项目 3

设计制作数字频率计

3.1 学习目标

(1) 理解 C51 指针应用。
(2) 掌握 MCS-51 系列定时器/计数器的应用。
(3) 熟悉定时器/计数器工作方式及应用。
(4) 熟练掌握 C51 程序设计。
(5) 巩固数码管显示技术。

3.2 项目描述

1. 项目名称

设计制作数字频率计。

2. 项目要求

(1) 用 Keil C51、Proteus、EASY 下载软件作为开发工具。
(2) 用 AT89C51 单片机作为控制。
(3) 数码管显示。
(4) 能测试 1～100Hz 信号频率,误差允许±1Hz。
(5) 发挥扩充功能,如高位消隐、扩展频率范围等。

3. 设计制作任务

(1) 拟定总体设计制作方案。
(2) 设计硬件电路。
(3) 编制软件流程图及设计相应的源程序。
(4) 仿真调试数字频率计。
(5) 安装元件,制作数字频率计,调试功能指标。
(6) 完成项目报告。

3.3　相关知识

3.3.1　指针

1. 指针的概念

(1) 指针与指针变量

C51 中对变量的存取有两种形式：一种是按变量名存取，即直接访问；另一种是间接访问。间接访问是通过一个变量(存储单元)访问到变量 i 的地址值，再通过这个地址找到 i 的值。

在间接访问时，地址起到寻找操作对象的作用，像一个指向对象的指针，所以把地址称为指针。这种指向变量的地址的变量叫做指针变量。因此，指针变量就是用来存放地址的变量，变量的指针就是变量的地址。

(2) 指针变量的定义

指针变量用符号“*”表示指向，它的一般形式为：

```
数据类型 [存储器类型 1] * [存储器类型 2] 变量名;
```

例如：

```
int * ap , * bp;                    //将 ap、bp 定义为 int 型指针
```

说明：

①“*”为指针运算符。

②“数据类型”说明该指针变量所指向的变量类型。

③“存储器类型 1”和“存储器类型 2”是可选项，它是 C51 编译器的扩展。如果带有“存储器类型 1”选项，则指针被定义为基于存储器，选择该项有助于指定指针的长度。

```
data、idata、pdata                  1 字节指针
xdata、code                         2 字节指针
未指定 (默认)                       3 字节通用指针
```

若无此选项，则被定义为通用指针，在内存中占 3 个字节。第一个字节存放该指针存储器类型的编码，第二个和第三个字节分别存放该指针的高位和低位地址偏移量。存储器类型编码值如表 3-1 所示。

表 3-1　存储器类型编码值表

存储器类型	idata	xdata	pdatd	data	code
编码值	1	2	3	4	5

④“存储器类型 2”选项用于指定指针本身的存储器空间，一般不指定。如不指定，由编译器存储模式决定。指定时，有 data、idata、pdata、xdata、code 等选项。

C51 库函数采用了一般指针，函数可以利用一般指针来存取位于任何存储器空间的数据。在定义指针时，除必须说明所指对象的类型外，还要指定对象所在存储器空间，并确定

指针长度。另外,对指针本身也要确定其存储区。指针本身放在片外存储空间时,存取时间较长。一般将指针定义在片内存储空间。一个指针变量只能指向同一类型的变量。

在使用变量指针时,如定义外部端口的地址,必须注意定义存储类型和偏移量。例如,要将数值 0x41 写入地址为 0x8000 的外部数据存储器中,可用如下代码实现。

```
#include "absace.h"
XBYTE[0x8000]=0x41;
```

其中,XBYTE 是一个指针,它在头文件 absace.h 中的定义为:

```
#define XBYTE ((unsigned char volatile xdata * ) 0)
          //XBYTE 被定义为指向 xdata 地址空间 unsigned char 数据类型的指针,指针值为
          //"0"(volatile 的作用是让编译器不至于优化掉它的操作),这样就可以直接用
          //XBYTE[0xnnnn]或 * (XBYTE+0xnnnn)访问外部 RAM 的 0xnnnn 单元
```

