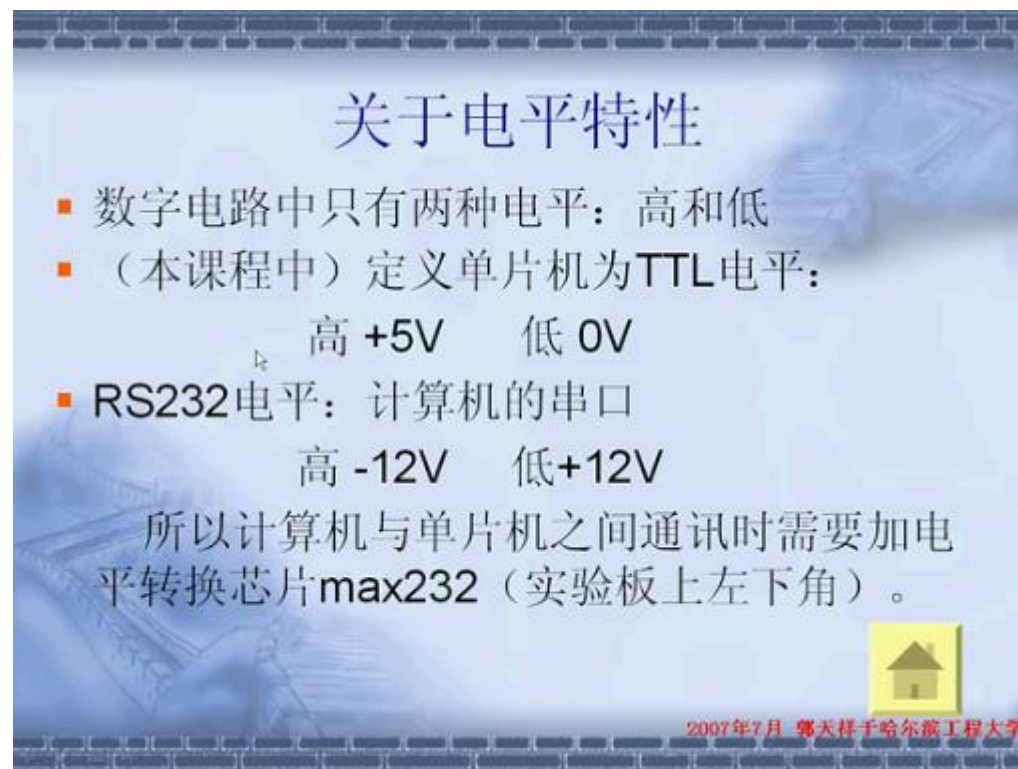


一小时学会 C51 单片机 <http://www.51hei.com/bbs/dpj-43893-1.html>

51 单片机视频教程 <http://www.51hei.com/sp/>


一. 基础



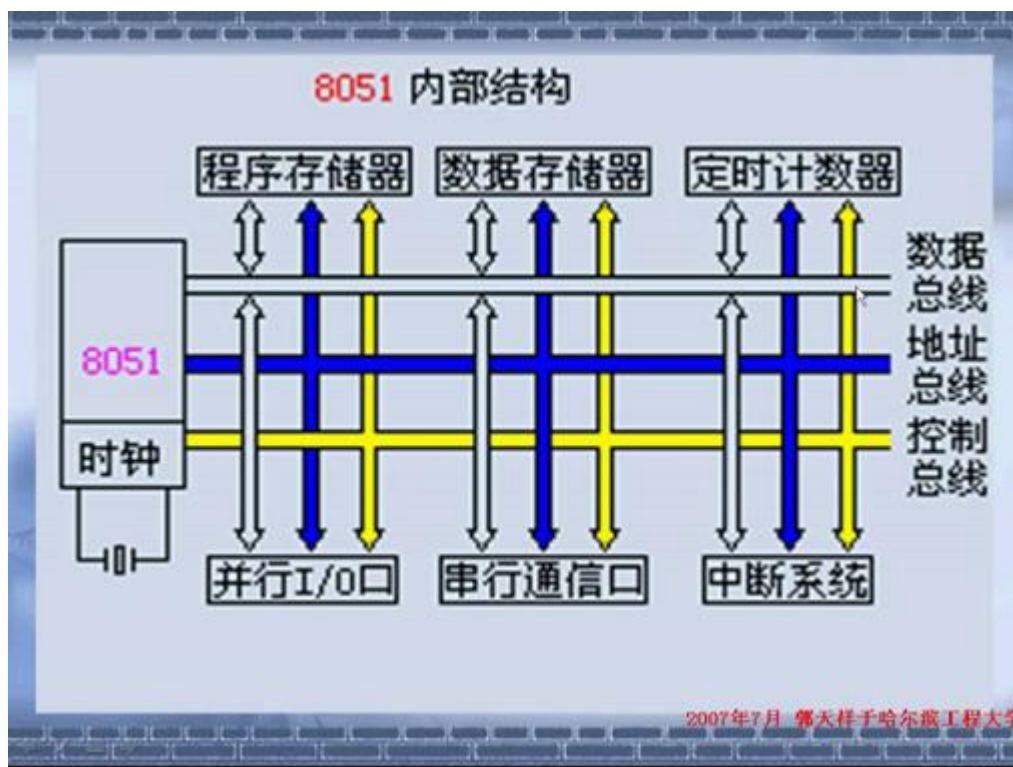
关于电平特性

- 数字电路中只有两种电平：高和低
- （本课程中）定义单片机为TTL电平：
高 +5V 低 0V
- RS232电平：计算机的串口
高 -12V 低+12V

所以计算机与单片机之间通讯时需要加电平转换芯片max232（实验板上左下角）。



2007年7月 郭天祥于哈尔滨工程大学



总线（BUS）是计算机各部件之间传送信息的公共通道。微机中有内部总线和外部总线两类。内部总线是CPU内部之间的连线。外部总线是指CPU与其它部件之间的连线。外部总线有三种：数据总线DB（Data Bus），地址总线 AB（Address Bus）和控制总线 CB（Control Bus）。

2007年7月 郭天祥于哈尔滨工程大学

- **CPU:** 由运算和控制逻辑组成，同时还包括中断系统和部分外部特殊功能寄存器；
- **RAM:** 用以存放可以读写的数据，如运算的中间结果、最终结果以及欲显示的数据；
- **ROM:** 用以存放程序、一些原始数据和表格；
- **I/O口:** 四个8位并行I/O口，既可用作输入，也可用作输出；
- **T/C:** 两个定时/计数器，既可以工作在定时模式，也可以工作在计数模式；

2007年7月 郭天祥于哈尔滨工程大学

C-51与ASM-51相比，有如下优点：

1. 对单片机的指令系统不要求了解，仅要求对8051 的存贮器结构有初步了解；
2. 寄存器分配、不同存贮器的寻址及数据类型等细节可由编译器管理；
3. 程序有规范的结构，可分成不同的函数，这种方式可使程序结构化；
4. 提供的库包含许多标准子程序，具有较强的数据处理能力；
5. 由于具有方便的模块化编程技术，使已编好程序可容易地移植；

2007年7月 郭天祥于哈尔滨工程大学

C-51的数据类型

基本数据类型

类型	符号	关键字	所占位数	数的表示范围
整型	有	(signed) int	16	-32768~32767
		(signed) short	16	-32768~32767
		(signed) long	32	-2147483648~2147483647
	无	unsigned int	16	0~65535
		unsigned short int	16	0~65535
		unsigned long int	32	0~4294967295
实型	有	float	32	3.4e-38~3.4e38
	有	double	64	1.7e-308~1.7e308
字符型	有	char	8	-128~127
	无	unsigned char	8	0~255

2007年7月 郭天祥于哈尔滨工程大学

C-51的数据类型扩充定义

sfr: 特殊功能寄存器声明

sfr16: sfr的16位数据声明

sbit: 特殊功能位声明

bit: 位变量声明

例: `sfr SCON = 0X98;`

`sfr16 T2 = 0xCC;`

`sbit OV = PSW^2;`

2007年7月 郭天祥于哈尔滨工程大学

C-51的包含的头文件

通常有: `reg51.h` `reg52.h` `math.h` `cctype.h`
`stdio.h` `stdlib.h` `absacc.h`

常用有: `reg51.h` `reg52.h`

(定义特殊功能寄存器和位寄存器);

`math.h` (定义常用数学运算);

2007年7月 郭天祥于哈尔滨工程大学

C-51的运算符

与C语言基本相同:

{	+	-	*	/	(加 减 乘 除)
	>	>=	<	<=	(大于 大于等于 小于 小于等于)
	=	!=	(测试等于 测试不等于)		
	&&		!	(逻辑与 逻辑或 逻辑非)	
{	>>	<<	(位右移 位左移)		
	&		(按位与 按位或)		
	^	~	(按位异或 按位取反)		

