第 5 章

定时器与数码管基础


第 5 章视频

通过前面的讲解，大家会发现，自己逐渐进入比较实质性的学习了，需要记住的内容也更多了，个别地方可能会感觉吃力。但是大家不要担心，要有信心。这个跟小孩学走路一样，刚开始走得不太稳，没关系，多走几步多练练。看教材的时候要注意专心，一遍看不懂，思考一下，再回头看第二遍和第三遍，没准一下就明白了。如果三遍还看不明白，那就把不懂的问题放一放，继续往下学，然后再回头看一次，也可以到群里或者论坛里多咨询一下其他的同学，讨论一下，可能就会茅塞顿开了。

5.1 逻辑电路与逻辑运算

在数字电路中经常会遇到逻辑电路，而在 C 语言中则经常用到逻辑运算。二者在原理上是相互关联的，在这里就先简单介绍一下，随着学习的深入，再慢慢加深理解。

首先，在“逻辑”这个概念范畴内，存在真和假这两个逻辑值，而将其对应到数字电路或 C 语言中，就变成了“非 0 值”和“0 值”这两个值，即逻辑上的“假”就是数字电路或 C 语言中的“0”这个值，而逻辑“真”就是其他一切“非 0 值”。

然后，来具体分析一下几个主要的逻辑运算符。假定有两个字节变量：A 和 B，二者进行某种逻辑运算后的结果为 F。

以下逻辑运算符都是按照变量整体值进行运算的，通常就叫作逻辑运算符。

- && 逻辑与。F = A && B，当 A、B 的值都为真（即非 0 值，下同）时，其运算结果 F 为真（具体数值为 1，下同）；当 A、B 值任意一个为假（即 0，下同）时，结果 F 为假（具体数值为 0，下同）。
- || 逻辑或。F = A || B，当 A、B 值任意一个为真时，其运算结果 F 为真；当 A、B 值都为假时，结果 F 为假。
- ! 逻辑非，F = !A，当 A 值为假时，其运算结果 F 为真；当 A 值为真时，结果 F 为假。

以下逻辑运算符都是按照变量内的每一个位来进行运算的，通常就叫作位运算符：

- & 按位与，F = A & B，将 A、B 两个字节中的每一位都进行与运算，再将得到的每

一位结果组合为总结果 F，例如 A = 0b11001100，B = 0b11110000，则结果 F 就等于 0b11000000。

- | 按位或，F = A | B，将 A、B 两个字节中的每一位都进行或运算，再将得到的每一位结果组合为总结果 F，例如 A = 0b11001100，B = 0b11110000，则结果 F 就等于 0b11111100。

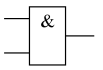
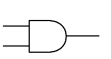
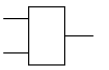
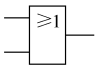
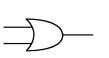
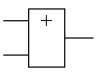
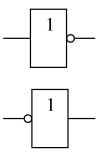
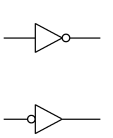
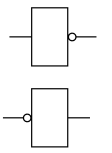
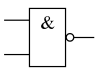
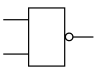
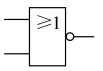
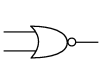
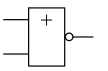
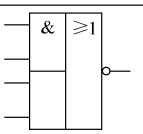
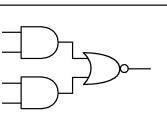
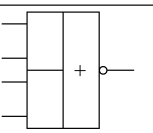
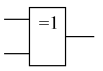
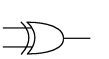
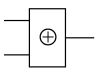
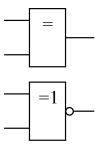
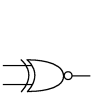
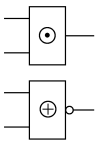
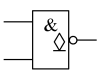
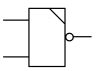
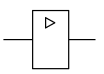
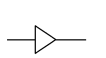
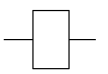
序号	名　称	GB/T 4728.12—1996		国外流行图形符号	曾用图形符号
		限定符号	国标图形符号		
1	与门	&			
2	或门	≥1			
3	非门	逻辑非入和出			
4	与非门				
5	或非门				
6	与或非门				
7	异或门	=1			
8	同或门	=			
9	集电极开路OC门、漏极开路OD门	L形开路输出			
10	缓冲器	▷			

