单片机原理及应用

项目三：单片机C语言

1. 熟悉单片机 C 语言编程基础；
2. 掌握单片机 C 语言中的顺序结构；
3. 掌握单片机 C 语言的应用；

1. 会阐述单片机 C 语言编程基础相关知识；
2. 会使用单片机 C 语言中的顺序结构；
3. 会使用单片机 C 语言中的顺序结构控制 LED 的编程方法；

1. 能分析设计任务，正确使用单片机 C 语中的顺序结构；
2. 能熟练编写单片机控制 LED ；
3. 能使用 Proteus 软件绘制电路原理图；
4. 能使用 Keil 软件编译程序对 LED 的控制，并与 Proteus 软件联调，实现控制电路仿真；
5. 能够完成流水灯电路设计与仿真。

4 课时

1.1 任务提出

利用单片机设计一种流水灯的效果，可以实现八只 LED 循环点亮，时间间隔为 5 s 。

1.2 方案设计

1. 单片机选择。采用 ATMEL 公司的 AT89C51 作为系统控制器的 CPU 方案。单片机算术运算功能强，软件编程灵活、自由度大，可以用软件编程实现各种算法和逻辑控制，并且由于其功耗低、体积小、技术成熟和成本低等优点，使其在各个领域应用广泛。

2. LED 电路的选择。每个发光二极管与单片机的一个引脚连接，一一对应。发光二极管的负极与单片机的引脚相连，正极通过一个限流电阻后与电源相连。

1.3 硬件设计

1. 电路组成

流水灯电路由单片机最小系统电路和八只发光二极管 LED 显示电路组成。

1.1 单片机最小系统电路

（1）主控制器

以 AT89C51 单片机作为主控处理器。

（2）晶振电路

晶振电路由 12MHZ 的石英晶振和两个 33uF 的电容并联。

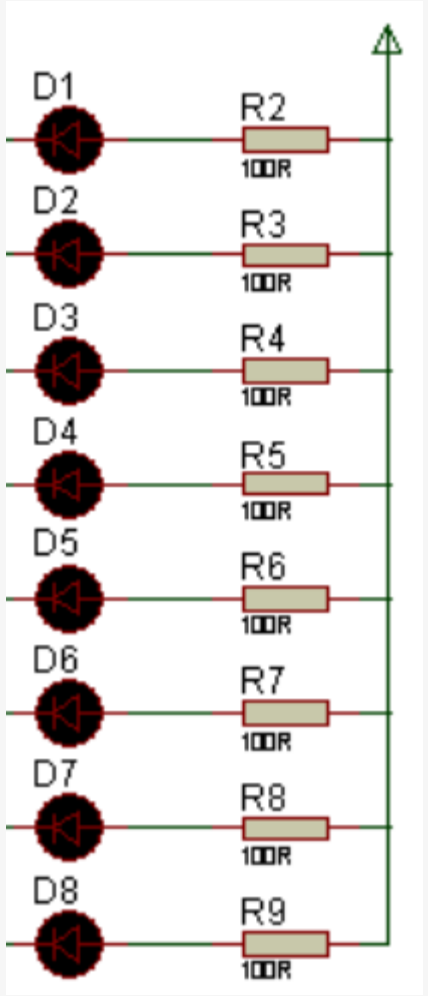
（3） 复位电路

复位电路由按键、 10uF 电容和 +5V 电源组成。

1.3 硬件设计

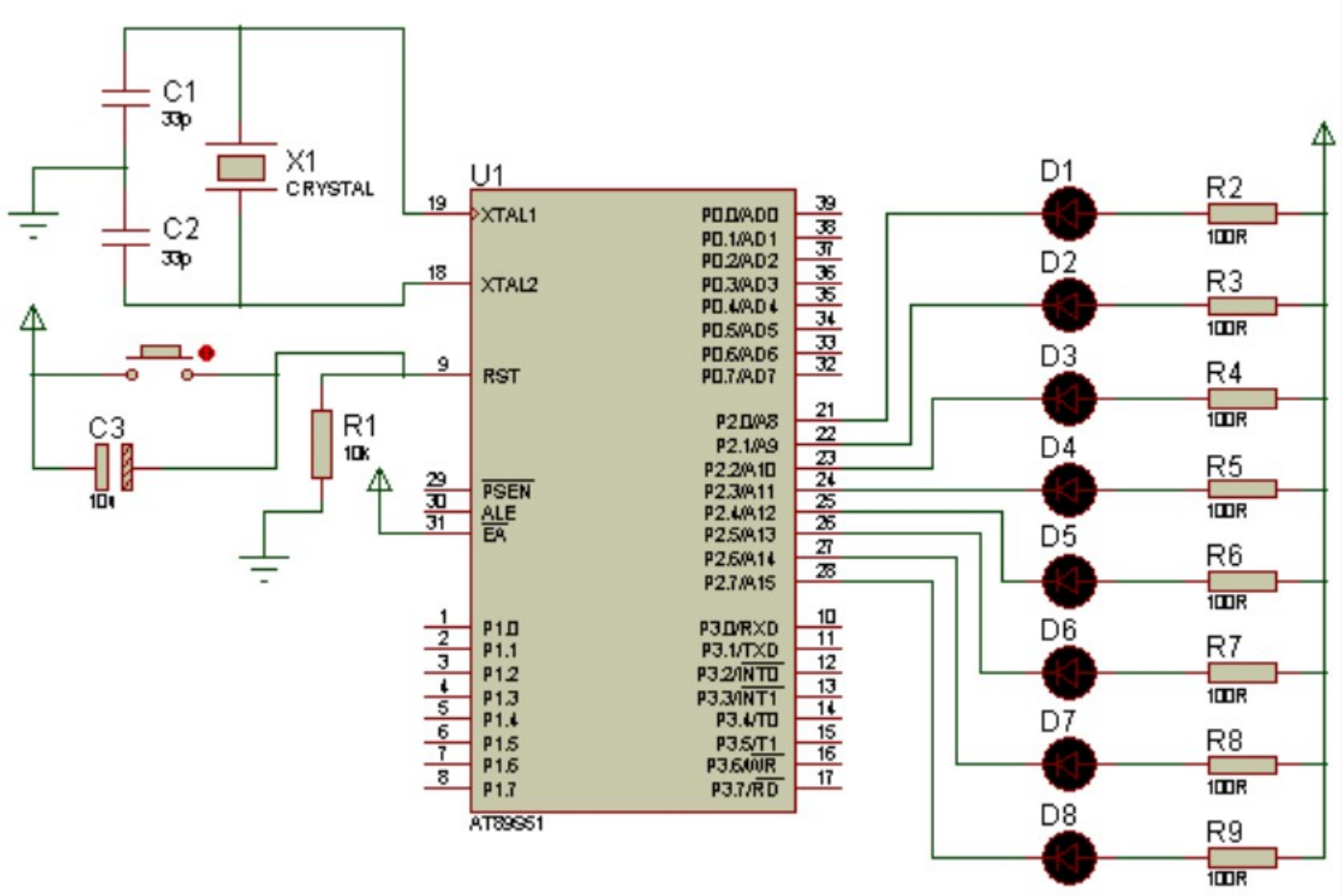
1.2 显示电路

显示电路有八只发光二极管和八个电阻组成。



1.3 硬件设计

2. 硬件电路



1.4 软件设计

1. 编程语言的选择

C51 交叉编译器提供了一种针对 MCS-51 系列微控制器用 C 语言编程的方法，可将 C 语言源程序编译生成 Intel 格
的可再定位目标代码。

C 语言是一种通用编程语言，符合 C 语言的 ANSI 标准，代码效率高，可结构化编程，在代码效率和速度上，完
可以和汇编语言相比拟，应用范围广。

利用 C 语言编程，具有极强的可移植性和可读性，同时，它只要求程序员对单片机的存储器结构有初步了解，而对处理器的指令集不要求了解。

1.4 软件设计

1. 编程语言的选择

C 语言的特点：

（ 1 ）结构化语言

C 语言由函数构成。函数包括标准函数和自定义函数，每个函数就是一个功能相对独立的模块。

C 语言还提供了多种结构化的控制语句，如顺序、条件、循环结构语句，满足程序设计结构化的要求。

1.4 软件设计

1. 编程语言的选择

C 语言的特点：

（ 2 ）丰富的数据类型

C 语言具有丰富的数据类型，便于实现各类复杂的数据结构，它还有与地址密切相关的指针及其运算符，直接访问内存地址，进行位 (bit) 一级的操作，能实现汇编语言的大部分功能，因此 C 语言被称为“高级语言中的低级语言”。

用 C 语言对 MCS-51 系列单片机开发应用程序，只要求开发者对单片机的存储器结构有初步了解，而不必十分熟悉处理器的指令集和运算过程，寄存器分配、存储器的寻址及数据类型等细节问题由编译器管理，不但减轻了开发者的负担，提高了效率，而且程序具有更好的可读性和可移植性。