2007年7月 郭天祥于哈尔滨工程大学

单片机主要掌握以下几点

- 最小系统能够运行起来的必要条件。
 - 1.电源 2.晶振3.复位电路
- 对单片机任意IO口的随意操作
 - 1.输出控制电平高低2.输出检测电平高低。
- 定时器：重点掌握最常用的方式2
- 中断：外部中断、定时器中断、串口中断
- 串口通信：单片机之间、单片机与计算机间

2007年7月 郭天祥于哈尔滨工程大学

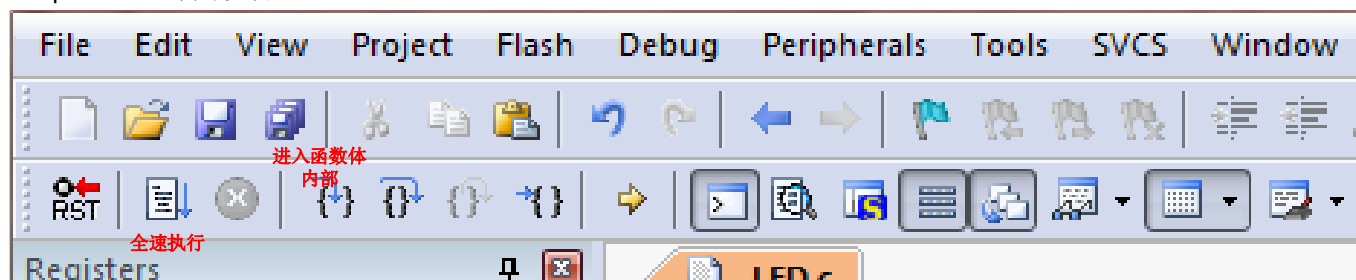
reg52.h 头文件的存放目录：

F:\Software\Keil_v5\C51\INC\ reg52.h

它定义了常用的 52、32 类型单片机寄存器的地址，里面定义过的变量，程序中可直接使用。同时也可以在这个头文件中添加自己的定义。可以看到寄存器声明 sfr 定义了 P0 的内存器地址。

软件调试/也可硬件调试（在调试模式下，需修改 Options-Debug-右边的 Use（当前已经连接了硬件调试设备，左边是软件仿真）-右边可选的都可以打钩）。

Peripherals（外围设备）



Reset: 复位CPU。
指回程序开始位置

单步执行：一次执行
一条语句。箭头所指
程序是下一步要执行
的语句。

单片机工作的基本时序

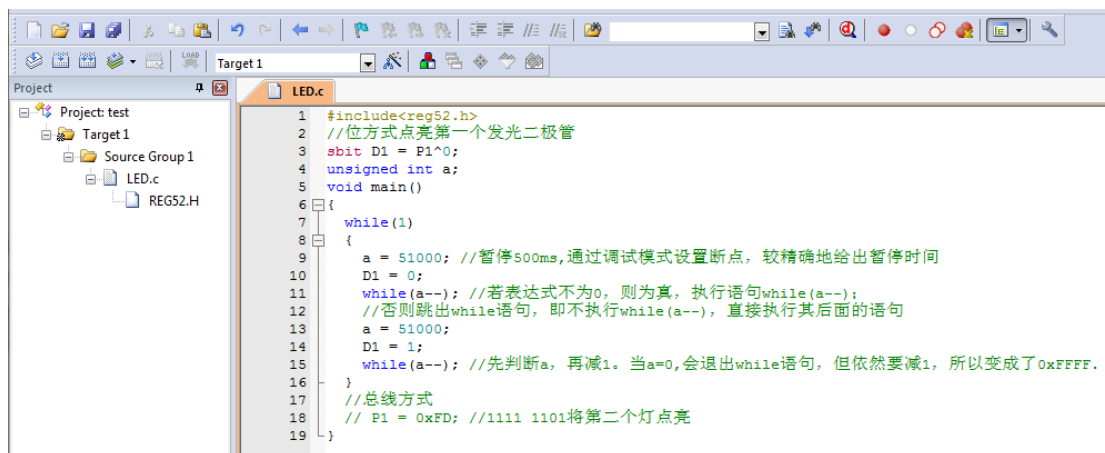
- 机器周期和指令周期
- (1) 振荡周期: 也称时钟周期, 是指为单片机提供时钟脉冲信号的振荡源的周期, **TX实验板上为11.0592MHZ。**
- (2) 状态周期: 每个状态周期为时钟周期的 **2** 倍, 是振荡周期经二分频后得到的。
- (3) 机器周期: **一个机器周期** 包含 **6** 个状态周期 **S1~S6**, **也就是 12 个时钟周期**。在一个机器周期内, **CPU** 可以完成一个独立的操作。
- (4) 指令周期: 它是指 **CPU** 完成一条操作所需的全部时间。每条指令执行时间都是有一个或几个机器周期组成。 **MCS - 51** 系统中, 有单周期指令、双周期指令和四周期指令。

一次振荡周期 (时钟周期, 晶振的 $1/\text{频率}$), 输出一个矩形脉冲。

状态周期是时钟周期的两倍, 则频率是时钟频率的一半, 即是经过时钟周期二分频得到的。机器周期就是访问一次存储器的时间。而 1 个机器周期包括 12 个时钟周期。如果单片机工作在 12M 晶体下, 那么一个时钟周期为: $1/12$ 微妙。一个机器周期 $12 \times 1/12 = 1$ 微妙。

但是在软件中修改时钟周期, 并不会影响单片机的时钟周期, 因为单片机的时钟周期是由它自己的硬件结构决定的。所以软件中设置的时钟周期应当与单片机时钟周期一样。

在任何某一时刻, 有且只有一个数码管能发光。但可以利用人眼的暂留效应和程序的快速运行。“同时”使多个数码管亮。



```
1 #include<reg52.h>
2 //位方式点亮第一个发光二极管
3 sbit D1 = P1^0;
4 unsigned int a;
5 void main()
6 {
7     while(1)
8     {
9         a = 51000; //暂停500ms,通过调试模式设置断点,较精确地给出暂停时间
10        D1 = 0;
11        while(a--); //若表达式不为0,则为真,执行语句while(a--);
12        //否则跳出while语句,即不执行while(a--),直接执行其后面的语句
13        a = 51000;
14        D1 = 1;
15        while(a--); //先判断a,再减1。当a=0,会退出while语句,但依然要减1,所以变成了0xFFFF.
16    }
17    //总线方式
18    // P1 = 0xFD; //1111 1101将第二个灯点亮
19 }
```

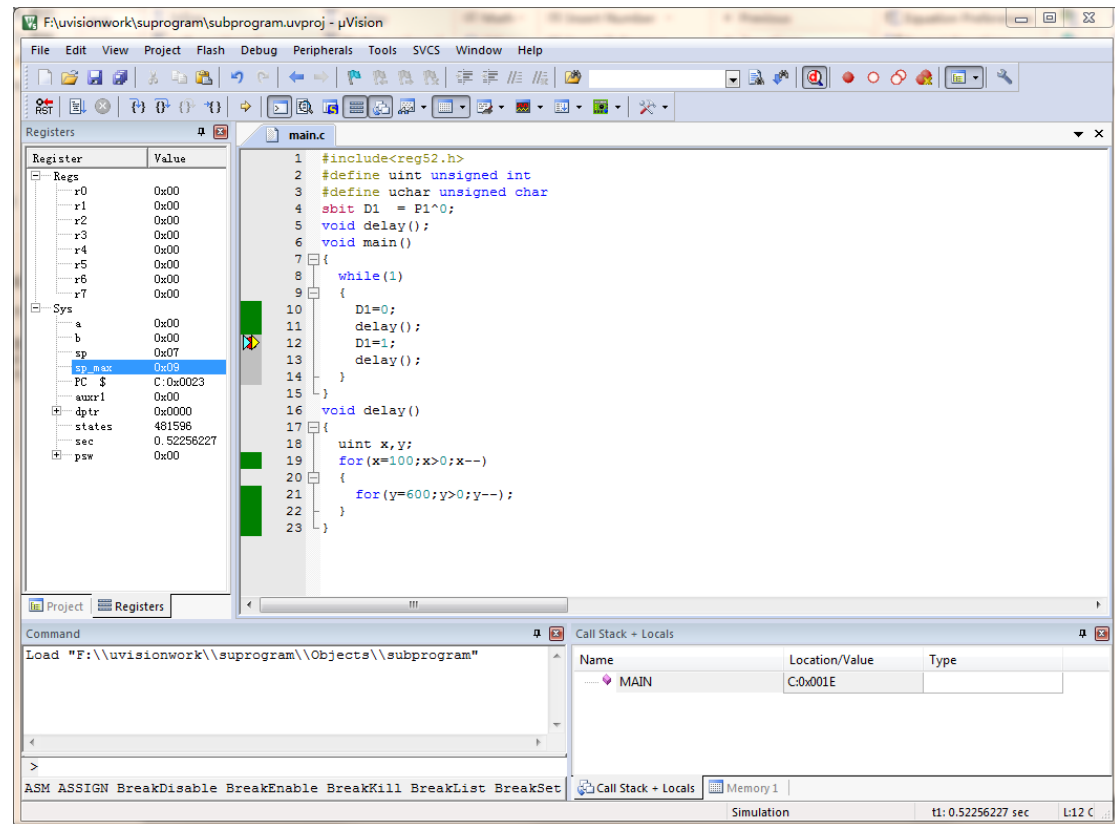
子程序调用:

宏定义

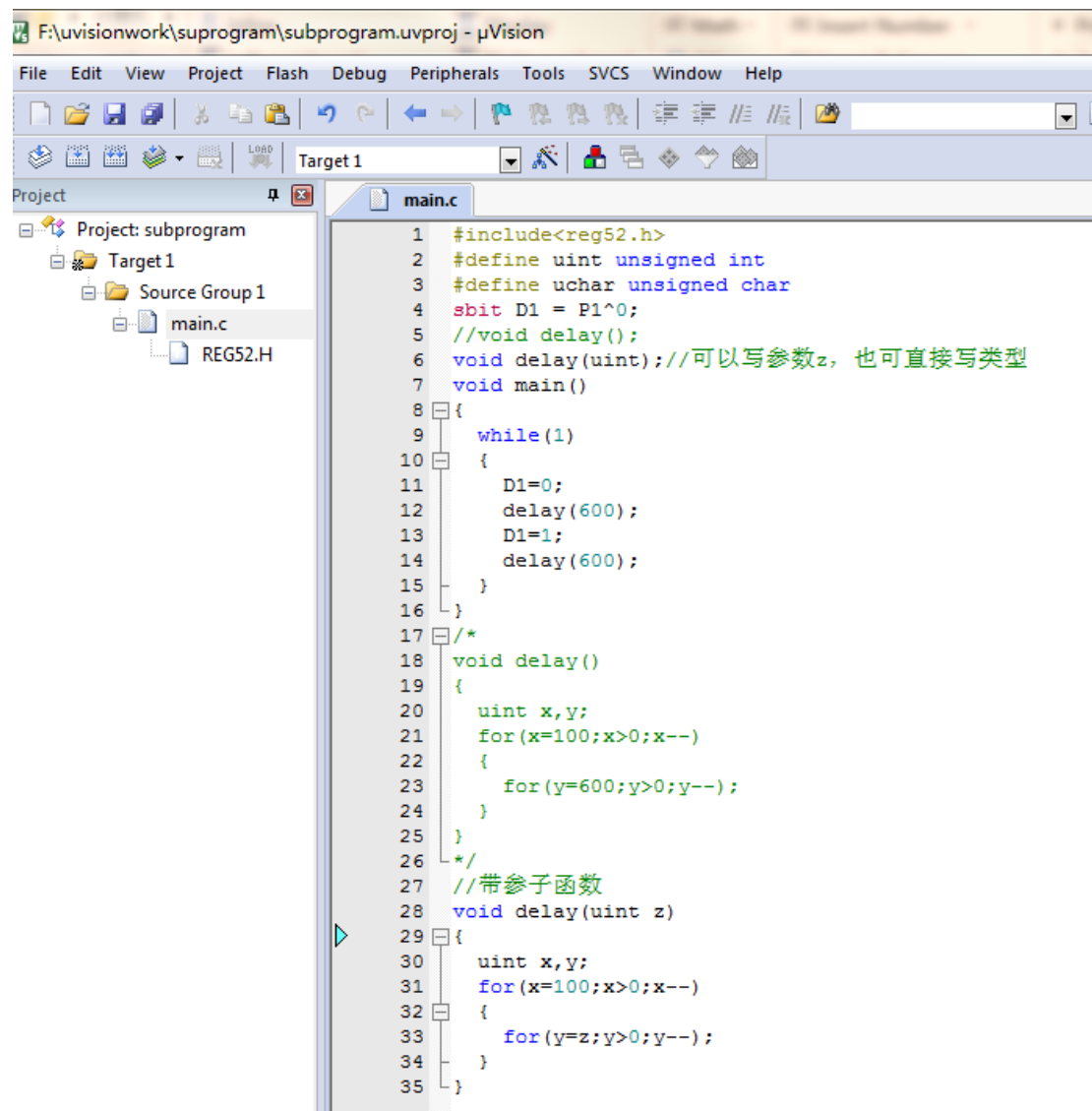
#define uint unsigned int

#define uchar unsigned char

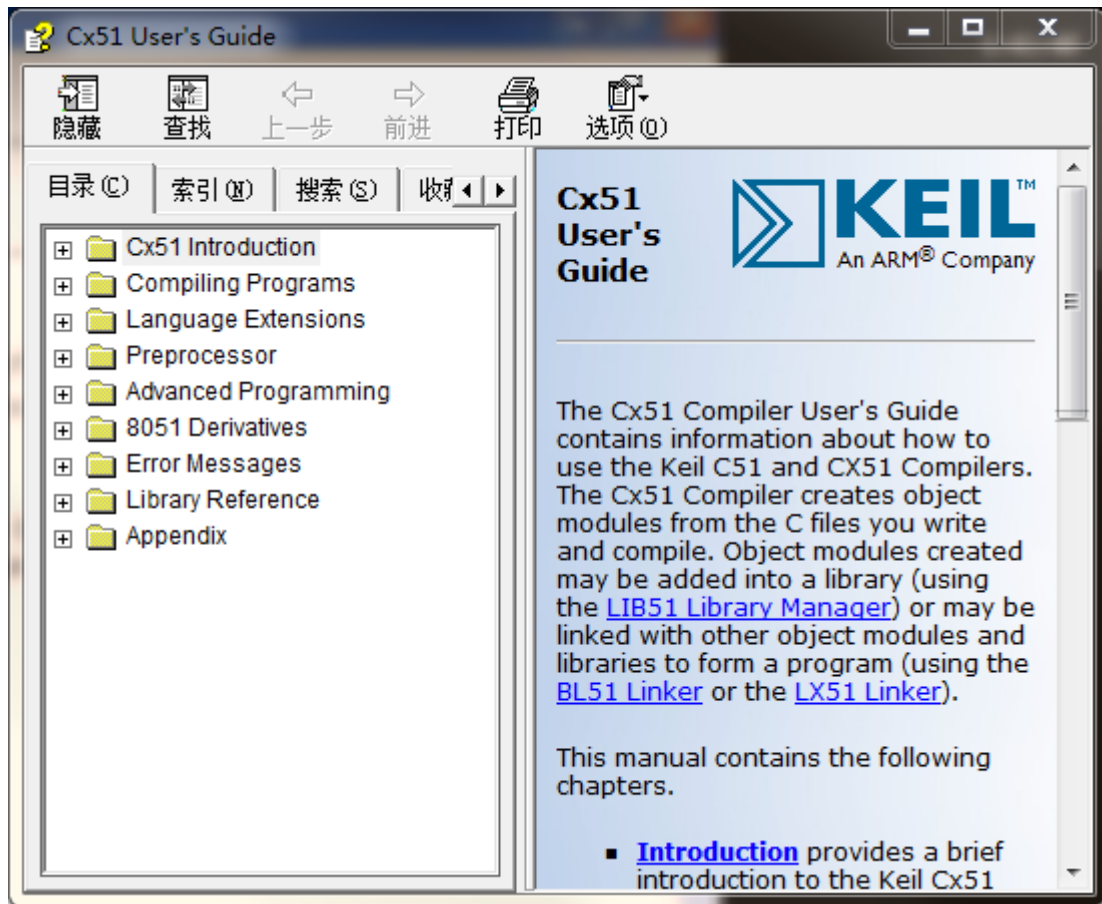
调用子函数之前需要对子函数进行声明，即将子函数原封不动地写一次。



带参函数，也需要声明：



51 是帮助文档 F:\Software\Keil_v5\C51\Hlp。 C51user guide-library reference。
点开“索引”，里面都是 C51 自带的一些函数。



调用 C51 自带的函数 `unsigned char _crol_ (unsigned char c,unsigned char b);` ， 编写流水灯程序，它是循环左移函数：