(3) 指针变量的操作

指针变量只能存放地址,使用之前不仅必须先定义(声明),而且必须赋予具体的值。

指针变量的操作运算符有取地址运算符 & 和指针运算符 * 。其中,取地址符 & 用来表示变量的地址,一般形式为:

```
&变量名
```

例如,&i 表示已定义变量 i 的地址。

访问指针变量所指向的变量的一般格式为:

```
*指针变量名
```

例如:

```
int a=0x3f;
int *p;                          //定义指针变量 p
p=&a;                            //把变量 a 的地址赋给指针变量 p
P0=*p;                           //指针变量 p 指向的变量的值从 P0 口输出
```

上述程序段运行后,P0 口输出 0x3f。

注意:定义指针变量时,变量前加 * ,表示该变量为指针变量。使用指针变量时,指针变量名前加 * ,表示该指针变量指向的变量。指针变量在使用前必须先赋值(与类型相匹配的变量的地址)。

2. 指针运算

(1) 指针的赋值运算

指针的赋值运算可以把一个地址赋给一个指针变量,例如:

```
int *p1,*p2,*p3;
int a, array[12];
p1=&a;                           //将变量 a 的地址赋给指针变量 p1
p2=array                         //将数组 array 的首地址赋给指针变量 p2
p1=array[i];                     //将数组 array 第 i 个元素的地址赋给指针变量 p1
p1=p2;                           //指针变量 p2 的值赋给 p1
```

注意：不能把一个数据赋给指针变量，例如：

```
int *p1;
p1=0x12;                          //错误
```

(2) 指针与整数的加减

指针变量加(减)是将该指针变量的值(地址)和它指向的变量所占内存的字节数与要加(减)的整数的乘积相加(减)。

例如，p 指向 int 型数据，那么 p+3 指向 p 后面第 3 个对象，其值(地址)是 p 的值(地址)加 6，即

```
p++;                              //p 指向下一个对象
p--;                              //p 指向前一个对象
p+i;                              //当前对象后第 i 个对象
p-i;                              //当前对象前指向第 i 个对象
p+=i;                             //p 指向当前对象后第 i 个对象
p-=i;                             //p 指向当前对象前第 i 个对象
```

(3) 两个指针变量相减

如果两个指针变量指向同一个数组的元素，则两个指针变量值之差是两个指针之间元素个数加 1，它表示两个指针之间的距离或元素的个数。

3. 指针与数组

(1) 指向数组的指针

数组的指针是指数组的起始地址(首地址)，数组元素的指针则是数组元素的地址。

一个数组占用一段连续的内存单元，C51 规定数组名即为这段内存单元的首地址。每个数组元素按其类型的不同占有几个连续的存储单元，一个数组元素的地址就是它所占用的连续内存单元的首地址。

定义一个指向数组元素的指针变量的方法与指向变量的指针变量相同。例如：

```
int a[4];                         //定义 a 为包含 4 个整型数据的数组
int *p;                           //定义 p 为指向 int 型变量的指针变量
```

对该指针元素赋值：

```
p=&a[0];
```

把 a[0]元素的地址赋给指针变量 p，即 p 指向 a 的第 0 个元素。

C51 规定，数组名代表数组的首地址。因此，“p＝&a[0];”与“p＝a;”等价。

(2) 用指针引用数组元素

在定义数组指针并把它指向某个数组后，可以通过指针来引用数组元素，如 p＋1 指向数组中的 a[1]元素。如果定义的数组是 int 型，那么 p＋1 就是 p 的值增加 2，以指向下一个元素。p＋i 和 a＋i 就是 a[i]的地址，即指向 a 数组的第 i 个元素，*(p＋i)＝ *(a＋i)＝a[i]＝p[i]。指向数组的指针变量也可以用下标表示。

例如，利用指针来实现输出数组元素，命令如下所示。

```
#include<reg51.h>
unsigned char code dispbit[]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
unsigned char code table[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07};
void delay(unsigned char i)
  {
    unsigned char j,k;
    for(j=0;j<i;j++)
    for(k=0;k<120;k++)
      ;
  }
void main()
  {
    char i;
    unsigned char *p1,*p2;
    p1=dispbit;
      p2=table;
      while(1)
      {
        for(i=0;i<8;i++)
        {
          P1=p2[i];
          P2=p1[i];
          delay(1);
          P2=0xff;
        }
      }
  }
```