1.4 软件设计

1. 编程语言的选择

C 语言的特点：

（ 3 ）便于维护管理

用 C 语言开发单片机应用系统程序，便于模块化程序设计。可采用开发小组计划和完成项目，分工合作，灵活管理。基本上杜绝了因开发人员变化所造成的对项目进度、后期维护及升级的影响，从而保证了整个系统的品质、可靠性及可升级性。

与汇编语言相比， C 语言的优点如下：

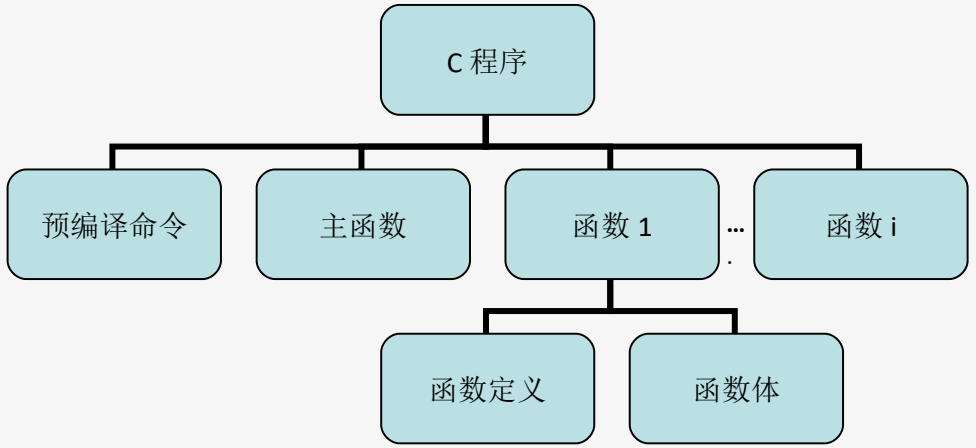
① 不要求编程者详细了解单片机的指令系统，但需了解单片机的存储器结构。
② 寄存器分配、不同存储器的寻址及数据类型等细节可由编译器管理。
③ 程序结构清晰，可读性强
④ 编译器提供了很多标准函数，具有较强的数据处理能力。

1.4 软件设计

2. C 语言的入门知识

（1） C 语言的基本结构

C 语言程序由函数组成， C 程序中的一个函数就相当于一个模块，每个模块具备一定相对独立的功能。 C 程序的结构如图 3-2 所示。



1.4 软件设计

2. C 语言的入门知识

（1） C 语言的基本结构

一个完整的 C 语言程序，是由一个且只能有一个 main() 函数 (又称主函数) 和若干个其他函数结合而成或仅由一个 main() 函数构成。每个 C 程序的执行总是从主函数 main() 开始，到主函数 main() 结束 , 不管 main() 函数在程序的哪个位置。

一个函数由两部分组成：函数说明和函数体。
函数说明部分包括函数类型、函数名、形参表等。
例如： void delay(unsigned char i)

```
{
unsigned char j,k;
for(k=0;k<i;k++)
for(j=0;j<255;j++);
}
```

1.4 软件设计

2. C 语言的入门知识

（1） C 语言的基本结构

对于 delay 函数来说， delay 是函数名，函数名前面的 void 说明函数的类型 (空类型，表示没有返回值) ，函数名后面必须跟一对圆括号，里面是函数的形式参数表，这里的 void delay(unsigned char i) 函数形参表是 unsigned char i(无符号字符型) 。

delay 函数后面一对大括号内的部分称为函数体，函数体由定义数据类型的说明部分和实现函数功能的执行部分组成。

使用 C 语言，要注意以下几点：
① C 语言程序中可以有预处理命令，预处理命令通常放在源程序的最前面。如 #include<reg51.h> 。
② C 语言程序中的每一个语句都要以“；”结束。一条语句可以多行书写，也可以一行书写多条语句。
③ C 语言区分大小写，例如：变量 i 和变量 I 表示两个不同的变量。

1.4 软件设计

2. C 语言的入门知识

（2）标识符和关键字

标识符用来标识源程序中某个对象的名字的，这些对象可以是语句、数据类型、函数、变量、数组等等，由字符串、数字和下划线等组成，注意的是第一个字符必须是字母或下划线。标识符在命名时应当简单，含义清晰，这样有助于阅读理解程序，区分大小写。

关键字则是编程语言保留的特殊标识符，其具有固定名称和含义。在程序编写中不允许标识符与关键字相同。关键字主要有 Auto 、 break 、 case 、 char 、 const 、 continue 、 default 、 do 、 double 、 else 、 enum 、 extern 、 flost 、 for 、 goto 、 if 、 int 、 long 、 register 、 return 、 short 、 signed 、 sizeof 、 static 、 struct 、 swicth 、 typedef 、 union 、 unsigned 、 void 、 volatile 、 while 等。

1.4 软件设计

2. C 语言的入门知识

（3）数据类型

在标准 C 语言中基本的数据类型为 char 、 int 、 short 、 long 、 float 和 double ， 而在 C51 编译器中 int 和 short 相同， float 和 double 相同。具体定义如表 3-1 所示。

数据类型	长　度	值　域
unsigned char	单字节	0 ～ 255
signed char	单字节	-128 ～ +127
unsigned int	双字节	0 ～ 65535
signed int	双字节	-32768 ～ +32767
unsigned long	四字节	0 ～ 4294967295
signed long	四字节	-2147483648 ～ +2147483647
float	四字节	±1.175494E-38 ～ ±3.402823E+38
*	1 ～ 3 字节	对象的地址
bit	位	0 或 1
sfr	单字节	0 ～ 255
sfr16	双字节	0 ～ 65535
sbit	位	0 或 1

1.4 软件设计

2. C 语言的入门知识

（4）常量和变量

常量是在程序运行过程中不能改变值的量，常量的数据类型只有整型、浮点型、字符型、字符串型和位标量，具体如下：

① 整型常量可以用十进制表示，也可以用十六进制（0x 开头）表示，长整型就在数字后面加字母 L，如 208L。

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。

1.4 软件设计

2. C 语言的入门知识

（4）常量和变量

④ 符串型常量由双引号内的字符组成，如“ test”， “ OK” 等。当引号内的没有字符时，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 中字符串常量是做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。

⑤ 位标量的值是一个二进制。

常量可用在不必改变值的场合，如固定的数据表，字库等。常量的定义方式有几种，下面来加以说明。

```
#define False 0x0 ;    // 用预定义语句可以定义常量
#define True 0x1 ;  // 这里定义 False 为 0,True 为 1 ，即在程序中用到 False 编译时自动用 0 替换， 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
```

以上两句它们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被修改的，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。

1.4 软件设计

2. C 语言的入门知识

（4）常量和变量

变量就是一种在程序执行过程中其值能不断变化的量。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，其它都是可选项。
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) 。

1.4 软件设计

2. C 语言的入门知识

（5）运算符和表达式

运算符就是完成某种特定运算的符号。运算符按其表达式中与运算符的关系可分为单目运算符，双目运算符和三目运算符。单目就是指需要有一个运算对象，双目就要求有两个运算对象，三目则要三个运算对象。

表达式则是由运算及运算对象所组成的具有特定含义的式子。 C 语言是一种表达式语言，表达式后面加 " ； " 号就构成了一个表达式语句。

1.4 软件设计

2. C 语言的入门知识

（5）运算符和表达式

C 语言中常用的运算符：

① 赋值运算符

在 C 语言中用 "=" 这个符号来表示赋值运算符，就是将数据赋给变量。使用 "=" 的赋值语句格式如下：

变量 = 表达式;

② 算术、增减量运算符

算术、增减量运算符具体如表 3-2 所示。

操作符	作用说明	操作符	作用说明
+	加或取正值运算符	%	取余运算符
—	减或取负值运算符	--	减 1
*	乘运算符	++	加 1
/	除运算符		

2. C 语言的入门知识

（5）运算符和表达式

C 语言中常用的运算符：

③ 关系运算符

关系运算符反映的是两个表达式之间的大小等于关系，在 C 中有 6 种关系运算符：

> 大于
< 小于
>= 大于等于
<= 小于等于
== 等于
!= 不等于

这里的运算符有着优先级别。前四个具有相同的优先级，后两个也具有相同的优先级，但是前四个的优先级要高于后两个的。

1.4 软件设计

2. C 语言的入门知识

（ 5 ）运算符和表达式

C 语言中常用的运算符：
④ 逻辑运算符是用于求条件式的逻辑值。用逻辑运算符将关系表达式或逻辑量连接起来就是逻辑表达式了。逻辑表达式的一般形式为：
逻辑与： 条件式 1 && 条件式 2 0b0001&&0b1111 0001
逻辑或： 条件式 1 || 条件式 2 0b0001||0b1111 1111
逻辑非： ！条件式 2 !0b0101 0b1010