Project: cycle
Target1
Source Group 1

```

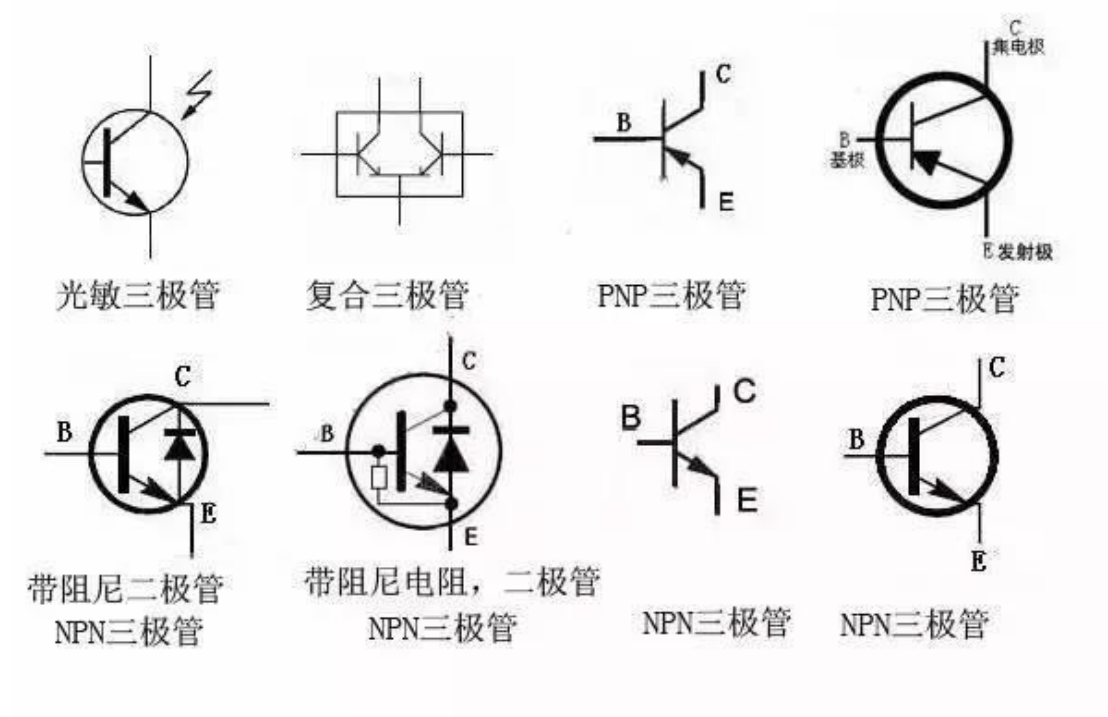
1#include<reg52.h>
2#include <intrins.h>
3#define uint unsigned int
4#define uchar unsigned char
5uchar temp; //设置全局变量给P1口赋值。P1是8位寄存器，定义成8位char型
6sbit D1 = P1^0;
7//void delay();
8void delay(uint); //可以写参数z，也可直接写类型
9void main()
10{
11    temp=0xfe; //点亮第一个灯
12    P1=temp;
13    while(1)
14    {
15        temp=_crol_(temp,1); //循环左移函数
16        delay(600);
17        P1=temp; //点亮第二个灯
18    }
19}
20//带参函数
21void delay(uint z)
22{
23    uint x,y;
24    for(x=100;x>0;x--)
25    {
26        for(y=z;y>0;y--);
27    }
28}

```

二、点亮数码管

三极管是电压控制电流的放大作用。

单片机可能不能直接驱动 LED，因为电流不够，所以我们可以控制三极管的导通或截止，来控制 LED 的亮与灭。



若对单片机 P3 口的内部寄存器进行设置，就能使用引脚的第二功能。

单片机冷启动：先电脑点击下载按钮，串口给单片机发送下载握手信息，单片机只有在启动的时候才会检测是否需要下载信息。

P3.2，P3.2 外部中断器 0,1.

P3.4,P3.5 定时器/计数器。

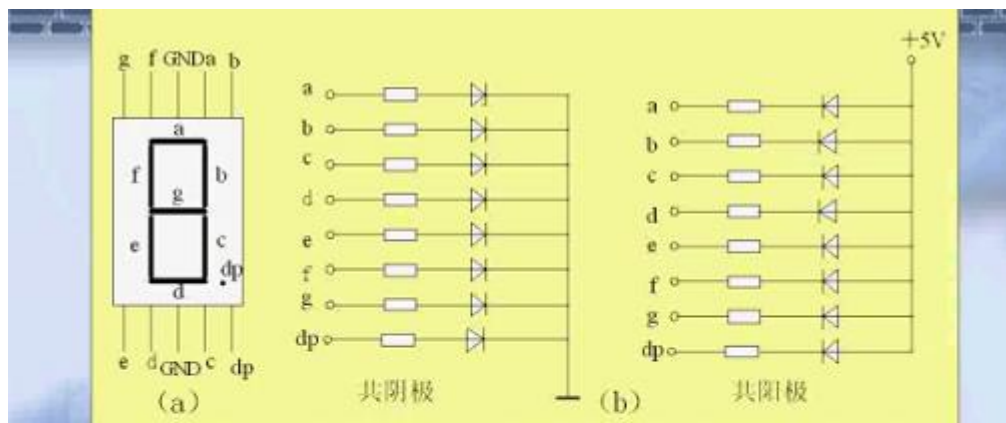
正弦波通过比较器可以变成方波，三角波通过积分可以变成方波。可计算方波的输入个数。

RST 电路的放电时间 $\tau = \sqrt{RC}$ 。

每个寄存器有一个地址。

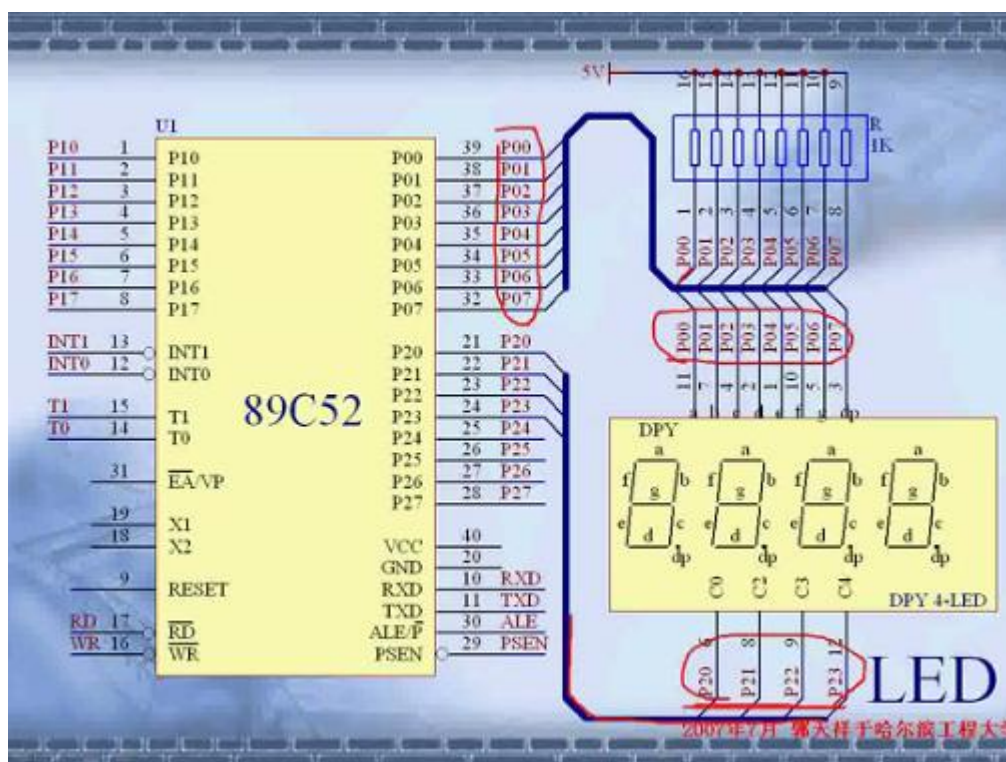
P0 口内部无上拉电阻，是三态状态，无法进行电平的拉高拉低操作，所以需要接上拉电阻。

上电后就是高电平。P0 口需接上拉电阻，就是把电平拉高，以提高驱动能力。



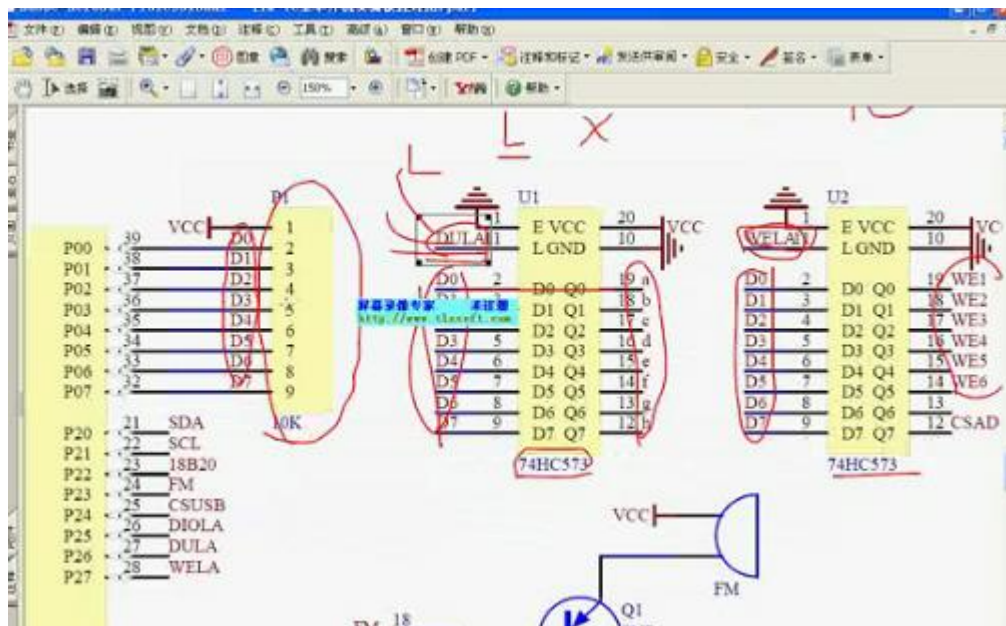
使用LED显示器时，要注意区分这两种不同的接法。为了显示数字或字符，必须对数字或字符进行编码。七段数码管加上一个小数点，共计8段。因此为LED显示器提供的编码正好是一个字节。TX实验板用共阴LED显示器，根据电路连接图显示16进制数的编码已列在下表。