图 5-1　逻辑电路符号

- ～按位取反，F = ～A，将 A 字节内的每一位进行非运算（就是取反），再将得到的每一位结果组合为总结果 F，例如 A = 0b11001100，则结果 F 就等于 0b00110011；这个运算符在前面的流水灯实验里已经用过了，现在再回头看一眼，是不是清楚多了。
- ^ 按位异或，异或的意思是，如果运算双方的值不同（即相异）则结果为真，双方值相同则结果为假。在 C 语言里没有按变量整体值进行的异或运算，所以仅以按位异或为例，F＝A^B，A＝0b11001100，B＝0b11110000，则结果 F 就等于 0b00111100。

今后要看资料或芯片手册的时候，会经常遇到一些电路符号，图 5-1 所示就是数字电路中的常用符号，知道这些符号有利于大家理解器件的逻辑结构，尤其重点认识图 5-1 中的国外流行图形符号。在这里我们先简单看一下，如果日后遇到了可以到这里来查阅。

5.2　定时器的学习

定时器是单片机系统的一个重点，但并不是难点，大家一定要完全理解并且熟练掌握定时器的应用。

5.2.1　定时器的初步认识

1. 时钟周期

时钟周期 T 是时序中最小的时间单位，具体计算的方法就是

$$\text{时钟周期} = \frac{1}{\text{时钟源频率}}$$

KST-51 单片机开发板上用的晶振是 11.0592MHz，那么对于这个单片机系统来说，时钟周期＝1/11059200 秒。

2. 机器周期

单片机完成一个操作的最短时间。机器周期主要针对汇编语言而言，在汇编语言下程序的每一条语句执行所使用的时间都是机器周期的整数倍，而且语句占用的时间是可以计算出来的，而 C 语言一条语句的时间是不确定的，受到诸多因素的影响。51 单片机系列，在其标准架构下一个机器周期是 12 个时钟周期，也就是 12/11059200 秒。现在有不少增强型的 51 单片机，其速度都比较快，有的 1 个机器周期等于 4 个时钟周期，有的 1 个机器周期就等于 1 个时钟周期，也就是说大体上其速度可以达到标准 51 架构的 3 倍或 12 倍。因为是讲标准的 51 单片机，所以后边的内容如果遇到这个概念，全部是指 12 个时钟周期。

这两个概念了解即可，下面来讲讲重头戏，定时器和计数器。定时器和计数器是单片机内部的同一个模块，通过配置 SFR（特殊功能寄存器）可以实现两种不同的功能，大多数情况下是使用定时器功能，因此主要来讲定时器功能，计数器功能读者自己了解一下即可。

顾名思义，定时器就是用来进行定时的。定时器内部有一个寄存器，让它开始计数后，这个寄存器的值每经过一个机器周期就会自动加 1，因此，可以把机器周期理解为定时器的

计数周期。就像钟表,每经过一秒,数字自动加 1,而这个定时器就是每过一个机器周期的时间,也就是 12/11059200 秒,数字自动加 1。还有一个特别注意的地方,就是钟表是加到 60 后,秒就自动变成 0 了,这种情况在单片机或计算机里称为溢出。那定时器加到多少才会溢出呢?后面会讲到定时器有多种工作模式,分别使用不同的位宽(指使用多少个二进制位),假如是 16 位的定时器,也就是两个字节,最大值就是 65535,那么加到 65535 后,再加 1 就算溢出,如果有其他位数,道理是一样的,对于 51 单片机来说,溢出后,这个值会直接变成 0。从某一个初始值开始,经过确定的时间后溢出,这个过程就是定时的含义。

5.2.2 定时器的寄存器

标准的 51 单片机内部有 T0 和 T1 这两个定时器,T 就是 Timer 的缩写,现在很多 51 系列单片机还会增加额外的定时器,在这里先讲定时器 0 和 1。前边提到过,对于单片机的每一个功能模块,都是由它的 SFR,也就是特殊功能寄存器来控制。与定时器有关的特殊功能寄存器,有以下几个,大家不需要去记忆这些寄存器的名字和作用,只要大概知道就行,用的时候,随时可以查手册,找到每个寄存器的名字和每个寄存器所起到的作用。