2. C 语言的入门知识

（5）运算符和表达式

C 语言中常用的运算符：
⑤ 位运算符的作用是按位对变量进行运算，但是并不改变参与运算的变量的值。如果要求按位改变变量的值，则要利用相应的赋值运算。还有就是位运算符是不能用来对浮点型数据进行操作的。 C51 中共有 6 种位运算符，分别如下：
" ~ " ： 按位取反
" << " ： 左移
" >> " ： 右移
" & " ： 按位与
" ^ " ： 按位异或
" | " ： 按位或
位运算一般的表达形式： 变量 1 位运算符 变量 2
位运算符也有优先级，从高到低依次是 :"~"(按位取反)→"<<"(左移) →">>"(右移) →"&"(按位与)→"^"(按位异或)→"|"(按位或)

2. C 语言的入门知识

（5）运算符和表达式

C 语言中常用的运算符：

⑥ 复合赋值运算符

复合赋值运算符就是在赋值运算符 "=" 的前面加上其他运算符。C 语言中的复合赋值运算符如下：

+=	加法赋值	>>=	右移位赋值
-=	减法赋值	&=	逻辑与赋值
*=	乘法赋值	|=	逻辑或赋值
/=	除法赋值	^=	逻辑异或赋值
%=	取模赋值	!=	逻辑非赋值
<<=	左移位赋值		

复合运算的一般形式为：　变量　复合赋值运算符　表达式

其含义就是变量与表达式先进行运算符所要求的运算，再把运算结果赋值给参与运算的变量。

2. C 语言的入门知识

（5）运算符和表达式

C 语言中常用的运算符：

⑦ 逗号运算符

逗号表达式的一般形式为：

表达式 1，表达式 2，表达式 3..... 表达式 n

这样用逗号运算符组成的表达式在程序运行时，是从左到右计算出各个表达式的值，而整个用逗号运算符组成的表达式的值等于最右边表达式的值，就是 " 表达式 n" 的值。

⑧ 条件运算符

就条件运算符 " ？ ： " 是 C51 中唯一的 1 个三目运算符，它要求有三个运算对象。用它以把三个表达式连接构成一个条件表达式。

条件表达式的一般形式如下：

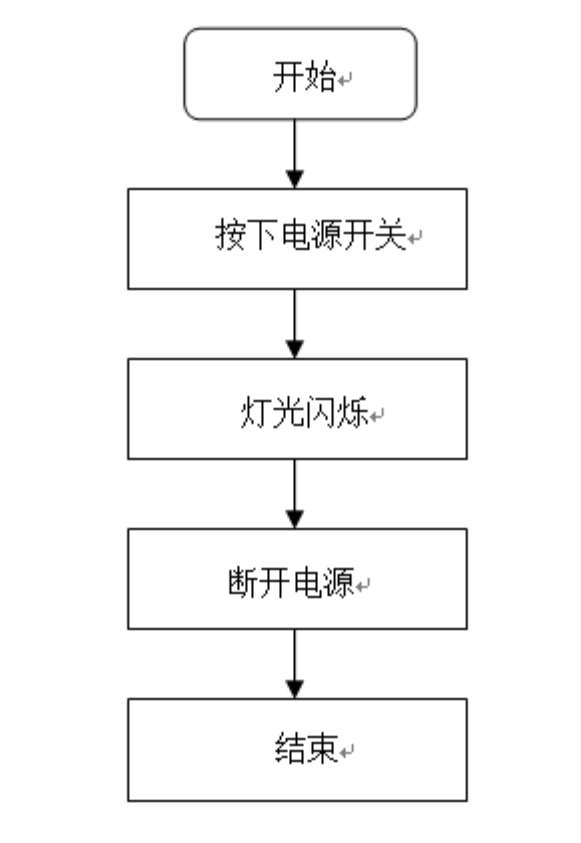
逻辑表达式？ 表达式 1 ： 表达式 2

条件运算符的作用是根据逻辑表达式的值选择使用表达式的值。当逻辑表达式的值为真时（非 0 值）时，整个表达式的值为表达式 1 的值；当逻辑表达式的值为假（值为 0 ）时，整个表达式的值为表达式 2 的值。

1.4 软件设计

3. 系统主程序设计

系统主程序设计流程图如图 3-4 所示。



1.4 软件设计

4. 软件编程

要使单片机能控制 LED 亮灭，需要编写相应的控制程序。将单片机 P2 口的每一个引脚都接有一只 LED，要实现 LED 循环点亮，就是要让各只 LED 依次点亮一定时间，熄灭一定时间，接着点亮下一只 LED 一定时间，然后再熄灭一定时间，重复循环。要实现电路功能，让 P2 口重复循环地输出低电平和高电平，实现流水灯流动的效果。表 3-3 列出了将要赋值给 P1 的数值。

发光二极管	D8	D7	D6	D5	D4	D3	D2	D1	P2 口输出（16 进制）	功能说明
P2 口	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0		
输出电平	1	1	1	1	1	1	1	0	0xfe	D1 点亮
	1	1	1	1	1	1	0	1	0xfd	D2 点亮
	1	1	1	1	1	0	1	1	0xfb	D3 点亮
	1	1	1	1	0	1	1	1	0xf7	D4 点亮
	1	1	1	0	1	1	1	1	0xef	D5 点亮
	1	1	0	1	1	1	1	1	0xdf	D6 点亮
	1	0	1	1	1	1	1	1	0xbf	D7 点亮
	0	1	1	1	1	1	1	1	0x7f	D8 点亮

1.4 软件设计

4. 软件编程

流水灯的程序：

```
#include <reg51.h>
unsigned char temp;
void delay05s(void)
{
  unsigned char i,j,k;
  for(i=50;i>0;i--)
  for(j=200;j>0;j--)
  for(k=248;k>0;k--);
}
void main(void)
{
   while(1)
    {
                temp=0xfe;
                P2=temp;
     delay05s();
                temp=0xfd;
                P2=temp;

     delay05s();
                temp=0xfb;
                P2=temp;
     delay05s();
                temp=0xf7;
                P2=temp;
     delay05s();
                temp=0xef;
                P2=temp;
     delay05s();
                temp=0xdf;
                P2=temp;
     delay05s();
                temp=0xbf;
                P2=temp;
     delay05s();
                temp=0x7f;
                P2=temp;
     delay05s();
    }
}
```