该程序实现用数码管显示“76543210”。

4. 指针与字符串

在C51中，处理字符串除了前面介绍的用字符数组实现外，还可用指针实现，即定义一个字符指针，用字符指针指向字符串中的字符。

例如：

```
#include<reg51.h>
unsigned char code dispbit[]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
unsigned char code table[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07};
void delay(unsigned char  i)
{  unsigned  char  j,k;
    for(j=0;j<i;j++)
    for(k=0;k<120;k++)
    ;
}
void main()
{
  char *string="01234567";
  char i;
  while(1)
```

```
  {
    for(i=0;i<8;i++)
    {
      P1=table[string[i]-0x30];
      P2=dispbit[i];
      delay(1);
      P2=0xff;
    }
  }
}
```

该程序同样实现用数码管显示“76543210”。

用字符指针指向字符串中的字符与用数组处理方法的不同之处是格式差异，即“char * string＝"C51";”可以写成“char * string;string＝"C51";”，而“char sd[]＝"C51";”不能写成“char sd[4];sd＝"C51";”。

5. 指针与函数

指针变量既可以作为函数的形参，也可以作为函数的实参。指针变量用作实参时，与普通变量一样，但被调用函数的形参必须是一个指针变量。

例如：

```
#include<reg51.h>
unsigned char code dispbit[]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
unsigned char code table[]={0x3f,0x06,0x5b,0x4f,0x66, 0x6d,0x7d,0x07};
void delay(unsigned  char *m)
{   unsigned char j,k;
    for(j=0;j<*m;j++)
    for(k=0;k<120;k++)
    ;
}
void main()
{
  char  i,k=5;
  char *string="01234567";
  unsigned char *p;
  p=&k;
  while(1)
  {
    for(i=0;i<8;i++)
    {
      P2=0xff;
      P1=table[string[i]-0x30];
      P2=dispbit[i];
      delay(p);
    }
  }
}
```

在程序中，延迟函数的形式参数为指针，调用时指针 p 用作实际参数。该程序实现用

数码管显示“76543210”。

3.3.2 定时器/计数器

定时器/计数器是单片机的重要部件。MCS-51 内部有两个可编程的 16 位定时器/计数器用于精确地定时与计数，广泛用于工业检测与控制。

1. MCS-51 单片机的定时器/计数器

8051 单片机内部有两个 16 位的可编程定时器/计数器 T0 和 T1。T0 由 TH0 和 TL0 构成，T1 由 TH1 和 TL1 构成。TL0、TL1、TH0、TH1 的访问地址依次为 8AH～8DH，每个寄存器均可单独访问。

T0 或 T1 用作计数器时，对芯片引脚 T0(P3.4)或 T1(P3.5)上输入的脉冲计数。用作定时器时，对内部机器周期脉冲计数。

2. 定时与计数功能

定时器/计数器 T0 和 T1 的核心是计数器，其基本功能是加 1。在特殊功能寄存器 TMOD 中都有一个控制位，选择 T0 或 T1 作定时器还是计数器使用。

作计数器使用时，对来自输入引脚 T0(P3.4)或 T1(P3.5)的外部信号计数，外部脉冲的下降沿将触发计数，计数器加 1，新计数值于下一个机器周期装入计数器。因而，识别一个计数脉冲需要两个机器周期，外部脉冲的最高频率为振荡频率的 1/24。