表 5-1 的寄存器是存储定时器的计数值的。TH0/TL0 用于 T0,TH1/TL1 用于 T1。

表 5-1 定时值存储寄存器

名称	描 述	SFR 地址	复位值
TH0	定时器 0 高字节	0x8C	0x00
TL0	定时器 0 低字节	0x8A	0x00
TH1	定时器 1 高字节	0x8D	0x00
TL1	定时器 1 低字节	0x8B	0x00

表 5-2 是定时器控制寄存器 TCON 的位分配,表 5-3 则是对每一位的具体含义的描述。

表 5-2 TCON——定时器控制寄存器的位分配(地址 0x88、可位寻址)

位	7	6	5	4	3	2	1	0
符号	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
复位值	0	0	0	0	0	0	0	0

表 5-3 TCON——定时器控制寄存器的位描述

位	符号	描 述
7	TF1	定时器 1 溢出标志。一旦定时器 1 发生溢出时硬件置 1。清零有两种方式:软件清零或者进入定时器中断时硬件清零
6	TR1	定时器 1 运行控制位。软件置位/清零来进行启动/停止定时器
5	TF0	定时器 0 溢出标志。一旦定时器 0 发生溢出时硬件置 1。清零有两种方式:软件清零或者进入定时器中断时硬件清零

续表

位	符号	描　　述
4	TR0	定时器 0 运行控制位。软件置位/清零来进行启动/停止定时器
3	IE1	外部中断部分，与定时器无关，暂且不看
2	IT1	
1	IE0	
0	IT0	

大家注意在表 5-3 中的描述中，只要写到硬件置 1 或者清 0 的，就是指一旦符合条件，单片机将自动完成动作，只要写软件置 1 或者清 0 的，就是指必须用程序去完成这个动作，后续遇到此类描述就不再另做说明了。

对于 TCON 这个 SFR，其中有 TF1、TR1、TF0、TR0 这 4 位需要理解清楚，它们分别对应于 T1 和 T0，以定时器 1 为例讲解，那么定时器 0 同理。先看 TR1，当程序中写 TR1 = 1 以后，定时器值就会每经过一个机器周期自动加 1，当程序中写 TR1 = 0 以后，定时器就会停止加 1，其值会保持不变化。TF1 是一个标志位，它的作用是告诉大家定时器溢出了。比如定时器设置成 16 位的模式，那么每经过一个机器周期，TL1 加 1 一次，当 TL1 加到 255 后，再加 1，TL1 变成 0，TH1 会加 1 一次，如此一直加到 TH1 和 TL1 都是 255（即 TH1 和 TL1 组成的 16 位整型数为 65535）以后，再加 1 一次，就会溢出了，TH1 和 TL1 同时都变为 0，只要一溢出，TF1 马上自动变成 1，告诉大家定时器溢出了，仅仅是提供给读者一个信号，让读者知道定时器溢出了，它不会对定时器是否继续运行产生任何影响。

本节开头就提到了定时器有多种工作模式，工作模式的选择就由 TMOD 来控制，TMOD 的位分配和描述如表 5-4 和表 5-5 所示，TMOD 的位功能如表 5-6 所示。

表 5-4　TMOD——定时器模式寄存器的位分配（地址 0x89、不可位寻址）

位	7	6	5	4	3	2	1	0
符号	GATE (T1)	C/T (T1)	M1 (T1)	M0 (T1)	GATE (T0)	C/T (T0)	M1 (T0)	M0 (T0)
复位值	0	0	0	0	0	0	0	0

表 5-5　TMOD——定时器模式寄存器的位描述

符号	描　　述
T1/T0	在表 5-4 中，标 T1 的表示控制定时器 1 的位，标 T0 的表示控制定时器 0 的位
GATE	该位被置 1 时为门控位。仅当 INTx 引脚为高并且 TRx 控制位被置 1 时使能定时器 x，定时器开始计时，当该位被清零时，只要 TRx 位被置 1，定时器 x 就使能开始计时，不受到单片机引脚 INTx 外部信号的干扰，常用来测量外部信号脉冲宽度。这是定时器一个额外功能，暂不介绍
C/T	定时器或计数器选择位。该位被清零时用作定时器功能（内部系统时钟），被置 1 用作计数器功能