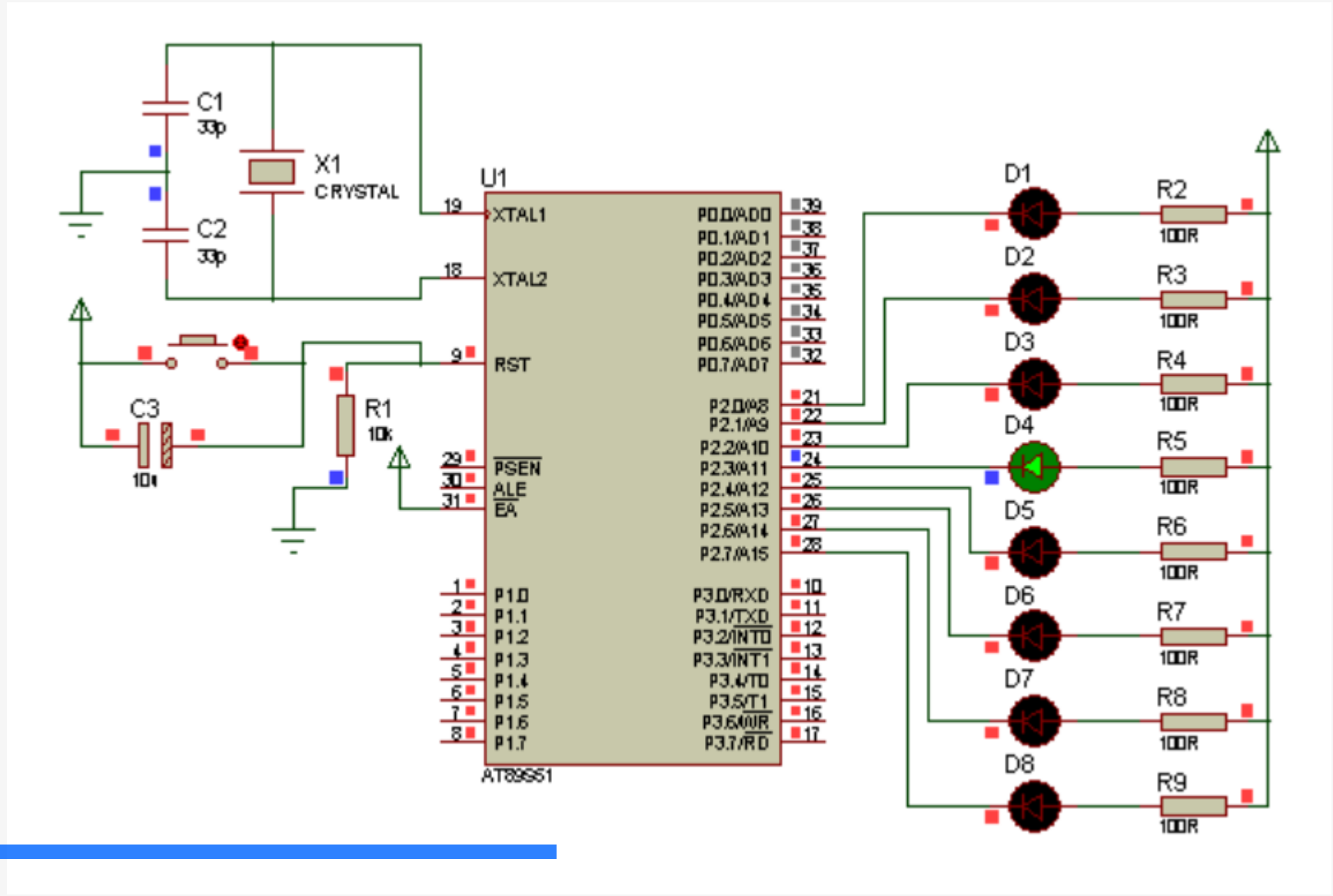
1.5 软件仿真

1. 仿真步骤

1. 使用 Proteus 软件，绘制如图 3-1 的硬件电路图，并保存到指定的位置。
2. 使用 KeilC51 建立一个流水灯工程项目，建立一个 .c 文件，在编辑区中输入上述的源程序，并以“ shiyan1.c” 为文件名存放流水灯设计文件夹中；进行编译，排除所有程序故障，直到无错误为止，目标代码文件“ shiyan1.hex” 。
3. 使用 Proteus 打开绘制好的按键电路设计图，双击电路图中 AT89C51 单片机，把编译好的“ shiyan1.hex” 文件下载到单片机中。点击模拟调试按钮，就可以看到流水灯的效果。具体仿真效果图如图 3-5 所示。

1.5 软件仿真

5. 仿真效果图



1.6 项目总结

本项目实现了流水灯设计与仿真，介绍了单片机 C 语言入门知识、 C 语言的基本的结构和顺序结构，并阐述了单片机实现流水灯编程控制的方法。

项目提升

1、C 语言编程主要特点是什么？

2、C 语言的基本结构是什么？

3、标识符是什么？有什么规定？

4、修改原程序，使 8 个 LED 暗点从 D8 到 D1 移动，循环不止。

5、修改原程序，实现依次点亮 D8、D7、D6、D5、D4、D3、D2、D1，（点亮时间间隔为 1 秒），重复循环。

1. 熟悉单片机 C 语言编程基础；
2. 掌握单片机 C 语言中的循环和选择结构；
3. 掌握单片机 C 语言的应用。

1. 会阐述单片机 C 语言编程基础相关知识；
2. 会使用单片机 C 语言中的循环和选择结构；
3. 会使用单片机 C 语言中的循环和选择结构控制 LED 的编程方法；

1. 能分析设计任务，正确单片机 C 语中的循环和选择结构；
2. 能熟练编写单片机控制 LED ；
3. 能使用 Proteus 软件绘制电路原理图；
4. 能使用 Keil 软件编译程序对 LED 的控制，并与 Proteus 软件联调，实现控制电路仿真；
5. 能够完成流跑马灯电路设计与仿真；

4 课时

1.1 任务提出

用单片机 P2 口接 8 个发光二极管 LED， LED 分别命名为

D1 、 D2 、 D3 、 D4 、 D5 、 D6 、 D7 、 D8 。使用 Keil 编写程序控制 8 个 LED 的亮灭状态。 K1 键是启动键， K2 键是停止键， K3 键可实现 LED 从左到右顺序点亮， K4 键是控制 LED 从右到左顺序点亮，程序利用循环移位函数 _crol_ 和 _cror_ 实现跑马灯效果。

1.2 方案设计

1. 单片机选择。采用 ATMEL 公司的 AT89C51 作为系统控制器的 CPU 方案。单片机算术运算功能强，软件编程灵活、自由度大，可以用软件编程实现各种算法和逻辑控制，并且由于其功耗低、体积小、技术成熟和成本低等优点，使其在各个领域应用广泛。

2.LED 电路的选择。每个发光二极管与单片机的一个引脚连接，一一对应。发光二极管的负极与单片机的引脚相连，正极通过一个限流电阻后与电源相连。

3. 按键电路的选择。根据题目的要求，采用独立式按键设计。每个按键与单片机的一个引脚连接，一一对应。按键的一端与单片机引脚相连，另外一端与地相连。

1.3 硬件设计

1. 电路组成

流水灯电路由单片机最小系统电路、八只发光二极管 LED 显示电路和按键电路组成。

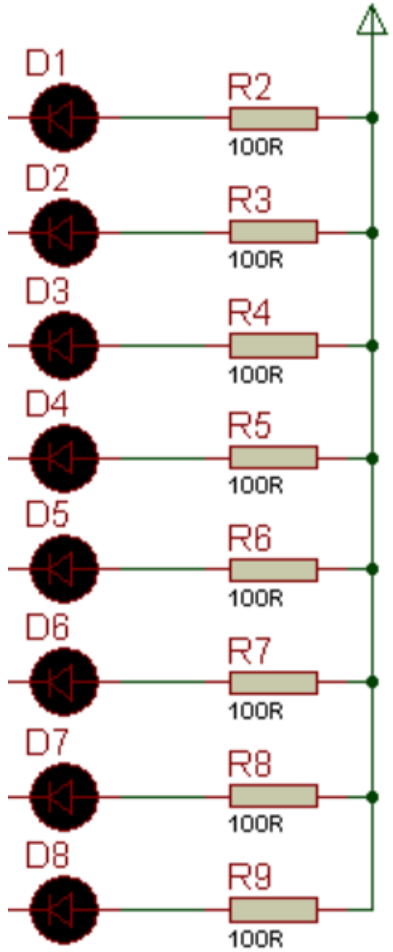
1.1 单片机最小系统电路

单片机最小系统电路以 AT89C51 单片机作为主控处理器，其 P2 口（可以根据需要使用 P0 、 P1 或 P3 口）接 8 只 LED 和 8 个限流电阻，四个按键分别接到 P1.2 、 P1.3 、 P1.4 和 P1.5 引脚。