2007年7月 鄧天祥于哈尔滨工程大学



2007年7月 鄧天祥于哈尔滨工程大学

发光二极管 LED (light emitting diode)，液晶 LCD (liquid crystal display)。LED、LCD 显示器有两种显示结构：段显示（7 段、米字型等），点阵显示（5*8、8*8 点阵等）。



多个数码管位选+段选。通过位选控制哪个数码管亮，通过段选控制数码管显示什么东西。锁存器 74HC573 的锁存端为高电平时，它的输入和输出是直通的；为低电平时，输入与输出断开，输出保持原来的值。一个锁存器控制数码管的位选，一个锁存器控制数码管的段选。AD 芯片片选 CS，低电平有效。

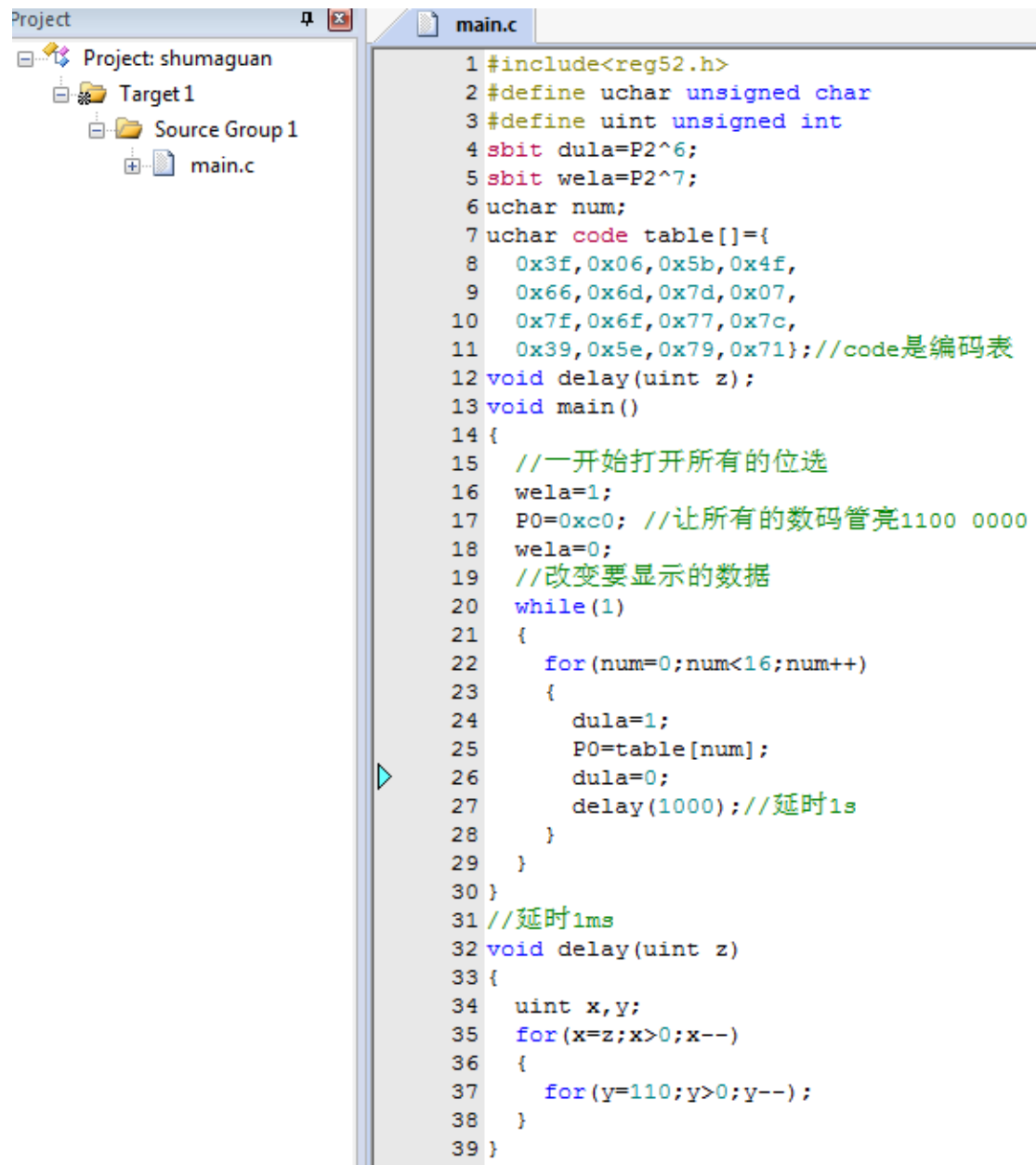
编码表：

```
uchar code table[]={};
```

code 编码完后会放在文件存储区中，如果不用则是放在随机存储区中。单片机的随机存储区（RAM）是 128 个字节（char 型），是有限的。

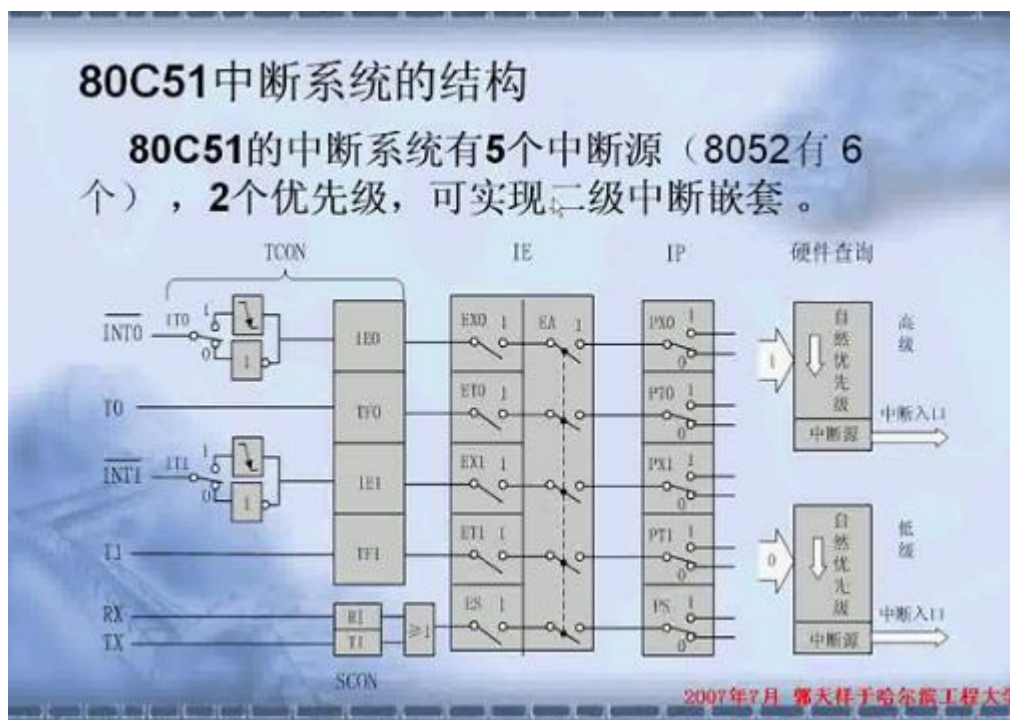
数码管的静态显示：

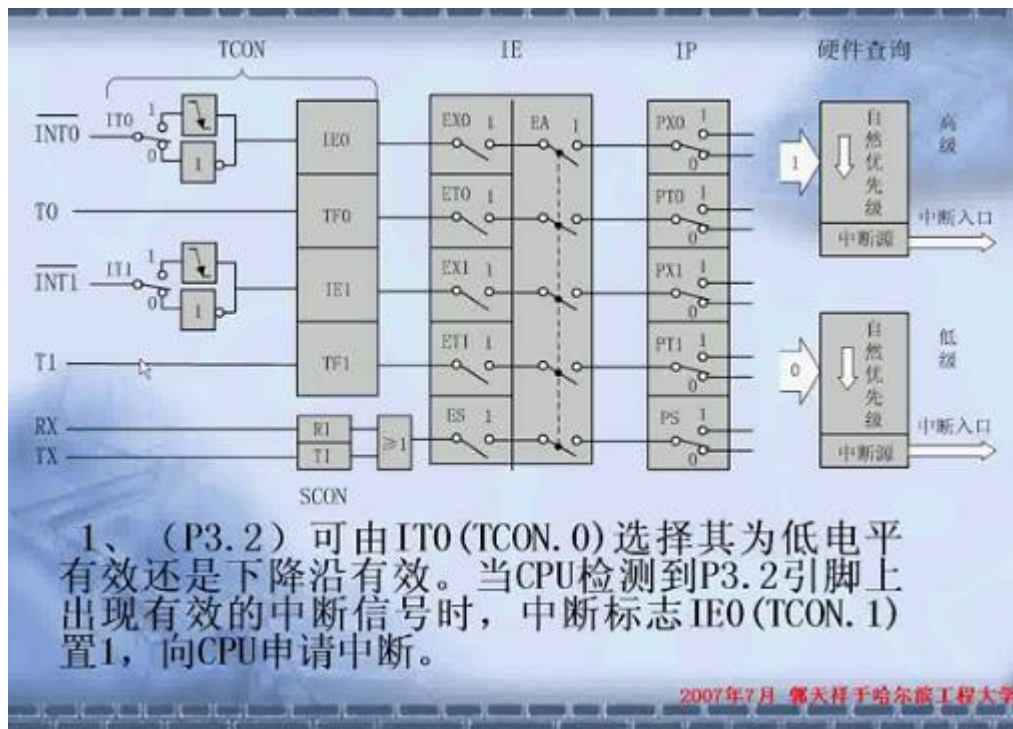
对所有的数码管操作，所有数码管显示相同的数，间隔 1s 显示下一个数：



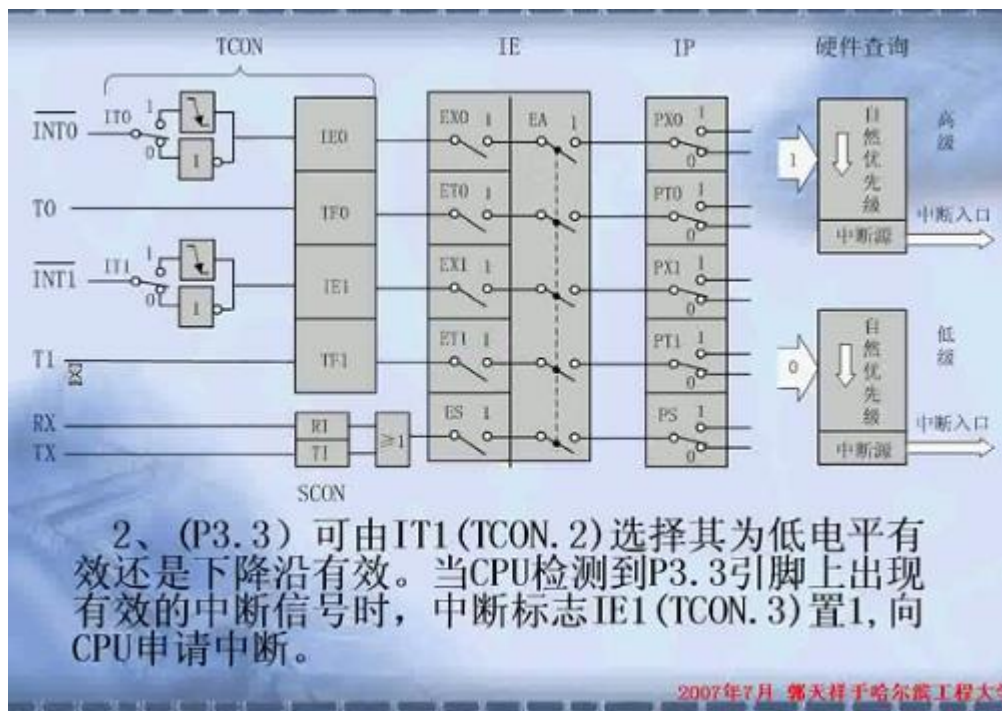
```
1 #include<reg52.h>
2 #define uchar unsigned char
3 #define uint unsigned int
4 sbit dula=P2^6;
5 sbit wela=P2^7;
6 uchar num;
7 uchar code table[]={
8     0x3f,0x06,0x5b,0x4f,
9     0x66,0x6d,0x7d,0x07,
10    0x7f,0x6f,0x77,0x7c,
11    0x39,0x5e,0x79,0x71}; //code是编码表
12 void delay(uint z);
13 void main()
14 {
15     //一开始打开所有的位选
16     wela=1;
17     P0=0xc0; //让所有的数码管亮1100 0000
18     wela=0;
19     //改变要显示的数据
20     while(1)
21     {
22         for(num=0;num<16;num++)
23         {
24             dula=1;
25             P0=table[num];
26             dula=0;
27             delay(1000); //延时1s
28         }