作定时器使用时，计数器对内部机器周期计数，每过一个机器周期，计数器加 1。计满溢出时，若计数值为 N，定时器的定时时间为

$$t = T_c(\text{机器周期}) \times N$$

MCS-51 单片机的一个机器周期由 12 个振荡脉冲组成，则机器周期为

$$T_c = 12/f_{osc}$$

如果单片机系统采用 12MHz 晶振，则计数周期(即机器周期)为

$$T_c = 12/(12 \times 10^6) = 1(\mu s)$$

若计数值为 N，则定时 $N\mu s$。

定时器/计数器 T0 和 T1 是加法计数器，每来一个计数脉冲，计数器的值加 1，加满则溢出(溢出值即计数最大值，常用 M 表示)。如果要计 N 个单位就溢出，应向计数器赋初值 X。

$$X = M(\text{最大计数值}) - N(\text{计数值})$$

假设对 12MHz 时钟定时 10μs，则计数值 N 为

$$N = t/T_c = 10/1 = 10$$

初值 X 为

$$X = M(\text{最大计数值}) - N(\text{计数值}) = M(\text{最大计数值}) - 10$$

3. 定时器/计数器的工作方式

对于 8051 单片机，通过设置 TMOD 寄存器中的 M0、M1 位，可选择下述 4 种工作方式。

(1) 方式 0

方式 0 构成一个 13 位定时器/计数器，最大计数值 $M=2^{13}=8192$。T0 逻辑电路结

构如图 3-1 所示。T1 的结构和操作与 T0 完全相同。

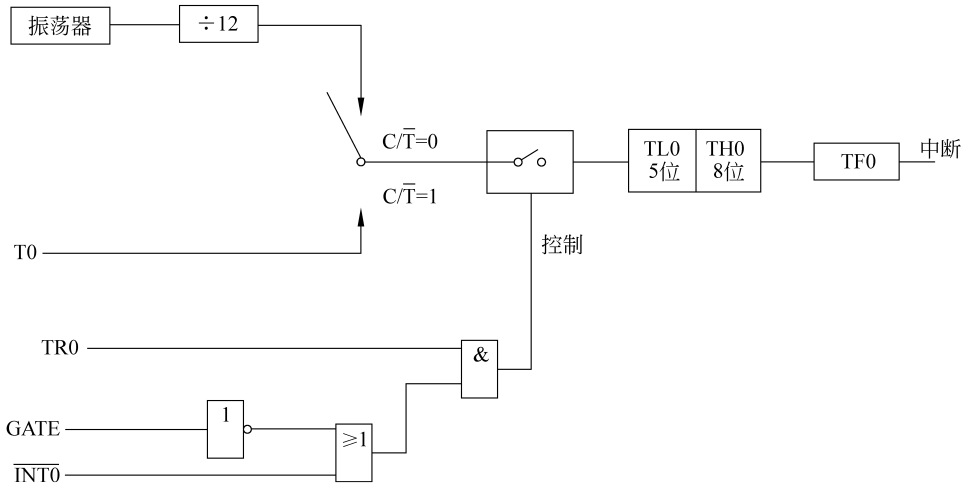

图 3-1　T0(或 T1)方式 0 时的逻辑电路结构图

16 位加法计数器(TH0 和 TL0)只用了 13 位。其中,TH0 占高 8 位,TL0 占低 5 位(高 3 位未用)。当 TL0 低 5 位溢出时,自动向 TH0 进位;而 TH0 溢出时,中断位 TF0 自动置位,并申请中断。

当 C/$\overline{T}$=0 时,多路开关连接 12 分频器输出,T0 对机器周期计数。此时,T0 为定时器。

当 C/$\overline{T}$=1 时,多路开关与 T0(P3.4)相连,T0 为计数器。外部计数脉冲由 T0 脚输入,当外部信号电平发生由 0 到 1 的跳变时,计数器加 1。

当 GATE=0 时,或门被封锁,$\overline{INT0}$信号无效。或门输出常“1”,打开与门,TR0 直接控制 T0 的启动和关闭。TR0=1,接通控制开关,T0 从初值开始计数直至溢出。溢出时,16 位加法计数器为“0”,TF0 置位,申请中断。如要循环计数,T0 需重置初值,且需用软件将 TF0 复位。TR0=0,则与门被封锁,控制开关被关断,停止计数。

当 GATE=1 时,与门的输出由$\overline{INT0}$的输入电平和 TR0 位的状态来确定。若 TR0=1,则与门打开,外部信号电平通过$\overline{INT0}$引脚直接开启或关断定时器 T0。当$\overline{INT0}$为高电平时,允许计数,否则停止计数。TR0=0,则与门被封锁,控制开关被关断,停止计数。

