→（D，A）→ （A，B）……

反转时二相励磁顺序：

（A，B）→（D，A）→（C，D）→（B，C）→ （A，B）……

2.4.2　步进电机控制电路设计

按照工作任务要求，步进电机控制电路由单片机最小应用系统、步进电机驱动电路及步进电机构成。步进电机控制电路设计如图 2-17 所示。

步进电机驱动电路由 ULN2003A 和 74LS04 构成，其中 ULN2003A 驱动器是一个高电压、大电流的达灵顿对数组，其中包含 7 个具备共射极的开集极达灵顿对。由于 ULN2003A 的输入与 TTL 电平兼容，所以一般能直接连接到驱动组件或负载上，例如，继电器、电机或 LED 显示器等。

运行 Proteus 软件，新建“步进电机控制”设计文件。按图 2-17 所示，放置并编辑 AT89S52、CRYSTAL、CAP、CAP-ELEC、RES、MOTOR-STEPPER（步进电机）、ULN2003A（驱动器）和 74LS04（反相器）等元器件。设计完成步进电机控制电路后，进行电气规则检测。

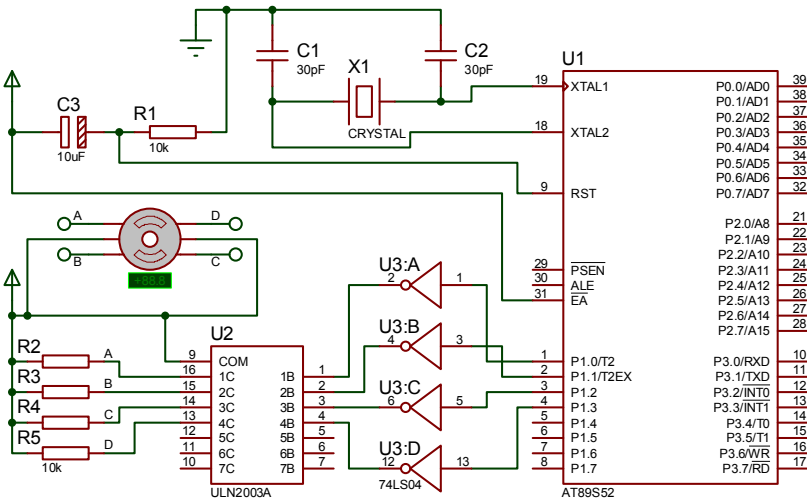


图 2-17　步进电机控制电路

2.4.3　步进电机控制程序设计

步进电机控制电路设计完成以后，还要根据工作任务要求，按照励磁方式为 2 相励磁方式进行程序设计，实现步进电机正转。

1. 电机正转功能实现分析

在步进电机控制系统中，P1 口的 P1.0、P1.1、P1.2 和 P1.3 四个引脚通过步进电机驱动电路分别接在四相步进电机的四相绕组，步进电机正转时，2 相励磁顺序：（A，B）→（B，C）→（C，D）→（D，A）→（A，B），控制状态与 P1 口的控制码的对应关系如表 2-7 所示。

表 2-7　控制状态与 P1 口的控制码的对应关系

控制状态	P1 口 控制码	P1.3 D 相	P1.2 B 相	P1.1 C 相	P1.0 A 相
A 相、B 相绕组通电	03H	0	0	1	1
B 相、C 相绕组通电	06H	0	1	1	0
C 相、D 相绕组通电	0CH	1	1	0	0
D 相、A 相绕组通电	09H	1	0	0	1

由表 2-7 可以看出，步进电机 2 相励磁顺序与 P1 口的控制码之间的关系如下：

（1）A、B 绕组励磁，P1 口输出 0x03（二进制 00000011B），初始控制码为 0x03；

（2）B、C 绕组励磁，P1 口输出 0x06（二进制 00000110B），控制码为 0x06；

（3）C、D 绕组励磁，P1 口输出 0x0C（二进制 00001100B），控制码为 0x0C；

（4）D、A 绕组励磁，P1 口输出 0x09（二进制 00001001B），控制码为 0x09；

（5）重复第一步，进入下一个循环。

2. 步进电机控制程序设计

从以上分析可以看出，首先将初始控制码 0x03 从 P1 口输出，步进电机固定旋转一个步进角，然后按励磁顺序从 P1 口依次输出控制码 0x06、0x0C、0x09，不断循环，依次输出控制码，控制步进电机正转。在输出控制码之间一定要加延时，延时时间的长短决定步进电机的速度。