29     }
30 }
31 //延时1ms
32 void delay(uint z)
33 {
34     uint x,y;
35     for(x=z;x>0;x--)
36     {
37         for(y=110;y>0;y--);
38     }
39 }
```

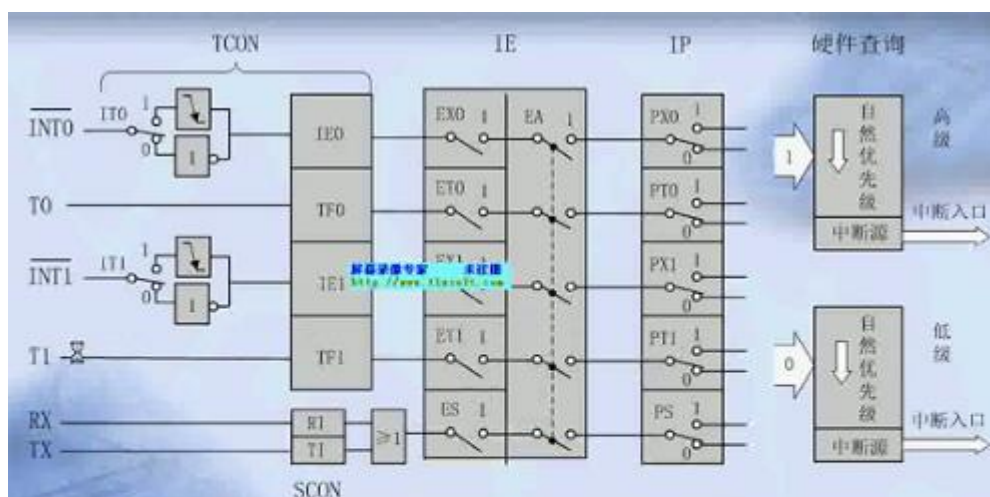

三、中断





通过设置 IT0 可以设置外部中断是低电平触发，还是下降沿跳变触发。





3、**TF0** (TCON.5)，片内定时/计数器T0溢出中断请求标志。当定时/计数器T0发生溢出时，置位TF0，并向CPU申请中断。

2007年7月 魏天祥于哈尔滨工程大学

二、中断请求标志

1、TCON的中断标志

位	7 ⁰	6 ⁰	5 ⁰	4 ⁰	3 ⁰	2 ⁰	1 ⁰	0 ⁰	
字节地址: 88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	TCON

IT0 (TCON.0)，外部中断0触发方式控制位。

当**IT0=0**时，为电平触发方式。

当**IT0=1**时，为边沿触发方式（下降沿有效）。

IE0 (TCON.1)，外部中断0中断请求标志位。

IT1 (TCON.2)，外部中断1触发方式控制位。

IE1 (TCON.3)，外部中断1中断请求标志位。

TF0 (TCON.5)，定时/计数器T0溢出中断请求标志位。

TF1 (TCON.7)，定时/计数器T1溢出中断请求标志位。

2007年7月 魏天祥于哈尔滨工程大学

同一优先级中的中断申请不止一个时，则有中断优先权排队问题。同一优先级的中断优先权排队，由中断系统硬件确定的自然优先级形成，其排列如所示：

各中断源响应优先级及中断服务程序入口表

中断源	中断标志	中断服务程序入口	优先级顺序
外部中断 0 ($\overline{\text{INT0}}$)	IE0	0003H	高
定时/计数器 0 (T0)	TF0	000BH	↓
外部中断 1 ($\overline{\text{INT1}}$)	IE1	0013H	↓
定时/计数器 1 (T1)	TF1	001BH	↓
串行口	RI 或 TI	0023H	低

2007年7月 魏天祥于哈尔滨工程大学

80C51单片机的中断优先级有三条原则：

- CPU同时接收到几个中断时，首先响应优先级别最高的中断请求。
- 正在进行的中断过程不能被新的同级或低优先级的中断请求所中断。
- 正在进行的低优先级中断服务，能被高优先级中断请求所中断。

为了实现上述后两条原则，中断系统内部设有两个用户不能寻址的优先级状态触发器。其中一个置1，表示正在响应高优先级的中断，它将阻断后来所有的中断请求；另一个置1，表示正在响应低优先级中断，它将阻断后来所有的低优先级中断请求。