表 5-6 TMOD——定时器模式寄存器 M1/M0 工作模式

M1	M0	工作模式	描　　述
0	0	0	兼容 8048 单片机的 13 位定时器，THn 的 8 位和 TLn 的 5 位组成一个 13 位定时器
0	1	1	THn 和 TLn 组成一个 16 位的定时器
1	0	2	8 位自动重装模式，定时器溢出后 THn 重装到 TLn 中
1	1	3	禁用定时器 1，定时器 0 变成两个 8 位定时器

可能读者已经注意到了，表 5-2 的 TCON 最后标注了“可位寻址”，而表 5-4 的 TMOD 标注的是“不可位寻址”。意思就是说：比如 TCON 有一个位叫 TR1，可以在程序中直接进行 TR1 = 1 这样的操作。但对 TMOD 里的位比如(T1)M1 = 1 这样的操作就是错误的。要操作就必须一次操作这整个字节，也就是必须一次性对 TMOD 所有位操作，不能对其中某一位单独进行操作，那么能不能只修改其中的一位而不影响其他位的值呢？当然可以，在后续内容中读者就会学到方法的，现在就先不关心它了。

表 5-6 列出的就是定时器的 4 种工作模式，其中模式 0 是为了兼容老的 8048 系列单片机而设计的，现在的 51 几乎不会用到这种模式，而模式 3 根据应用经验，它的功能用模式 2 完全可以取代，所以基本上也是不用的，那么就重点来学习模式 1 和模式 2。

模式 1，是 THn 和 TLn 组成了一个 16 位的定时器，计数范围是 0～65535，溢出后，只要不对 THn 和 TLn 重新赋值，则从 0 开始计数。模式 2，是 8 位自动重装载模式，只有 TLn 做加 1 计数，计数范围 0～255，THn 的值不会变化，而会保持原来的值；TLn 溢出后，TFn 就直接置 1 了，并且 THn 原先的值直接赋给 TLn，然后 TLn 从新赋值的这个数字开始计数。这个功能可以用来产生串口的通信波特率，讲串口的时候要用到，本节我们重点来学习模式 1。为了加深大家理解定时器的原理，先来看一下模式 1 的电路示意，如图 5-2 所示。

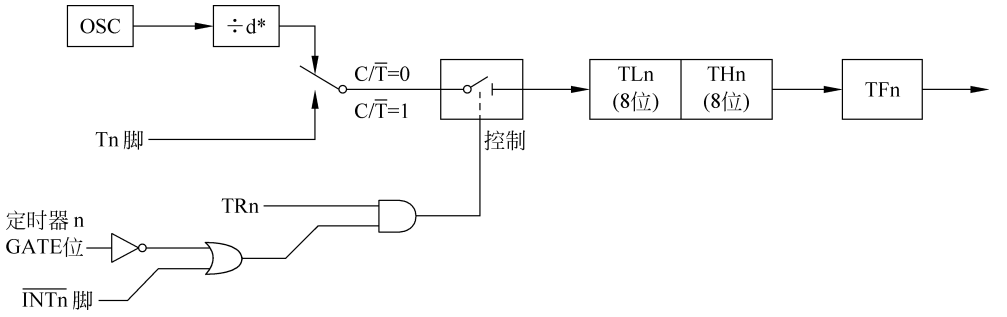


图 5-2　定时器/计数器模式 1 示意图

大家一起来分析一下这个示意图，日后如果再遇到类似的图，就可以自己研究了。OSC 框表示时钟频率，因为 1 个机器周期等于 12 个时钟周期，所以那个 d 就等于 12。下边 GATE 右边的那个门是一个非门电路，再右侧是一个或门，再往右是一个与门电路，大家可以对照一下 5.1 节的内容。

图上可以看出来，下边部分电路是控制了上边部分，那先来看下边是如何控制的，以定

时器 0 为例。

(1) TR0 和下边或门电路的结果要进行与运算，TR0 如果是 0 的话，与运算完了肯定是 0，所以如果要让定时器工作，那么 TR0 就必须置 1。

(2) 这里的与门结果要想得到 1，那么前面的或门出来的结果必须也得是 1 才行。在 GATE 位为 1 的情况下，经过一个非门变成 0，或门电路结果要想是 1，那 INT0 即 P3.2 引脚必须是 1 的情况下，这个时候定时器才会工作，而 INT0 引脚是 0 的情况下，定时器不工作，这就是 GATE 位的作用。