1.3 硬件设计

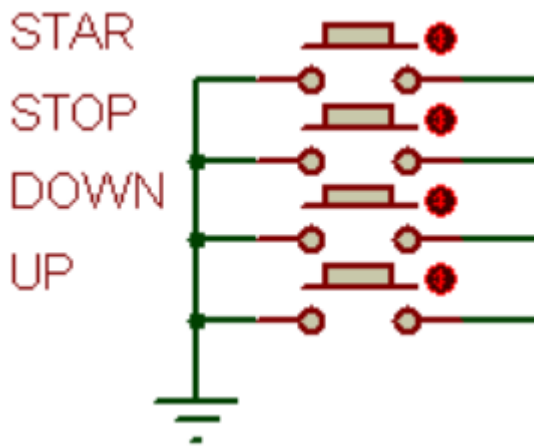
1.2 显示电路

显示电路由八个发光二极管和八个电阻组成。



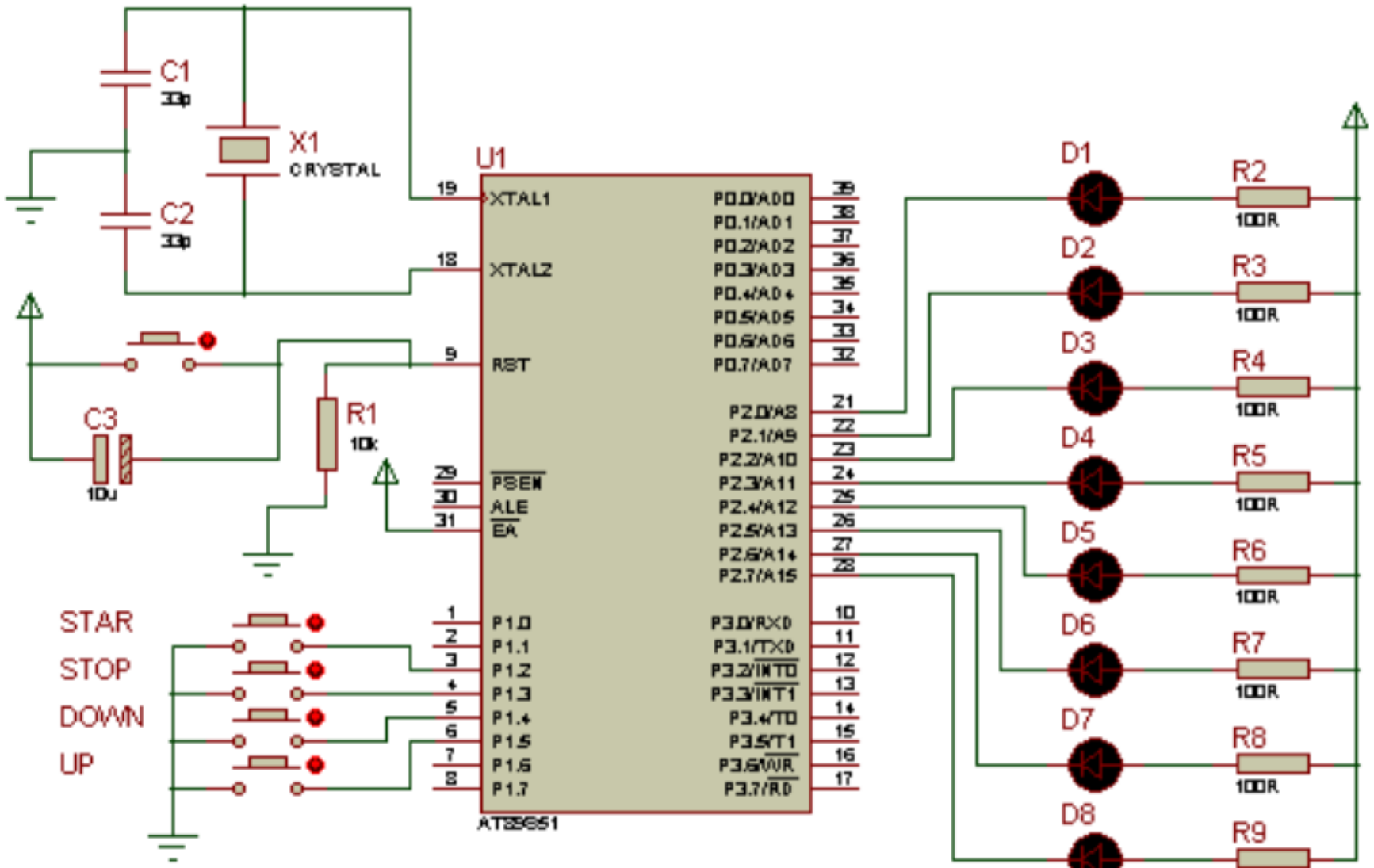
1.3 硬件设计

1.3 按键电路



1.3 硬件设计

2. 仿真电路



1.4 软件设计

1. 条件语句

条件语句的一般形式为：

```
if ( 表达式 )
    语句 1；
else
    语句 2；
```

如果表达式的值为非 0 (TURE) 即真，则执行语句 1，执行完语句 1 从语句 2 后开始继续向下执行；如果表达式的值为 0 (FALSE) 即假，则跳过语句 1 而执行语句 2，执行完语句 2 后继续向下执行。

所谓表达式是指关系表达式和逻辑表达式的结合式。 注意：

（1）条件执行语句中 "else 语句 2；" 部分是选择项，可以缺省，此时条件语句变成：

```
If ( 表达式 ) 语句 1; // 表示若表达式的值为非 0 则执行语句 1，否则跳过语句 1 继续执行
```

。

1.4 软件设计

1. 条件语句

（2）如果语句 1 或语句 2 有多于一条语句要执行时，必须使用"{" 和 "}" 把这些语句包括在其中，此时条件语句形式为：

```
if( 表达式 )
{
    语句体 1;
}
else
{
    语句体 2;
}
```

（3）条件语句可以嵌套，这种情况经常碰到，但条件嵌套语句容易出错，其原因主要是不知道哪个 if 对应哪个 else 。例如：

```
if (x >30 || x < -20)
if (y<=100 && y>x)
  printf("Right");
else
  printf("Error");
```

对于上述情况规定： else 语句与最近的一个 if 语句匹配，上例中的 else 与 if(y<=100&&y>x) 相匹配。为了使 else 与 if (x>30||x<-20) 相匹配，必须用花括号。如下所示：

```
if (x >30 || x < -20)
{
  if( y<=100 && y>x)
   printf("Right");
}
else
printf("Error");
```

1.4 软件设计

1. 条件语句

（4）可用阶梯式 if-else-if 结构

阶梯式结构的一般形式为：

```
if( 表达式  1)
      语句  1;
else if( 表达式 2)
     语句 2;
else if( 表达式  3)
     语句  3;
  .
  .
  .
else
     语句 n;
```

这种结构是从上到下逐个对条件进行判断，一旦发现条件满足就执行与它有关的语句，并跳过其它剩余阶梯；若没有一个条件满足，则执行最后一个 else 语句 n 。最后这个 else 常起着 " 缺省条件 " 的作用。同样，如果每一个条件中有多于一条语句要执行时，必须使用 "{" 和 "}" 把这些语句包括在其中。

1.4 软件设计

2. 循环语句

三种基本的循环语句 : for 语句、 while 语句和 do-while 语句。

（ 1 ） for 循环

for 循环是开界的。它的一般形式为：

for (< 初始化 > ； < 条件表过式 > ； < 增量 >)

语句;

初始化总是一个赋值语句，它用来给循环控制变量赋初值 ； 条件表达式是一个关系表达式，它决定什么时候退出循环 ； 增量定义循环控制变量每循环一次后按什么方式变化。这三个部分之间用 " ； " 分开。

例如：

```
for(i=1; i<=10; i++)
    语句 ;
```

上例中先给 i 赋初值 1 ，判断 i 是否小于等于 10 ，若是则执行语句，之后值增加 1 。再重新判断，直到条件为假，即 i>10 时，结束循环。

注意：

1 ） for 循环中语句可以为语句体，但要用 "{" 和 "}" 将参加循环的语句括起来。

2 ） for 循环中的 " 初始化 " 、 " 条件表达式 " 和 " 增量 " 都是选择项，即可以缺省 ， 但 " ； " 不能缺省。省略了初始化，表示不对循环控制变量赋初值。省略了条件表达式，则不做其它处理时便成为死循环。省略了增量 , 则不对循环控制变量进行操作 , 这时可在语句体中加入修改循环控制变量的语句。

3 ） for 循环可以有多层嵌套。

1.4 软件设计

2. 循环语句

（2） while 循环

while 循环的一般形式为：

while (条件)

语句；

while 循环表示当条件为真时，便执行语句。直到条件为假才结束循环。并继续执行循环程序外的后续语句。

与 for 循环一样， while 循环总是在循环的头部检验条件，这就意味着循环可能什么也不执行就退出。

注意：

1）在 while 循环体内也允许空语句。例如：

```
while((c=getche())!='\X0D');
```

这个循环直到键入回车为止。

2）可以有多层循环嵌套。

3）语句可以是语句体，此时必须用 "{" 和 "}" 括起来。

（3） do-while 循环

do-while 循环的一般格式为：

```
do
        语句;
While ( 条件 );
```

这个循环与 while 循环的不同在于：它先执行循环中的语句，然后再判断条件是否为真，如果为真则继续循环；如果为假，则终止循环。因此， do-while 循环至少要执行一次循环语句。同样当有许多语句参加循环时，要用 "{" 和 "}" 把它们括起来。

1.4 软件设计

3. 开关语句

在编写程序时，经常会碰到按不同情况分转的多路问题，这时可用嵌套 if-else-if 语句来实现，但 if-else-if 语句使用不方便，并且容易出错。对这种情况，应该应用开关语句。

开关语句格式为：

```
switch ( 变量 )
{
  case 常量 1 :
          语句 1 或空;
  case 常量 2 :
          语句 2 或空;
   .
   .
   . case 常量 n :
          语句 n 或空;
  Default :
          语句 n+1 或空;
 }
```

执行 switch 开关语句时，将变量逐个与 case 后的常量进行比较，若与其中一个相等，则执行该常量下的语句，若不与任何一个常量相等，则执行 default 后面的语句。

注意：

1）switch 中变量可以是数值，也可以是字符。

2）可以省略一些 case 和 default 。

3）每个 case 或 default 后的语句可以是语句体，但不需要使用 "{" 和 "}" 括起来。

1.4 软件设计

4.break 、 continue 语句

1 ） break 语句

break 语句通常用在循环语句和开关语句中。

当 break 用于开关语句 switch 中时， 可使程序跳出 switch 而执行 switch 以后的语句；

如果没有 break 语句，则将成为一个死循环而无法退出。

当 break 语句用于 do-while 、 for 、 while 循环语句中时， 可使程序终止循环而执行循环后面的语句， 通常 break 语句总是与 if 语句联在一起。 即满足条件时便跳出循环。