3.2 80C51单片机中断处理过程

中断响应条件

刘福源专家 非注册
<http://www.ckeyfx.com>

- 中断源有中断请求;
- 此中断源的中断允许位为1;
- CPU开中断（即EA=1）。

以上三条同时满足时，CPU才有可能响应中断。

```
#include<reg52.h>
#define uchar unsigned char
#define uint unsigned int
sbit dula=P2^6;
sbit wela=P2^7;
sbit d1=P1^0;//控制第一个 LED
uchar num;
uchar code table[]={
    0x3f,0x06,0x5b,0x4f,
    0x66,0x6d,0x7d,0x07,
    0x7f,0x6f,0x77,0x7c,
    0x39,0x5e,0x79,0x71};//code 是编码表
void delay(uint z);
void main()
{
    //将外部中断 0 设置为电平触发 IT0=0;
    //单片机打开默认是电平触发
    EA=1; //打开总中断
    EX0=1; //打开外部中断 0
    IT0=1; //将外部中断 0 设置为电平触发
    //TCON=0x01;//直接对寄存器中的某一位进行操作
    //与上面一行的作用是相同的
    wela=1; //一开始打开所有的位选
    P0=0xc0; //让所有的数码管亮 1100 0000
    wela=0;
    //改变要显示的数据
    while(1)
    {
```

```

        for(num=0;num<16;num++)
        {
            dula=1;
            P0=table[num];
            dula=0;
            delay(1000); //延时 1s
        }
    }
}
//延时 1ms
void delay(uint z)
{
    uint x,y;
    for(x=z;x>0;x--)
    {
        for(y=110;y>0;y--);
    }
}
//中断让第一个 LED 亮
//中断函数，中断程序不需要声明，没有返回值
void exter0() interrupt 0
{
    d1=0; //中断中若 P2.2 口为低电平则进入中断
}

```

定时/计数器

3.3 80C51的定时/计数器

实现定时功能，**比较方便的办法是利用单片机内部的定时/计数器**。也可以采用下面三种方法：

- **软件定时**：软件定时不占用硬件资源，但占用了**CPU**时间，降低了**CPU**的利用率。
- **采用时基电路定时**：例如采用**555**电路，外接必要的元器件（电阻和电容），即可构成硬件定时电路。但在硬件连接好以后，定时值与定时范围不能由软件进行控制和修改，即不可编程。
- **采用可编程芯片定时**：这种定时芯片的定时值及定时范围很容易用软件来确定和修改，此种芯片定时功能强，使用灵活。在单片机的定时/计数器不够用时，可以考虑进行扩展。

2007年7月 郭天祥于哈尔滨工程

3.1.3 80C51中断的控制

一、中断允许控制

CPU对中断系统所有中断以及某个中断源的开放和屏蔽是由中断允许寄存器IE控制的。

位	7	6	5	4	3	2	1	0	
字节地址: A8H	EA			ES	ET1	EX1	ET0	EX0	IE

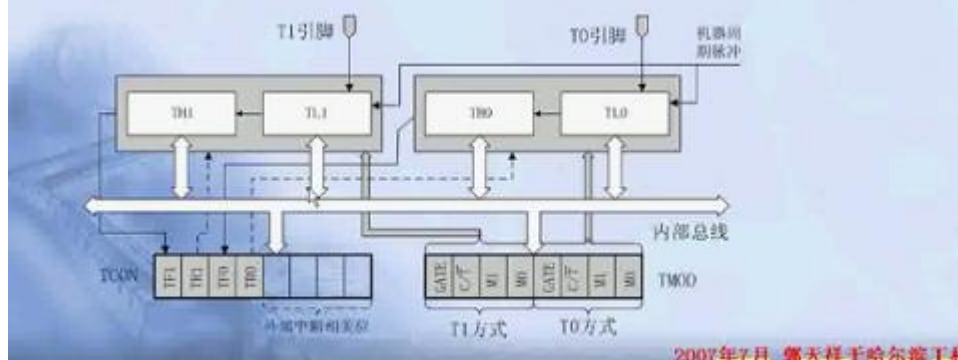
- EX0(IE.0), 外部中断0允许位;
- ET0(IE.1), 定时/计数器T0中断允许位;
- EX1(IE.2), 外部中断1允许位;
- ET1(IE.3), 定时/计数器T1中断允许位;
- ES (IE.4), 串行口中断允许位;
- EA (IE.7), CPU中断允许(总允许)位。

四、定时/计数器

3.3.1 定时/计数器的结构和工作原理

一、定时/计数器的结构

定时/计数器的实质是加1计数器(16位), 由高8位和低8位两个寄存器组成。TMOD是定时/计数器的工作方式寄存器, 确定工作方式和功能; TCON是控制寄存器, 控制T0、T1的启动和停止及设置溢出标志。



2007年7月 第天祥于哈尔滨工大

二、定时/计数器的工作原理

加1计数器输入的计数脉冲有两个来源,一个是由系统的时钟振荡器输出脉冲经12分频后送来;一个是T0或T1引脚输入的外部脉冲源。每来一个脉冲计数器加1,当加到计数器为全1时,再输入一个脉冲就使计数器回零,且计数器的溢出使TCON中TF0或TF1置1,向CPU发出中断请求(定时/计数器中断允许时)。如果定时/计数器工作于定时模式,则表示定时时间已到;如果工作于计数模式,则表示计数值已满。

可见,由溢出时计数器的值减去计数初值才是加1计数器的计数值。

■ 设置为定时器模式时,加1计数器是对内部机器周期计数(1个机器周期等于12个振荡周期,即计数频率为晶振频率的1/12)。计数值N乘以机器周期Tcy就是定时时间t。

$$f_0 \times \left(\frac{1}{12} \times 12 \right) = t$$
$$65535 \times 1 = t \quad \mu s$$

3.3.2 定时/计数器的控制

80C51单片机定时/计数器的工作由两个特殊功能寄存器控制。TMOD用于设置其工作方式;TCON用于控制其启动和中断申请。

一、工作方式寄存器TMOD

工作方式寄存器TMOD用于设置定时/计数器的工作方式,低四位用于T0,高四位用于T1。其格式如下:

位	7	6	5	4	3	2	1	0	
字节地址: 89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	TMOD

GATE: 门控位。**GATE=0**时, 只要用软件使**TCON**中的**TR0**或**TR1**为**1**, 就可以启动定时/计数器工作; **GATE=1**时, 要用软件使**TR0**或**TR1**为**1**, 同时外部中断引脚或也为高电平时, 才能启动定时/计数器工作。即此时定时器的启动多了一条条件。

C/T: 定时/计数模式选择位。**C/T=0**为定时模式; **C/T=1**为计数模式。

M1M0: 工作方式设置位。定时/计数器有四种工作方式, 由**M1M0**进行设置。

定时/计数器工作方式设置表			
M1M0	工作方式	说	明
00	方式 0	13 位定时/计数器	
01	方式 1	16 位定时/计数器	
10	方式 2	8 位自动重装定时/计数器	
11	方式 3	T0 分成两个独立的 8 位定时/计数器; T1 此方式停止计数	

二、控制寄存器 TCON

TCON的低4位用于控制外部中断, 已在前面介绍。**TCON**的高4位用于控制定时/计数器的启动和中断申请。其格式如下:

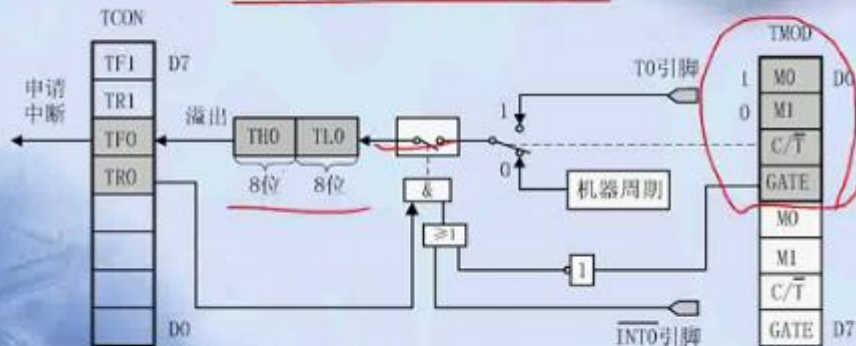
位	7	6	5	4	3	2	1	0	
字节地址: 88H	TF1	TR1	TF0	TR0					TCON

- **TF1 (TCON.7)**: T1溢出中断请求标志位。T1计数溢出时由硬件自动置TF1为1。CPU响应中断后TF1由硬件自动清0。T1工作时, CPU可随时查询TF1的状态。所以, TF1可用作查询测试的标志。TF1也可以用软件置1或清0, 同硬件置1或清0的效果一样。
- **TR1 (TCON.6)**: T1运行控制位。TR1置1时, T1开始工作; TR1置0时, T1停止工作。TR1由软件置1或清0。所以, 用软件可控制定时/计数器的启动与停止。
- **TF0 (TCON.5)**: T0溢出中断请求标志位, 其功能与TF1类同。
- **TR0 (TCON.4)**: T0运行控制位, 其功能与TR1类同。