(3) 当 GATE 位为 0 的时候，经过一个非门会变成 1，那么不管 INT0 引脚是什么电平，经过或门电路后都肯定是 1，定时器就会工作。

(4) 要想让定时器工作，就是自动加 1，从图上看有两种方式，第一种方式是那个开关打到上边的箭头，就是 C/T＝0 的时候，一个机器周期 TL 就会加 1 一次，当开关打到下边的箭头，即 C/T ＝1 的时候，T0 引脚即 P3.4 引脚来一个脉冲，TL 就加 1 一次，这也就是计数器功能。

5.2.3　定时器的应用

了解了定时器相关的寄存器，下面就来做一个定时器的程序，巩固一下前面学到的内容。下面的程序先使用定时器 0，在使用定时器的时候，需要以下几个步骤：

第 1 步：设置特殊功能寄存器 TMOD，配置好工作模式。

第 2 步：设置计数寄存器 TH0 和 TL0 的初值。

第 3 步：设置 TCON，通过 TR0 置 1 来让定时器开始计数。

第 4 步：判断 TCON 寄存器的 TF0 位，监测定时器溢出情况。

写程序之前，要先学会计算如何用定时器定时。晶振是 11.0592MHz，时钟周期就是 1/11059200，机器周期是 12/11059200，假如要定时 20ms，就是 0.02 秒，要经过 x 个机器周期得到 0.02 秒，下面来算一下 x×12/11059200＝0.02，得到 x＝ 18432。16 位定时器的溢出值是 65536(因 65535 再加 1 才是溢出)，于是就可以这样操作，先给 TH0 和 TL0 一个初始值，让它们经过 18432 个机器周期后刚好达到 65536，也就是溢出，溢出后可以通过检测 TF0 的值得知，就刚好是 0.02 秒。那么初值 y＝65536－18432＝47104，转成十六进制就是 0xB800，也就是 TH0＝0xB8，TL0＝0x00。

这样 0.02 秒的定时就做出来了，细心的读者会发现，如果初值直接给一个 0x0000，一直到 65536 溢出，定时器定时值最大也就是 71ms 左右，那么想定时更长时间怎么办呢？用小学学过的逻辑，倍数关系就可以解决此问题。

好了，下面就用程序来实现这个功能，代码如下：

```
#include <reg52.h>

sbit LED = P0^0;
sbit ADDR0 = P1^0;
```

```
sbit ADDR1 = P1^1;
sbit ADDR2 = P1^2;
sbit ADDR3 = P1^3;
sbit ENLED = P1^4;

void main()
{
    unsigned char cnt = 0;      //定义一个计数变量,记录 T0 溢出次数

    ENLED = 0;                  //使能 U3,选择独立 LED
    ADDR3 = 1;
    ADDR2 = 1;
    ADDR1 = 1;
    ADDR0 = 0;
    TMOD = 0x01;                //设置 T0 为模式 1
    TH0  = 0xB8;                //为 T0 赋初值 0xB800
    TL0  = 0x00;
    TR0  = 1;                   //启动 T0

    while (1)
    {
        if (TF0 == 1)           //判断 T0 是否溢出
        {
            TF0 = 0;            //T0 溢出后,清 0 中断标志
            TH0 = 0xB8;         //并重新赋初值
            TL0 = 0x00;
            cnt++;              //计数值自加 1
            if (cnt >= 50)      //判断 T0 溢出是否达到 50 次
            {
                cnt = 0;        //达到 50 次后计数值清 0
                LED = ~LED;     //LED 取反: 0-->1、1-->0
            }
        }
    }
}
```

程序中都写了注释,结合前几章学的内容,自己分析一下,不难理解。本程序实现的结果是开发板上最右边的小灯点亮一秒,熄灭一秒,也就是以 0.5Hz 的频率进行闪烁。

5.3 数码管的学习

LED 小灯是一种简单的 LED,只能通过亮和灭来表达简单的信息。而本节要来学习一种能表达更复杂信息的器件——LED 数码管。

5.3.1 数码管的基本介绍

数码管的原理图如图 5-3 所示。

这是比较常见的数码管的原理图，板子上一共有 6 个数码管。前边有了 LED 小灯的学习，数码管学习就会轻松得多了。从图 5-3 可以看出来，数码管共有 a、b、c、d、e、f、g、dp 这么 8 个段，而实际上，这 8 个段每一段都是一个 LED 小灯，所以一个数码管就是由 8 个 LED 小灯组成的。数码管内部结构的示意图如图 5-4 所示。