注意：

break 语句对 if-else 的条件语句不起作用。

在多层循环中，一个 break 语句只向外跳一层。

2 ） continue 语句

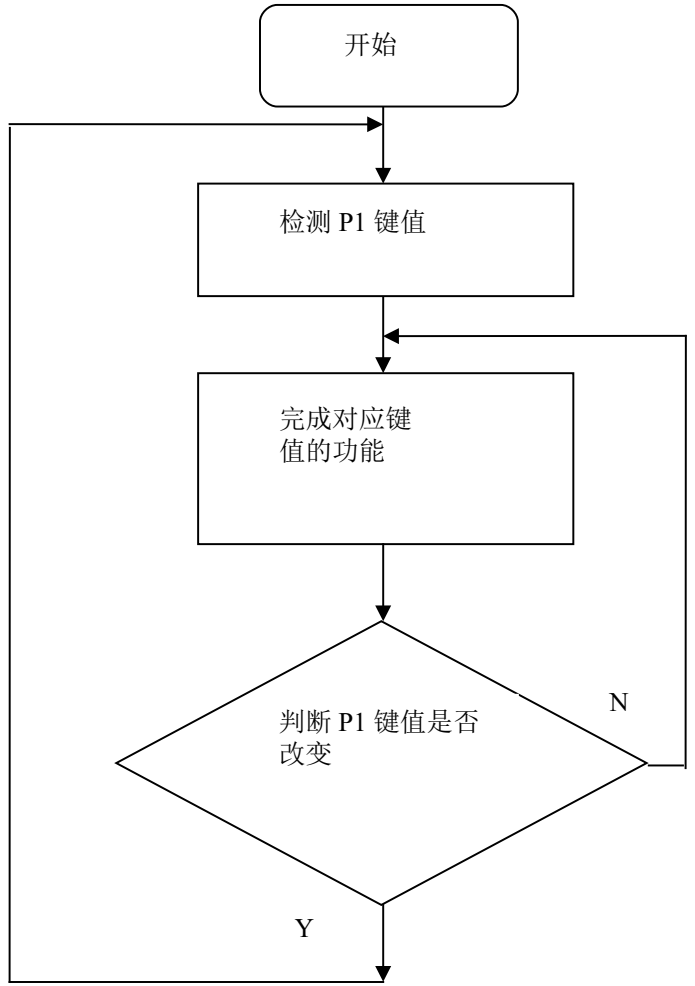
continue 语句的作用是跳过循环本中剩余的语句而强行执行下一次循环。

continue 语句只用在 for 、 while 、 do-while 等循环体中，常与 if 条件语句一起使用，用来加速循环。

1.4 软件设计

5. 系统主程序设计

系统主程序设计流程图如图 3.20 所示。



1.4 软件设计

6. 软件编程

设置四个按键，分别实现了启动、停止、向上和向下的功能。变量 KValue 用于保存四个按键按下时的不同数值，根据不同的 KValue 值判定不同的功能状态。除此，为了实现启动或停止、向上或向下等功能，定义了两个变量， Start 和 UpDown 。

。

按键功能说明：

① 当 Start 为 1 时， 系统启动；

② 当 Start 为 0 时， 系统停止；

③ 当 Start 为 1 的状态下， UpDown 为 1 时跑马灯由下向上流动； UpDown 为 0 时跑马灯由上向下流动；

④ 当 Start 为 0 的状态时， 系统处于停止状态， UpDown 不管是 0 或 1 ， 系统都无效。

1.4 软件设计

6. 软件编程

跑马灯功能实现的部分源码如下：

```
#include "reg51.h"
#include "intrins.h"
#define uchar unsigned char

sbit key1=P1^2;
sbit key2=P1^3;
sbit key3=P1^4;
sbit key4=P1^5;

uchar KValue;
```

```
void mDelay(unsigned int DelayTime)
{
        unsigned int  j=0;
        for(;DelayTime>0;DelayTime--)
        {
                for(j=0;j<125;j++);
  }
}
```

1.4 软件设计

6. 软件编程

```
void Key()
{           if(key1==0)
            {
                        KValue=1;
            }
            else if(key2==0)
            {
                        KValue=2;
            }
            else if(key3==0)
            {
                        KValue = 3;
            }
            else if(key4==0)
            {
                        KValue=4;
            }
            else
            KValue=0;
}
```

```
void main()
{
            unsigned char OutData=0xfe;
            bit UpDown=0;
            bit Start=0;

            for(;;)
            {
                        Key();
                        switch (KValue)
                        {
                        case 1:   //P1.2=0,Start
                        {
                         Start=1;
                         break;
                        }
```

1.4 软件设计

6. 软件编程

```
case 2:   //P1.3=0,Stop
{
 Start=0;
 break;
}
 case 3:    //P1.4=0 Up
{
UpDown=1;
 break;
}
case 4: //P1.5=0 Down
{
UpDown=0;
break;
 }
}
```

```
if(Start)
{
if(UpDown)
  OutData=_crol_(OutData,1);
 else
{ OutData=_cror_(OutData,1);
  P2=OutData;
}
else
  P2=0xff;// 否则灯全灭
  mDelay(1000);
}
```

1.5 软件仿真

1. 仿真步骤

1. 使用 Proteus 软件，绘制如图 3-6 的硬件电路图，并保存到指定的位置。

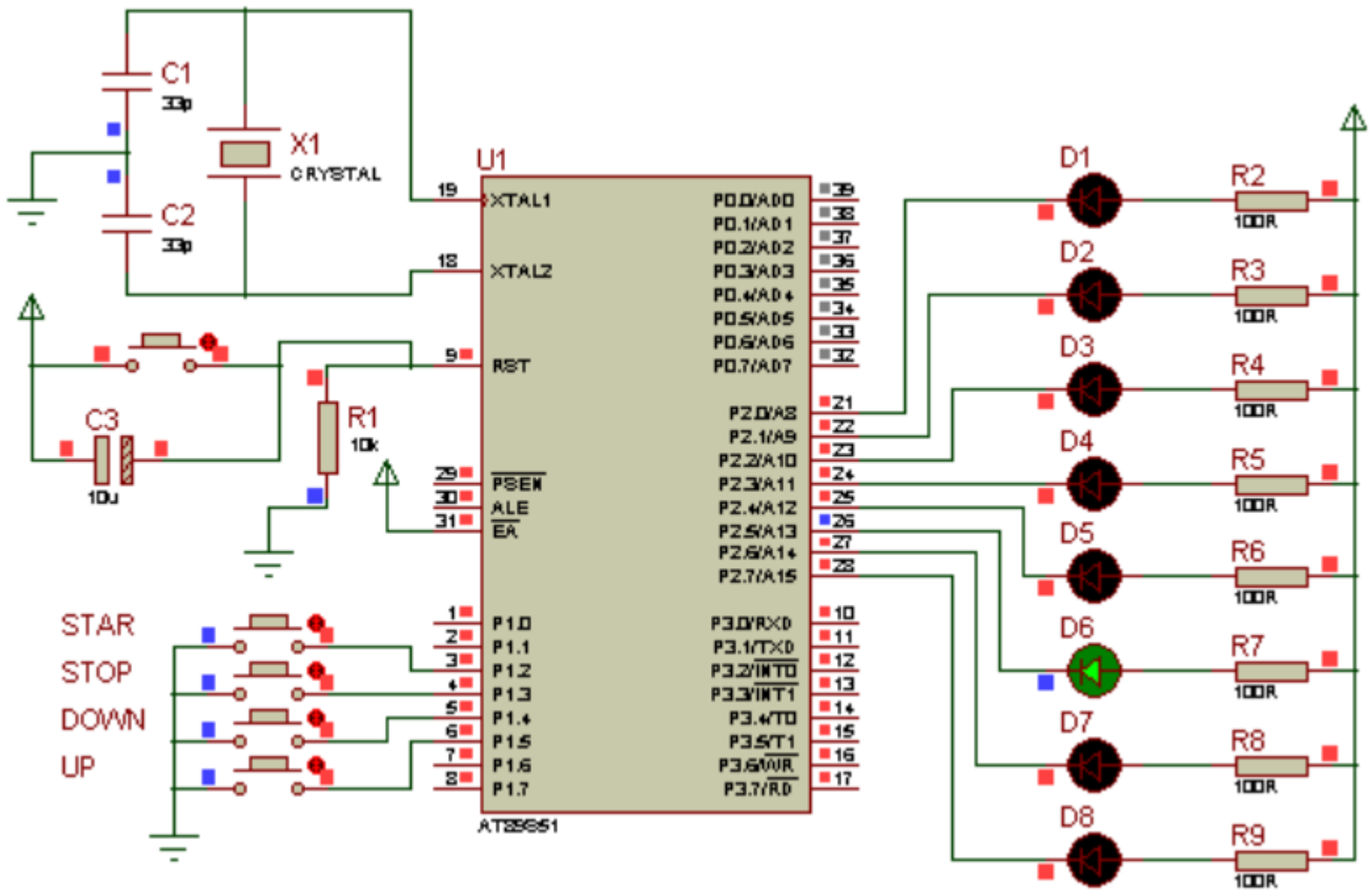
2. 使用 KeilC51 建立一个跑马灯工程项目，建立一个 .c 文件，在编辑区中输入上述的源程序，并以“ paomadeng.c” 为文件名存放跑马灯设计文件夹中；进行编译，排除所有程序故障，直到无错误为止，目标代码文件“ paomadeng.hex”。

3. 使用 Proteus 打开绘制好的按键电路设计图，双击电路图中 AT89S51 单片机，把编译好的“ paomadeng.hex” 文件下载到单片机中。点击模拟调试按钮，就可以看到跑马灯的效果。系统上电时，所有的 LED 全部熄灭，按下 start 按键，跑马灯开始流动，方向是从下向上；按下 stop 按键，跑马灯流动停止。在跑马灯流动的状态下，按下 down 按键，跑马灯改变流动的方向，由上向下流动，再按 up 按键，方向是从下向上，以次循环。具体仿真效果图如图 3-5 所示。