TCON = 0x88; //高四位控制定时/计数启动和中断申请; 第四位控制外部中断
TF1、TF0: 当计数器计数到溢出, 由单片机自动置 1, 不用管。

二、方式1

方式1的计数位数是16位，由TL0作为低8位，TH0作为高8位，组成了16位加1计数器。



计数个数与计数初值的关系为： $X=2^{16}-N$

开始会装一个初值，用于让计数器计多少，产生一次中断。

若初始值为 5000，计数个数 (65535-50000) 应当装在高八位和低八位中，高八位中装的是求模 $(65535-50000)/256$ ，低八位中装的是求余 $(65535-50000)\%256$ 。

3.3.5 定时/计数器应用举例

初始化程序应完成如下工作：

- 对TMOD赋值，以确定T0和T1的工作方式。
- 计算初值,并将其写入TH0、TL0或TH1、TL1。
- 中断方式时，则对IE赋值，开放中断。
- 使TR0或TR1置位，启动定时/计数器定时或计数。

16 位定时器，只能定时 65ms，若要定时 1 秒，可以让进入 20 次中断，每次中断的时间为 50ms。定义一个变量，每进入一次中断加 1，一旦到 20 次，则让数码管亮。

```
TMOD=0x01; //定时器 T0 工作在方式 1: 01
```

```
TH0=(65536-50000)/256;//计算初值
```

```
TL0=(65536-50000)%256;
```

想装多少初值，就用 65535-这个初值。

```
#include<reg52.h>
```

```
#define uchar unsigned char
```

```
#define uint unsigned int
```

```
sbit dula=P2^6;
```

```
sbit wela=P2^7;
```



```

sbit d1=P1^0;//控制第一个 LED
uchar num,tt;
uchar code table[]={
    0x3f,0x06,0x5b,0x4f,
    0x66,0x6d,0x7d,0x07,
    0x7f,0x6f,0x77,0x7c,
    0x39,0x5e,0x79,0x71};//code 是编码表
void delay(uint z);
void main()
{
    num=0;
    tt=0;
    /**定时器 T0 初始化**/
    TMOD=0x01; //定时器 T0 工作在方式 1: 01
    TH0=(65536-50000)/256;//计算初值 高八位
    TL0=(65536-50000)%256; //低八位
    //将外部中断 0 设置为电平触发 IT0=0;
    //单片机打开默认是电平触发
    EA=1; //打开总中断
    ET0=1;//打开定时器 0 中断
    TR0=1;//启动定时器 0

    wela=1; //一开始打开所有的位选
    P0=0xea; //1110 1010
    wela=0;
    dula=1;
    P0=0x3f; //段选为 0，不然进入循环时会出错，因为没定义段选，是随机的
    //过 1s 选 1
    dula=0;
    //改变要显示的数据
    while(1)
    {
        if(tt==20) //进如 20 次中断
        {
            tt=0;
            num++;
            if(num==16)
                num=0;
            dula=1;
            P0=table[num];
            dula=0;
            delay(1000);//延时 1s
        }
    }
}

```

```

}
//延时 1ms
void delay(uint z)
{
    uint x,y;
    for(x=z;x>0;x--)
    {
        for(y=110;y>0;y--);
    }
}

//定时器中断 0 的服务程序
//进入中断后要重新装一次初值
void exter0() interrupt 1
{
    TH0=(65536-50000)/256;//计算初值 高八位
    TL0=(65536-50000)%256; //低八位
    tt++;
}

```

设计流水灯 1s 循环闪烁，数码管每 2s 按数值循环显示。

```

#include<reg52.h>
#include<intrins.h>
#define uint unsigned int
#define uchar unsigned char
uchar temp,aa,num;
sbit dula=P2^6;
sbit wela=P2^7;
uchar code table[]={
    0x3f,0x06,0x5b,0x4f,
    0x66,0x6d,0x7d,0x07,
    0x7f,0x6f,0x77,0x7c,
    0x39,0x5e,0x79,0x71}; //数码管 code
void delay(uint z);
void main()
{
    num=0;
    aa=0;
    /**定时器 T0 初始化**/
    TMOD=0x01; //定时器 T0 工作在方式 1: 01
    TH0=(65536-50000)/256; //以 2s 为一次循环，50ms 产生一次中断
    TL0=(65536-50000)%256;
    EA=1; //开总中断
    ET0=1; //开定时器 0 中断
}

```

```

TR0=1;//启动定时器

/**流水灯程序**/
//点亮第一个 LED
temp=0xfe;
P1=temp;
/**刚开始时，需要让数码管亮 0**/
dula=1;
P0=table[0];
dula=0;
wela=1;
P0=0xc0; //所有数码管全部打开 1100 0000
wela=0;
while(1)
{
    delay(1000);
    temp=_crol_(temp,1);//循环左移
    P1=temp;
}
}
//延时函数 1ms
void delay(uint z)
{
    uint x,y;
    for(x=z;x>0;x--)
    {
        for(y=110;y>0;y--);
    }
}
/**定义定时器 0 中断**/
void timer0() interrupt 1
{
    TH0=(65536-50000)/256;//以 2s 为一次循环，50ms 产生一次中断
    TL0=(65536-50000)%256;
    aa++;
    /**检测是否到了 2s**/
    if(aa==40) // 2s/50ms.aa 为进入中断的次数，如果进入中断 40 次，即 2s，进行操作
    {
        //aa==40 说明 2s 到了，则
        aa=0; //aa 清零，则数码管的值改变一次，需要位选和段选
        num++; //num 每加一次，需要判断 num 是否到了 16，到了 16 则清零
        if(num==16)
        {
            num=0;

```

```

    }
    //让数码管显示下一个数值
    dula=1;
    P0=table[num];
    dula=0;
}
}

```

让 6 个数码管每一个显示一个数，显示 1 到 6:

```

#include<reg52.h>
#define uint unsigned int
#define uchar unsigned char
uchar temp,aa,numdu,numwe;
sbit dula=P2^6;
sbit wela=P2^7;
uchar code table[]={
    0x3f,0x06,0x5b,0x4f,
    0x66,0x6d,0x7d,0x07,
    0x7f,0x6f,0x77,0x7c,
    0x39,0x5e,0x79,0x71};//数码管段选 code
uchar code tablewe[]={
    0xfe,0xfd,0xfb,0xf7,
    0xef,0xdf};//数码管位选 code,一共 6 个数码管

void main()
{
    numdu=0;
    numwe=0;
    aa=0;
    /**定时器 T0 初始化**/
    TMOD=0x01; //定时器 T0 工作在方式 1: 01
    TH0=(65536-50000)/256;//以 2s 为一次循环，50ms 产生一次中断
    TL0=(65536-50000)%256;
    EA=1;//开总中断
    ET0=1;//开定时器 0 中断
    TR0=1;//启动定时器

    /**刚开始时，需要让所有数码管不亮**/

    while(1)
    {
        if(aa==20)
        {
            aa=0;

```



```

        //送段码
        numdu++;
        if(numdu==7) //只显示 1 到 6 之间的变化
            numdu=1;
        dula=1;
        P0=table[numdu];
        dula=0;
        //送位码
        wela=1;
        P0=tablewe[numwe];
        wela=0;
        numwe++;
        if(numwe==6) //只用 6 个数码管，限定不超过 6
            numwe=0;
    }
}

/**定义定时器 0 中断**/
void timer0() interrupt 1
{
    TH0=(65536-50000)/256;//以 2s 为一次循环，50ms 产生一次中断
    TL0=(65536-50000)%256;
    aa++;
    //主函数中没有别的程序，则不需要把对它的操作放在这个中断程序里面
}

```

让数码管显示 12，用简单的延时函数：

```

#include<reg52.h>
#include<intrins.h>
#define uint unsigned int
#define uchar unsigned char
uchar temp,numdu,shi,ge;
sbit dula=P2^6;
sbit wela=P2^7;
uchar code table[]={
    0x3f,0x06,0x5b,0x4f,
    0x66,0x6d,0x7d,0x07,
    0x7f,0x6f,0x77,0x7c,
    0x39,0x5e,0x79,0x71};//数码管 code
void delay(uint z);
void main()
{

```

```

/**定时器 T0 初始化**/
TMOD=0x01; //定时器 T0 工作在方式 1: 01
TH0=(65536-50000)/256; //以 2s 为一次循环, 50ms 产生一次中断
TL0=(65536-50000)%256;
EA=1; //开总中断
ET0=1; //开定时器 0 中断
TR0=1; //启动定时器

temp=12;
numdu=0;
while(1)
{
    shi=temp/10;
    ge=temp%10;
    dula=1;
    P0=table[shi];
    dula=0;
    wela=1;
    P0=0xfe;
    wela=0;
    delay(5);

    dula=1;
    P0=table[ge];
    dula=0;
    wela=1;
    P0=0xfd;
    wela=0;
    delay(5);
}
}
//延时函数 1ms
void delay(uint z)
{
    uint x,y;
    for(x=z;x>0;x--)
    {
        for(y=110;y>0;y--);
    }
}

```

五、键盘

键盘的工作方式通常有两种：

（1）扫描法，不断扫描键盘的状态，送 CPU 判断并处理。如果键盘数目一大的话，显然不适合。（2）线反转法，通过行列状态的改变来判断有无键被按下。