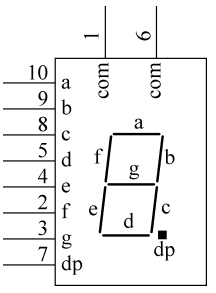


图 5-3　数码管原理图

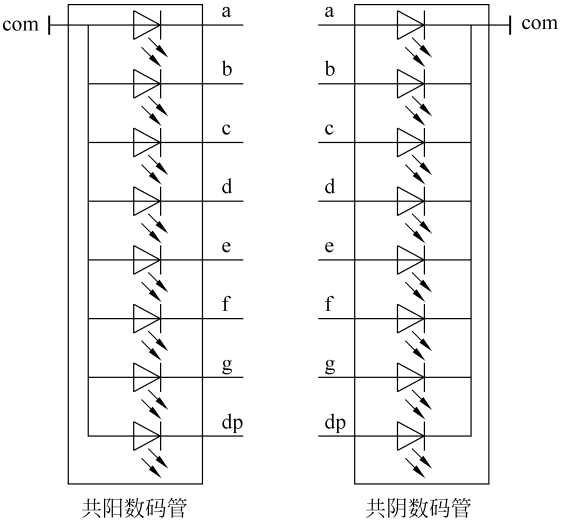


图 5-4　数码管内容结构示意图

数码管分为共阳和共阴两种，共阴数码管就是 8 只 LED 小灯的阴极是连接在一起的，阴极是公共端，由阳极来控制单个小灯的亮灭。同理，共阳数码管就是阳极接在一起，大家可以认真研究下图 5-4。细心的读者会发现，图 5-3 的数码管上边有 2 个 com，这就是数码管的公共端。为什么有 2 个呢，一方面是 2 个可以起到对称的效果，刚好是 10 个引脚，另外一个方面，公共端通过的电流较大，读者初中就学过，并联电路电流之和等于总电流，用 2 个 com 可以把公共电流平均到 2 个引脚上去，降低单条线路承受的电流。

从开发板的电路图上能看出来，所用的数码管都是共阳数码管，一共有 6 个，如图 5-5 所示。

6 个数码管的 com 都是接到了正极上，当然了，和 LED 小灯电路一样，也是由 74HC138 控制三极管的导通来控制整个数码管的使能。先来看最右边的 DS1 这个数码管，原理图上可以看出，控制 DS1 的三极管是 Q17，控制 Q17 的引脚是 LEDS0，对应到 74HC138 上边就是 U3 的 Y0 输出，如图 5-6 所示。

现在的目的是让 LEDS0 这个引脚输出低电平，相信大家现在可以根据前边学过的知识独立把 ADDR0、ADDR1、ADDR2、ADDR3、ENLED 这 4 个所需输入的值写出来了，现在大家不要偷懒，根据 74HC138 的手册去写一下，不需要你记住这些结论，但是遇到就写一次，锻炼过几次后，遇到同类芯片自己就知道如何去解决问题了。