步进电机控制 C 语言程序如下：

```
#include <AT89X52.H>
//由 delay 参数确定延迟时间
void mDelay (unsigned int delay)
{
  unsigned int i;
  for(;delay >0; delay--)
    for(i=0;i<124;i++);
}
void main()
{
  while(1)
  {
        P1=0x03;          //A、B 绕组励磁
        mDelay (50);
        P1=0x06;          //B、C 绕组励磁
        mDelay (50);
        P1=0x0C;          //C、D 绕组励磁
        mDelay (50);
        P1=0x09;          //D、A 绕组励磁
        mDelay (50);
  }
}
```

步进电机控制程序设计好以后，打开“步进电机控制”Proteus 电路，加载 “步进电机控制.hex”文件。进行仿真运行，观察步进电机是否与设计要求相符。

2.5 技能拓展 ULN2003A 驱动器应用

在电子电路应用上，大多要求具有大电流输出的能力，以便驱动各种类型的负载，驱动电路是电子设备输出电路中一个重要的组成部分。

在大型仪器系统中，经常要用到伺服马达、步进马达、各种电磁阀等驱动电压高和功率较大的组件。因此，开发出来像这种 ULN2000 和 ULN2800 等高电压、大电流的达灵顿晶体管数组的产品，从而控制大功率组件。由于这类组件功能强大、应用范围广，因此许多公司都开发出相关产品，进而演变成各系列产品。其中，ULN2000 和 ULN2800 系列为美国 Texas Instruments 公司以及美国 Sprague 公司开发的产品。UN2000 系列能够同时驱动 7 组负载，ULN2800 系列则能够同时驱动 8 组负载。

2.5.1 ULN2003A 的特点

一般市面上较为常见的是 ULN2003A。ULN2003A 组件是一个高电压、大电流的达灵顿对数组，其中包含 7 个具备共射极的开集极达灵顿对。ULN2003A 具有以下特点：

（1）电流增益高（大于 1000mA）；

（2）带负载能力强（输出电流大于 500mA）；

（3）温度范围宽（-40～85℃）；

（4）工作电压高（大于 50V）。

2.5.2　ULN2003A 的引脚功能

ULN2003A 电路的引脚配置排列如图 2-18 所示，这是 16 引脚的 DIP 封装。

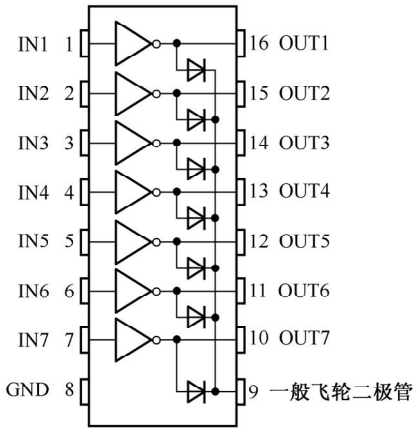


图 2-18　ULN2003AIC 引脚配置图

由于 ULN2003A 的输入与 TTL 电平兼容，所以一般能直接连接到驱动组件或负载上，例如，继电器、DC 马达或 LED 显示器等。对于每一个驱动器来说，都包含一个二极管，其阳极连接到输出端，阴极连接到 7 个二极管的共通点上。外部的负载连接到电源供应点和驱动器的输出端之间，该电源供应为小于+50V 的正电压。

【技能训练 2-2】单片机驱动继电器电路设计

如图 2-19 所示，通过 ULN2003A 的 3 个驱动器输出直接驱动 3 个继电器。每一个继电器线圈的端子连接到驱动器输出，另一端连接到供应电压上。连接多大的电压依赖继电器的规格。从图 2-18 和图 2-19 中可以看到二极管共通点也连接到供应电压上。输入到 ULN2003A IC 的是 TTL 电平的电压，因此，可以直接连接到单片机 I/O 口。这样就可轻易地控制外部高电压、大电流的负载。