1.5 软件仿真

2. 仿真效果图

跑马灯仿真效果如图 3-7 所示。



1.6 项目总结

本项目实现了跑马灯设计与仿真，介绍了单片机 C 语言的数据类型、 C 语言的运算符和表达式、循环结构和选择结构，并阐述了单片机实现跑马灯编程控制的方法

项目提升

1、 C 语言的基本结构有哪三种？

2、循环三种基本的语句分别是什么？各有什么特点？

3、 break 语句的作用是什么？如何使用？

4、开关语句 switch 使用过程中注意什么？

5、修改原程序，按下四个按键分别控制 4 种不同的 LED 显示效果。

1. 熟悉单片机 C 语言编程基础；
2. 掌握单片机 C 语中的数组知识；
3. 掌握单片机 C 语言的应用。

1. 会阐述单片机 C 语言编程基础相关知识；
2. 会使用单片机 C 语中的一维数组；
3. 会使用单片机 C 语言中的一维数组控制 LED 的编程方法；

1. 能分析设计任务，正确单片机 C 语中的循一维数组；
2. 能熟练编写单片机控制 LED ；
3. 能使用 Proteus 软件绘制电路原理图；
4. 能使用 Keil 软件编译程序对 LED 的控制，并与 Proteus 软件联调，实现控制电路仿真；
5. 能够完成多样彩灯电路设计与仿真；

4 课时

1.1 任务提出

用单片机 P2 口接 8 个发光二极管 LED， LED 分别命名为
D1 、 D2 、 D3 、 D4 、 D5 、 D6 、 D7 、 D8 。使用 Keil 编写程序控制 8 个 LED 先从左到右顺序点亮，然后从右到左顺序点亮，再从两边向中间依次点亮，最后从中间向两边依次点亮的程序（分别称为右移、左移、两端向中间移、中间向两端移），再从头开始，循环不停止，产生一种多样彩灯的效果。

1.2 方案设计

1. 单片机选择。采用 ATMEL 公司的 AT89C51 作为系统控制器的 CPU 方案。单片机算术运算功能强，软件编程灵活、自由度大，可以用软件编程实现各种算法和逻辑控制，并且由于其功耗低、体积小、技术成熟和成本低等优点，使其在各个领域应用广泛。

2.LED 电路的选择。每个发光二极管与单片机的一个引脚连接，一一对应。发光二极管的负极与单片机的引脚相连，正极通过一个限流电阻后与电源相连。

1.3 硬件设计

1. 电路组成

流水灯电路由单片机最小系统电路、八只发光二极管 LED 显示电路组成。

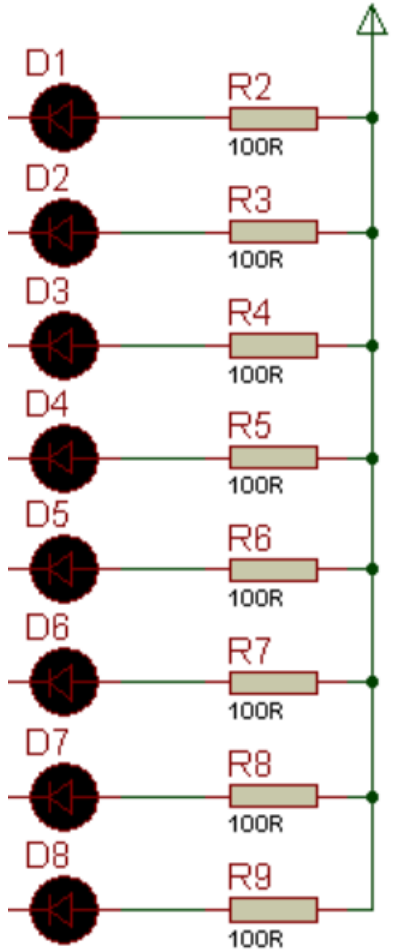
1.1 单片机最小系统电路

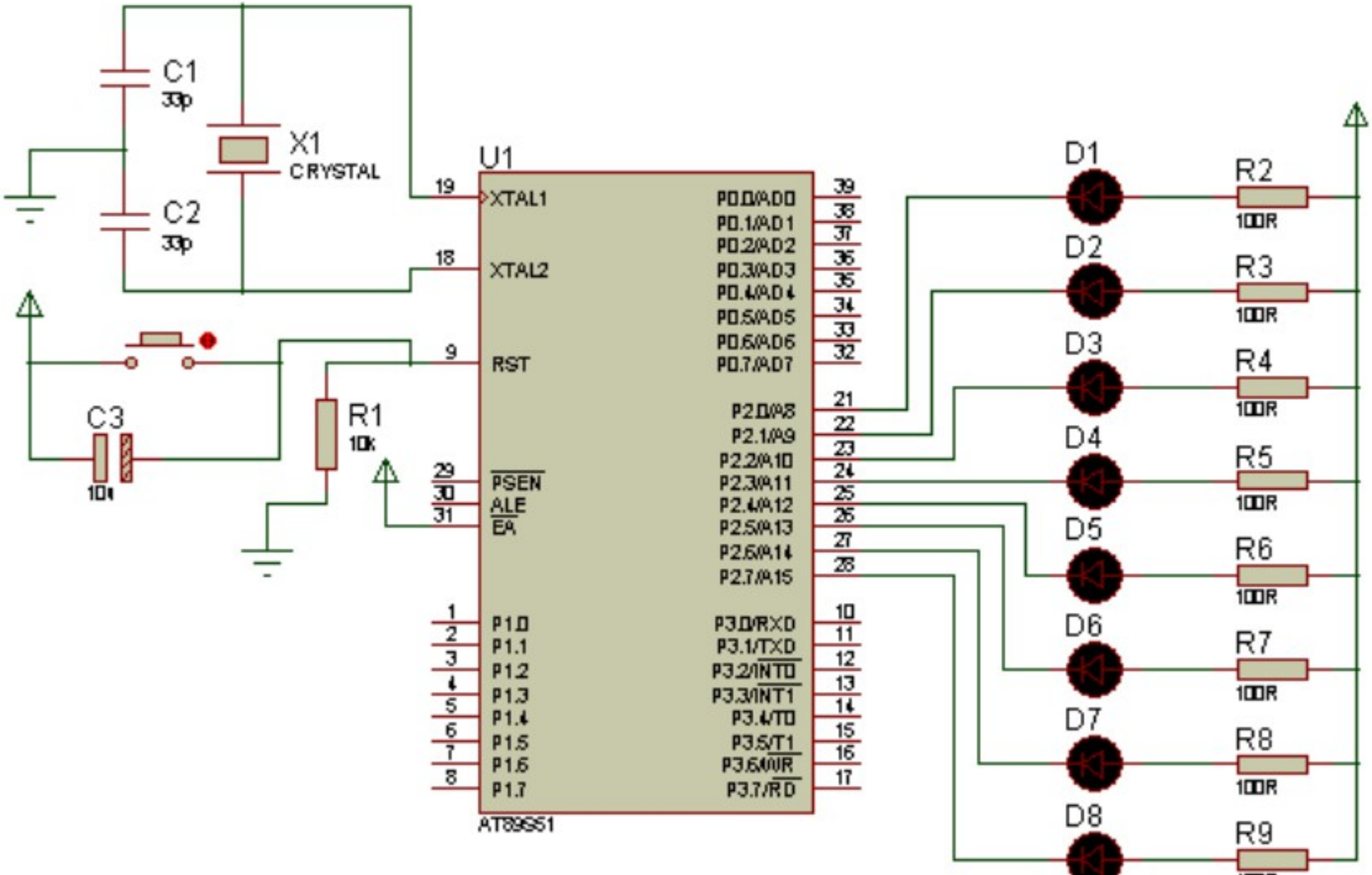
单片机最小系统电路以 AT89C51 单片机作为主控处理器，其 P2 口（可以根据需要使用 P0 、 P1 或 P3 口）接 8 只 LED 和 8 个限流电阻

1.3 硬件设计

1.2 显示电路

显示电路由八个发光二极管和八个电阻组成。




C1
33p
X1
CRYSTAL
C2
33p
U1
19
XTAL1
18
XTAL2
9
RST
29
PSEN
30
ALE
31
EA
C3
10u
R1
10k
1
P1.0
2
P1.1
3
P1.2
4
P1.3
5
P1.4
6
P1.5
7
P1.6
8
P1.7
AT89S51
39
P0.0/AD0
38
P0.1/AD1
37
P0.2/AD2
36
P0.3/AD3
35
P0.4/AD4
34
P0.5/AD5
33
P0.6/AD6
32
P0.7/AD7
21
P2.0/A8
22
P2.1/A9
23
P2.2/A10
24
P2.3/A11
25
P2.4/A12
26
P2.5/A13
27
P2.6/A14
28
P2.7/A15
10
P3.0/RXD
11
P3.1/TXD
12
P3.2/INT0
13
P3.3/INT1
14
P3.4/T0
15
P3.5/T1
16
P3.6/WR
17
P3.7/RD
D1
D2
D3
D4
D5
D6
D7
D8
R2
100R
R3
100R
R4
100R
R5
100R
R6
100R
R7
100R
R8
100R
R9