(2) 方式 1

定时器工作在方式 1 时,是 16 位的定时计数器,最大计数值 $M=2^{16}=65536$,其逻辑结构如图 3-2 所示。

方式 1 构成一个 16 位定时器/计数器,其结构与操作几乎完全与方式 0 相同,差别是二者计数位数不同。

(3) 方式 2

定时器/计数器工作在方式 2 时,为 8 位定时计数器,最大计数值 $M=2^8=256$。其逻辑结构如图 3-3 所示。

此时,16 位加法计数器的 TH0 和 TL0 具有不同功能。其中,TL0 是 8 位计数器,

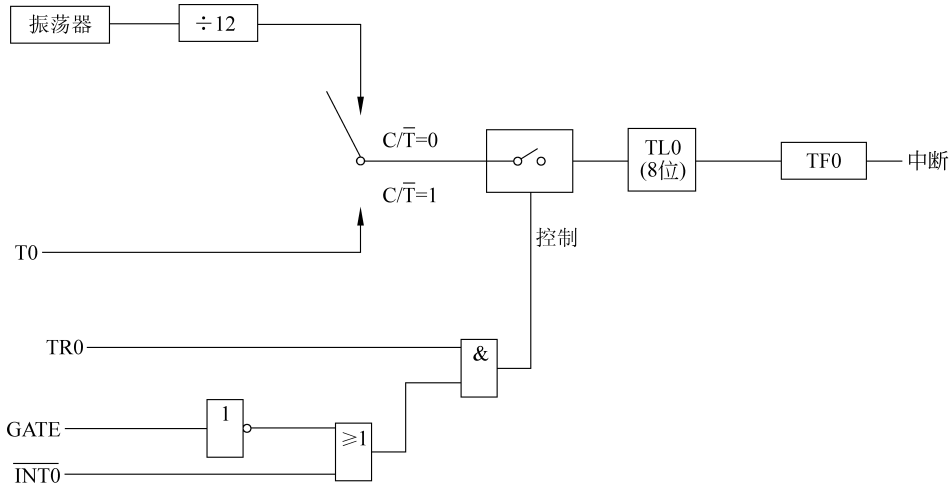

图 3-2　T0(或 T1)方式 1 时的逻辑结构图

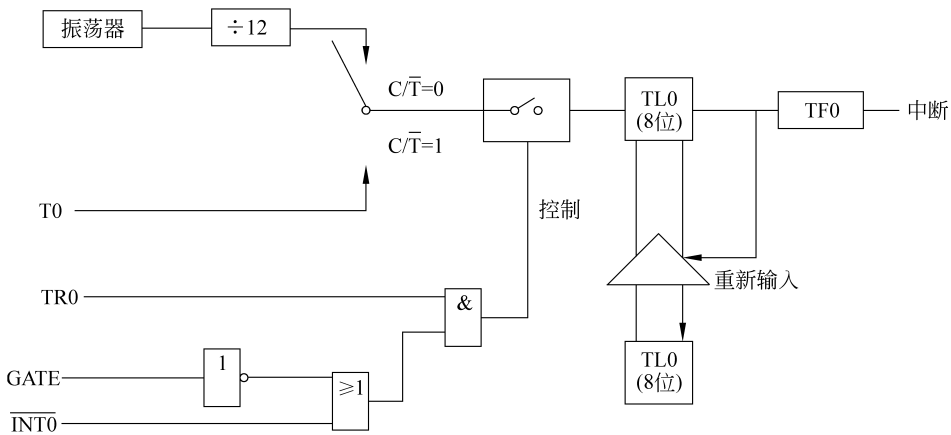

图 3-3　T0(或 T1)方式 2 时的逻辑结构图

TH0 是重置初值 8 位缓冲器。TH0 和 TL0 赋相同的初值，一旦 TL0 计数溢出，TF0 将被置位，TH0 中的初值自动装入 TL0。因此，方式 2 具有初值自动装入功能，适合用作较精确的定时脉冲信号发生器。