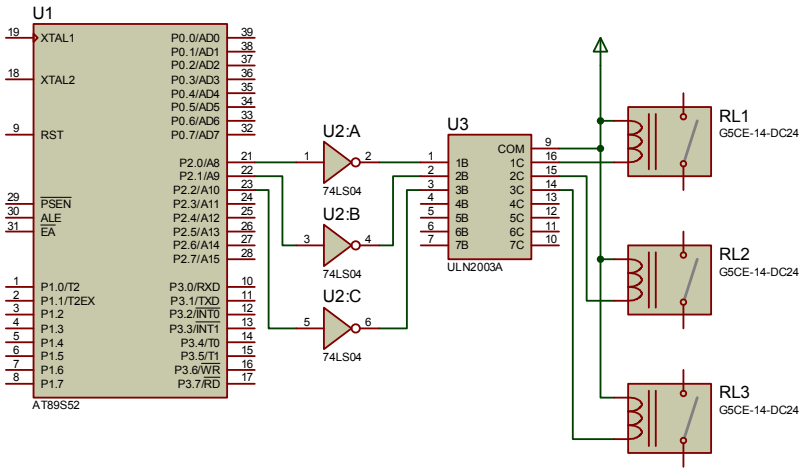


图 2-19　通过 I/O 端口和 ULN2003A 驱动继电器

有了上述介绍的各种输出外围电路的应用后，读者可以将其连接到单片机 I/O 口扩充的输出电路上，从而进一步控制或驱动所需的负载电路。

1. MCS-51 的 I/O 口

MCS-51 单片机有 P0 口、P1 口、P2 口和 P3 口，每个端口都各有 8 条 I/O 口线，每条 I/O 口线都能独立地用作输入或输出。P0 口的负载能力为 8 个 LSTTL 门电路，P1 口、P2 口和 P3 口的负载能力为 4 个 LSTTL 门电路。

（1）P0 口（P0.0～P0.7）是一个漏极开路双向 I/O 端口，须外接上拉电阻才能有高电平输出。访问外部存储器时，P0 口分时送出低 8 位地址 AB0～AB7 和 8 位数据 D0～D7。

（2）P1 口（P1.0～P1.7）是双向 I/O 端口，仅供用户作为输入输出用的端口。

（3）P2 口（P2.0～P2.7）是双向 I/O 端口，访问外部存储器时，P2 口送出高 8 位地址 AB8～AB15。

（4）P3 口（P3.0～P3.7）的第一功能和 P1 口一样是双向 I/O 端口。每一根线还具有第二种功能，包括串行通信、外部中断控制、计时计数控制及外部随机存储器内容的读取或写入控制等功能。

2. MCS-51 的存储器

MCS-51 单片机的程序存储器和数据存储器是各自独立的，有各自的寻址系统、控制信号和功能。在物理结构上分为内部数据存储器（RAM）256B、内部程序存储器（ROM）4KB、外部数据存储器（RAM）64KB 和外部程序存储器（ROM）64KB 四个存储空间。

内部 RAM 分为两个地址空间：00H～7FH 单元组成的低 128 字节（52 子系列为 256 字节）RAM 空间和 80H～FFH 单元组成的高 128 字节特殊功能寄存器 SFR 空间。低 128 字节分为通用寄存器区（00H～1FH）、位寻址区（20H～2FH）和用户 RAM 区（30H～7FH）。

3. 数据类型

C51 定义了标准 C 语言的所有数据类型，为了更加有效地利用单片机的硬件资源，对标准 C 语言进行了扩展，更加注意对系统资源的合理利用。

（1）bit 定义位变量，位变量必须定位在内部 RAM 的位寻址空间中。

（2）sfr 定义特殊功能寄存器，利用它可以访问单片机内部的所有特殊功能寄存器。

（3）sfr16 定义 16 位特殊功能寄存器，如 52 子系列的定时器/计数器 2。

（4）sbit 定义可寻址位（某些特殊位），利用它可以访问芯片内部的 RAM 中的可寻址位或特殊功能寄存器中的可寻址位。

4. 常量

常量在程序运行过程中不能改变，常量的数据类型只有整型、浮点型、字符型、字符串型和位变量。常量可用在不必改变值的场合，如固定的数据表、字库等。

5. 变量

变量在程序运行过程中可以不断变化。变量的定义可以使用所有 C51 编译器支持的数据类型。

2-1　P0 口、P1 口、P2 口和 P3 口的负载能力是多少？它们是否具有位寻址功能？

2-2　在输出时，P0 口为什么要外接上拉电阻才能有高电平输出？

2-3　MCS-51 单片机有哪几个存储空间，是如何分布的？

2-4　MCS-51 单片机的内部 RAM 分成几个不同区域及地址范围？

2-5　PSW 的作用是什么？常用的状态标志有哪几位？其作用是什么？能否位寻址?

2-6　bit 和 sbit 有什么区别？

2-7　在 C 语言里，sbit P1_0 = 0x90 语句的作用是什么？能不能直接使用 P1.0（说明原因）？

2-8　试一试能否将工作模块 3 的 LED 循环点亮改为 LED 双向循环点亮？

2-9　设计用开关控制步进电机转向的 AT89S52 单片机控制系统，功能要求：开关闭合，正转；开关断开，反转。

2-10　设计开关控制电灯点亮的 AT89S52 单片机控制系统，驱动电路采用 ULN2003A 和继电器。功能要求：开关闭合，电灯点亮；开关断开，电灯熄灭。提示：参考【技能训练 2-2】的单片机驱动继电器电路设计。