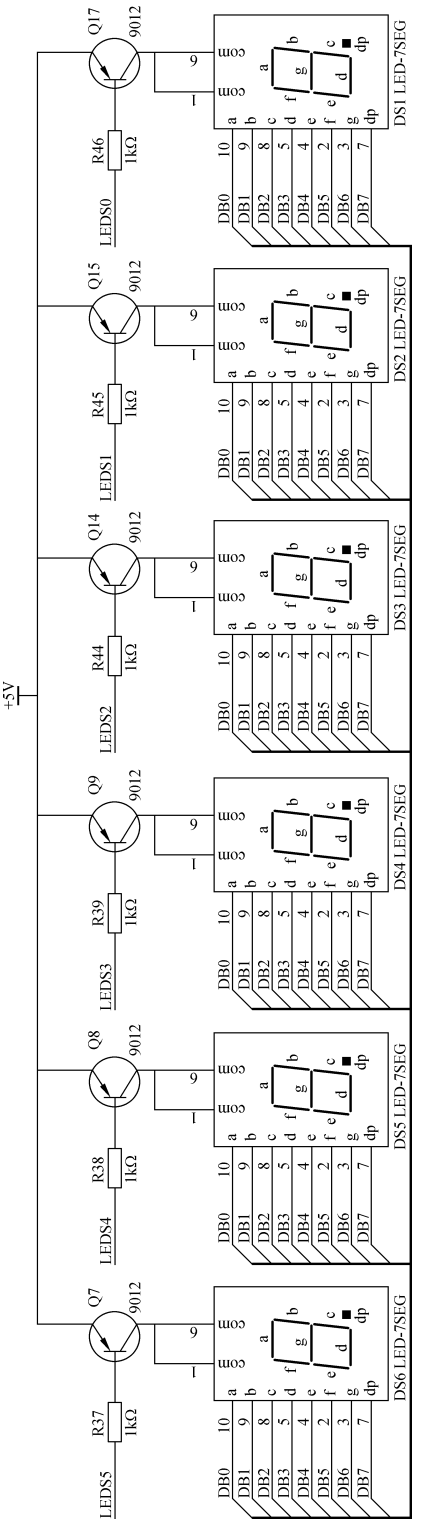


图 5-5 KST-51 数码管电路

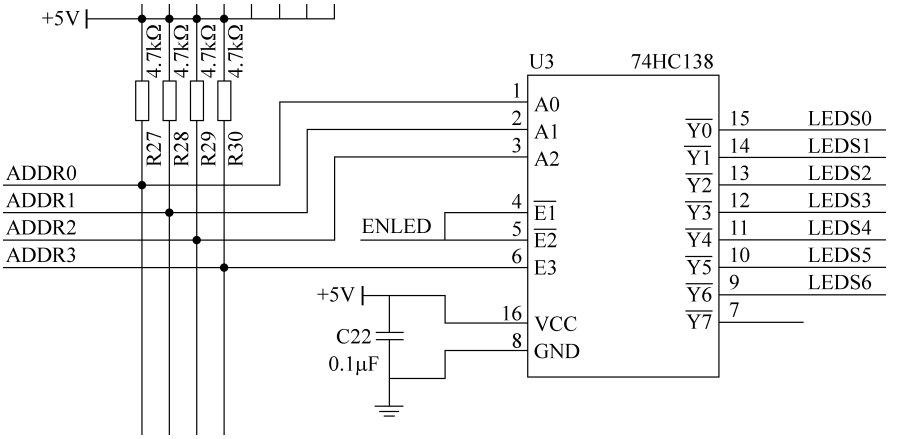


图 5-6　74HC138 控制图

数码管通常是用来显示数字的，开发板上有 6 个数码管，习惯上称之为 6 位，那控制位选择的就是 74HC138 了。而数码管内部的 8 个 LED 小灯称为数码管的段，那么数码管的段选择（即该段的亮灭）是通过 P0 口控制，经过 74HC245 驱动。

5.3.2　数码管的真值表

数码管的 8 个段直接当成 8 个 LED 小灯来控制，那就是 a、b、c、d、e、f、g、dp 一共 8 个 LED 小灯。通过图 5-3 可以看出，如果点亮 b 和 c 这两个 LED 小灯，也就是数码管的 b 段和 c 段，其他的所有的段都熄灭的话，就可以让数码管显示出一个数字 1，那么这个时候实际上 P0 的值就是 0b11111001，十六进制就是 0xF9。那么写一个程序进去，来看一看数码管显示的效果，代码如下：

```
#include <reg52.h>

sbit ADDR0 = P1^0;
sbit ADDR1 = P1^1;
sbit ADDR2 = P1^2;
sbit ADDR3 = P1^3;
sbit ENLED = P1^4;

void main()
{
    ENLED = 0;      //使能 U3,选择数码管 DS1
    ADDR3 = 1;
    ADDR2 = 0;
    ADDR1 = 0;
    ADDR0 = 0;
    P0 = 0xF9;      //点亮数码管段 b 和 c
    while (1);
}
```