1.4 软件设计

1. 一维数组

（1）一维数组的定义方式

在 C 语言中使用数组必须先进行定义。一维数组的定义方式为：

类型说明符 数组名 [常量表达式];

其中，类型说明符是任一种基本数据类型或构造数据类型。数组名是用户定义的数组标识符。方括号中的常量表达式表示数据元素的个数，也称为数组的长度。

对于数组类型说明应注意以下几点：

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，其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) 数组名不能与其它变量名相同。

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。但是其下标从 0 开始计算。因此 5 个元素分别为 a[0], a[1], a[2], a[3], a[4] 。

5) 不能在方括号中用变量来表示元素的个数，但是可以是符号常数或常量表达式。

6) 允许在同一个类型说明中，说明多个数组和多个变量。

1.4 软件设计

1. 一维数组

（2）一维数组元素的引用

数组元素是组成数组的基本单元。数组元素也是一种变量， 其标识方法为数组名后跟一个下标。下标表示了元素在数组中的顺序号。数组元素的一般形式为：

数组名 [下标]

其中下标只能为整型常量或整型表达式。如为小数时， C 编译将自动取整。数组元素通常也称为下标变量。必须先定义数组，才能使用下标变量。在 C 语言中只能逐个地使用下标变量，而不能一次引用整个数组。也不能用一个语句输出整个数组。

1.4 软件设计

1. 一维数组

（3）一维数组的初始化

给数组赋值的方法除了用赋值语句对数组元素逐个赋值外， 还可采用初始化赋值和动态赋值的方法。

数组初始化赋值是指在数组定义时给数组元素赋予初值。数组初始化是在编译阶段进行的。这样将减少运行时间，提高效率。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 { } 中的各数据值即为各元素的初值，各值之间用逗号间隔。

C 语言对数组的初始化赋值还有以下几点规定：

1) 可以只给部分元素赋初值。当 { } 中值的个数少于元素个数时，只 给前面部分元素赋值。表示只给 a[0] ～ a[4]5 个元素赋值，而后 5 个元素自动赋 0 值。

2) 只能给元素逐个赋值，不能给数组整体赋值。

3) 如给全部元素赋值，则在数组说明中，可以不给出数组元素的个数。

一维数组程序举例：可以在程序执行过程中，对数组作动态赋值。这时可用循环语句配合 scanf 函数逐个对数组元素赋值。

C 语言支持一维数组和多维数组。如果一个数组的所有元素都不是数组，那么该数组称为一维数组。

1.4 软件设计

1. 一维数组

（4）字符数组

用来存放字符数据的数组称为字符数组。是一种常用的数组。在 C 语言中，字符数组用于存放一组字符或字符串。字符串以“ \0”作为结束符，只存放一般字符的字符数组的赋值与使用和一般的完全数组相同。

例如： char string[10];

Char string[20];

上面定义了两个字符数组，分别定义了 10 个元素和 20 个元素。

1.4 软件设计

2. 程序设计

（1）首先将所需要显示的效果对应的数值保存在一维数组中。
（2）程序运行时，从数组逐个取出数值，送到端口显示即可。

```
#include "reg51.h"// 预处理文件里面定义了特殊寄存器的名称如 P1 口定义为 P1
void main(void)
{
// 定义花样数据
const unsigned char
design[32]={0xFF,0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F,0x7F,0xBF,0xDF,0xEF,0xF7,0xFB,0xFD,0xFE,0xFF,0xFF,0xFE,0xFC,0xF8,0xF0,0xE0,0xC0,0x80,0x0,0xE7,0xDB,0xBD,0x7E,0xFF};

}
```

```
    unsigned int a; // 定义循环用的变量
    unsigned char b; // 在 C51 编程中因内存有限尽可能注意变量类型的使用
    // 尽可能使用少字节的类型，在大型的程序中很受用
    do{
    for (b=0; b<32; b++)
    {
    for(a=0; a<30000; a++); // 延时一段时间
    P2 = design[b]; // 读已定义的花样数据并写花样数据到 P1 口
    }
    }
    while(1);
    }
```

1.5 软件仿真

1. 仿真步骤

1. 使用 Proteus 软件，绘制如图 3-8 的硬件电路图，并保存到指定的位置。

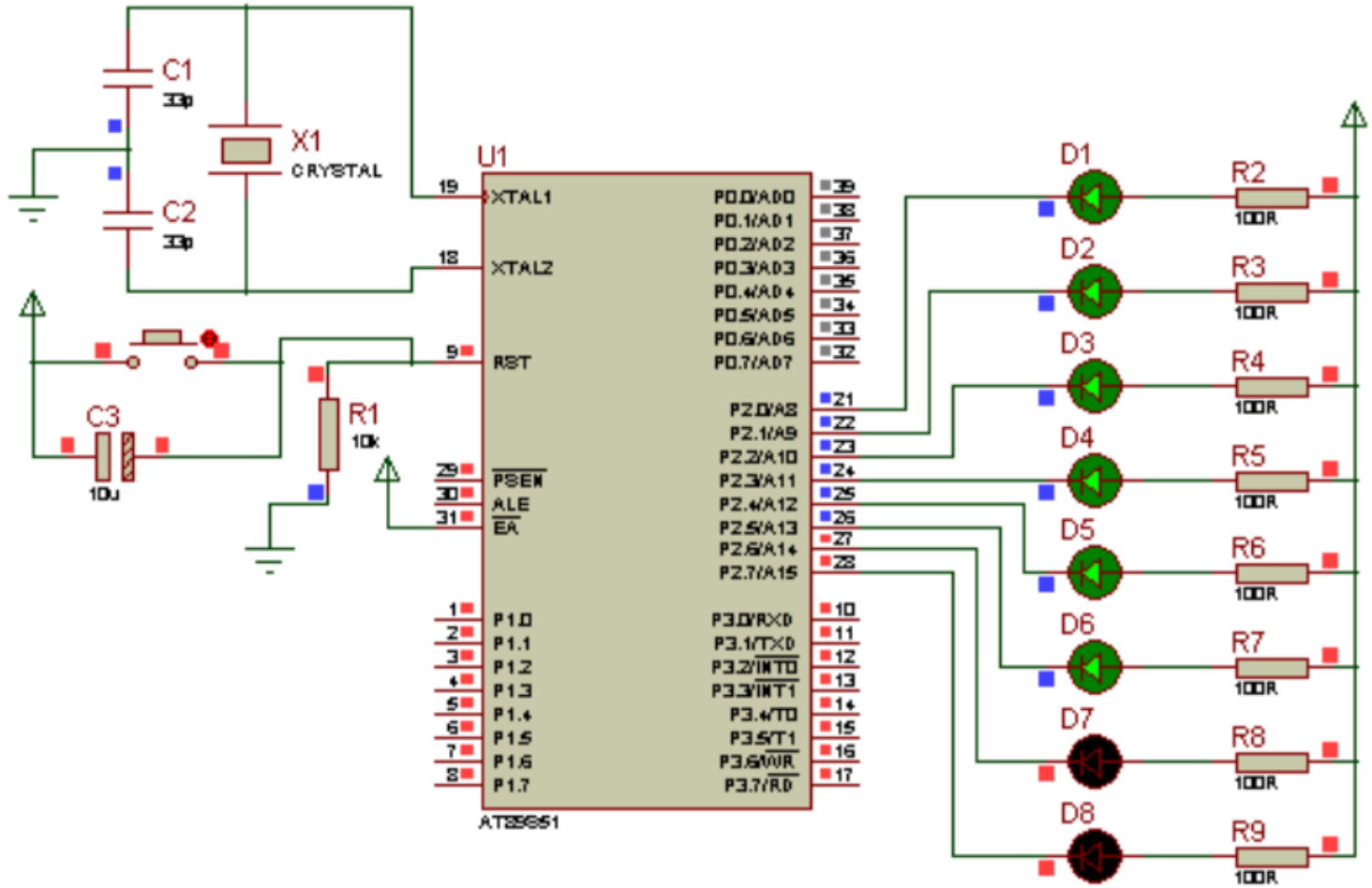
2. 使用 KeilC51 建立一个多样彩灯工程项目，建立一个 .c 文件，在编辑区中输入上述的源程序，并以“ caideng.c” 为文件名存放多样彩灯计文件夹中；进行编译，排除所有程序故障，直到无错误为止，目标代码文件“ caideng.hex” 。

3. 使用 Proteus 打开绘制好的电路设计图，双击电路图中 AT89C51 单片机，把编译好的“ caideng.hex” 文件下载到单片机中。点击模拟调试按钮，就可以看到多样彩灯运行的效果。具体仿真效果图如图 3-9 所示。

1.5 软件仿真

2. 仿真效果图

多样彩灯仿真效果如图 3-9 所示



1.6 项目总结

本项目实现了流水灯设计与仿真，介绍了单片机 C 语言的一维数组的基本知识，并阐述了单片机实现多样彩灯编程控制的方法。

项目提升

1、一维数组的如何定义？试简述。

2、一维数组元素的引用有什么规定？

3、一维数组如何初始化，有什么规定？

4、字符数组如何定义？

5、修改原程序，实现不同的效果。