大家把这个程序编译一下,并下载到单片机中,就可以看到程序运行的结果是在最右侧的数码管上显示了一个数字 1。

用同样的方法,可以把其他的数字字符都在数码管上显示出来,而数码管显示的数字字符对应给 P0 的赋值,叫作数码管的真值表。下面来列一下电路图的数码管真值表,注意,这个真值表里显示的数字都不带小数点的,如表 5-7 所示。

表 5-7 数码管真值表

字符	0	1	2	3	4	5	6	7
数值	0xC0	0xF9	0xA4	0xB0	0x99	0x92	0x82	0xF8
字符	8	9	A	B	C	D	E	F
数值	0x80	0x90	0x88	0x83	0xC6	0xA1	0x86	0x8E

大家可以把上边那个用数码管显示数字 1 程序中的 P0 的赋值随便修改成表 5-7 真值表中的数值,看看显示的数字的效果。

5.3.3 数码管的静态显示

在第 3 章学习了 74HC138,了解到 74HC138 在同一时刻只能让一个输出口为低电平,也就是说在一个时刻内,只能使能一个数码管,并根据给出的 P0 的值来改变这个数码管的显示字符,可以将此理解为数码管的静态显示。

数码管静态显示是对应动态显示而言的,静态显示对于一两个数码管还行,对于多个数码管,静态显示实现的意义就没有了。本节先用一个数码管的静态显示来实现一个简单的秒表,为后面的动态显示打下基础。

先来介绍一个 51 单片机的关键字 code。前边定义变量的时候,一般用到 unsigned char 或者 unsigned int 这两个关键字,这样定义的变量都是放在单片机的 RAM 中,在程序中可以随意去改变这些变量的值。但是还有一种数据,在程序中要使用,却不会改变它的值,定义这种数据时可以加一个 code 关键字修饰一下,这个数据就会存储到程序空间 Flash 中,这样可以大大节省单片机的 RAM 的使用量,毕竟单片机 RAM 空间比较小,而程序空间则大得多。那么现在要使用的数码管真值表,只会使用它们的值,而不需要改变它们,就可以用 code 关键字把它放入 Flash 中了,具体程序代码如下。

```
#include <reg52.h>

sbit ADDR0 = P1^0;
sbit ADDR1 = P1^1;
sbit ADDR2 = P1^2;
sbit ADDR3 = P1^3;
sbit ENLED = P1^4;

//用数组来存储数码管的真值表,数组将在下一章详细介绍
```

```
unsigned char code LedChar[ ] = {
    0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,
    0x80, 0x90, 0x88, 0x83, 0xC6, 0xA1, 0x86, 0x8E
};

void main()
{
    unsigned char cnt = 0;          //记录 T0 中断次数
    unsigned char sec = 0;          //记录经过的秒数

    ENLED = 0;                      //使能 U3,选择数码管 DS1
    ADDR3 = 1;
    ADDR2 = 0;
    ADDR1 = 0;
    ADDR0 = 0;
    TMOD = 0x01;                    //设置 T0 为模式 1
    TH0  = 0xB8;                    //为 T0 赋初值 0xB800
    TL0  = 0x00;
    TR0  = 1;                       //启动 T0

    while (1)
    {
        if (TF0 == 1)               //判断 T0 是否溢出
        {
            TF0 = 0;                //T0 溢出后,清 0 中断标志
            TH0 = 0xB8;             //并重新赋初值
            TL0 = 0x00;
            cnt++;                  //计数值自加 1
            if (cnt >= 50)          //判断 T0 溢出是否达到 50 次
            {
                cnt = 0;            //达到 50 次后计数值清 0
                P0 = LedChar[sec]; //当前秒数对应的真值表中的值送到 P0 口
                sec++;              //秒数记录自加 1
                if (sec >= 16)      //当秒数超过 0x0F(15)后,重新从 0 开始
                {
                    sec = 0;
                }
            }
        }
    }
}
```

5.4 练习题

1. 熟练掌握单片机定时器的原理和应用方法。

2. 通过研究定时器模式 1 的示意图，自己打开 STC89C52RC 数据手册的定时器部分，独立研究模式 0、模式 2 和模式 3 的示意图，锻炼一下研究示意图的能力。

3. 使用定时器来实现左右移动的流水灯程序。

4. 了解数码管的原理，掌握数码管的真值表的计算方法。

5. 编程实现数码管静态显示秒表的倒计时。