(4) 方式 3

定时器/计数器工作在方式 3 时，T0 被分解成两个独立的 8 位计数器 TL0 和 TH0，最大计数值 M 值均为 256，其逻辑结构图如图 3-4 所示。其中，TL0 占用原 T0 的控制位、引脚和中断源。除计数位数与方式 0、方式 1 不同外，其功能、操作与方式 0、方式 1 相同，可定时、计数。而 TH0 占用原定时器 T1 的控制位 TF1 和 TR1，同时还占用了 T1 的中断源，其启动和关闭仅受 TR1 置 1 或清 0 控制，TH0 只能对机器周期进行计数，因此，TH0 只能用作简单的内部定时，不能用作对外部脉冲进行计数，是定时器 T0 附加的一个 8 位定时器。

方式 3 时，T1 仍可设置为方式 0、方式 1 或方式 2。但由于 TR1、TF1 及 T1 的中断源已

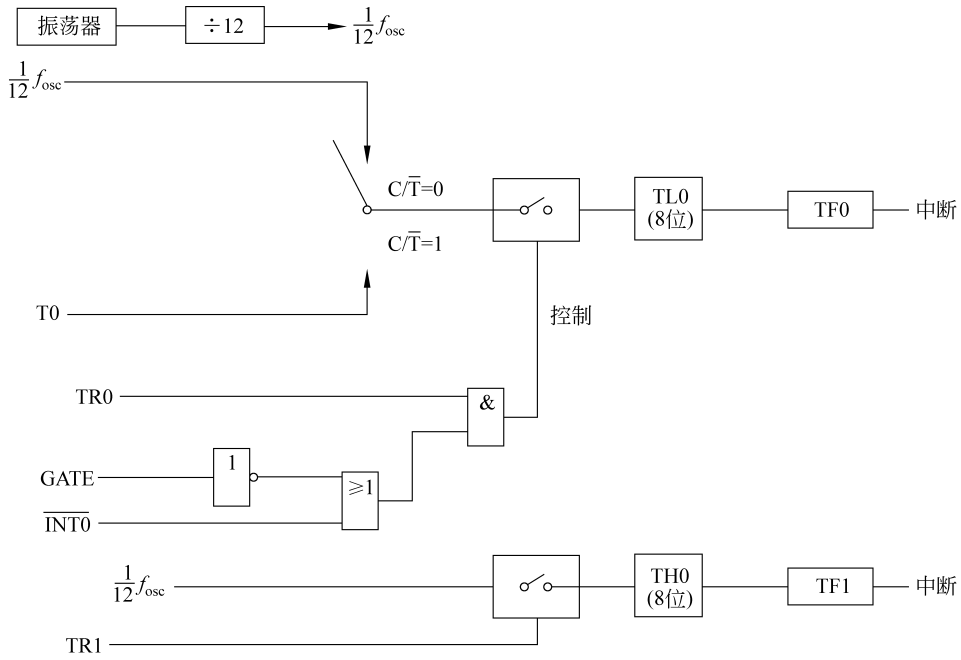

图 3-4　T0 方式 3 时的逻辑结构

被定时器 T0 占用，此时，T1 仅由控制位 C/$\overline{T}$ 切换其定时或计数功能，当计数器溢出时，只能将输出送往串行口。此时，T1 一般用作串行口波特率发生器或不需要中断的场合。

4. 定时器/计数器的控制

MCS-51 单片机的定时器/计数器有方式寄存器 TMOD 和控制寄存器 TCON 两个工作寄存器，用户通过编程定时器/计数器的方式寄存器 TMOD 和控制寄存器 TCON 的控制内容来选择其用途，设定其工作方式，赋计数初值以及启动等。

(1) 定时器/计数器方式寄存器 TMOD

TMOD 为 T1、T2 的工作方式寄存器，其格式如下所示。

TMOD：	D7	D6	D5	D4	D3	D2	D1	D0
(89H)	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
	T1				T0			

TMOD 的低 4 位为 T0 的方式字段，高 4 位为 T1 的方式字段，它们的含义完全相同。

M1 和 M0：方式选择位，定义如表 3-2 所示。

表 3-2　计数器工作方式

M1	M0	工作方式	功 能 说 明	最大计数值
0	0	方式 0	13 位计数器	$2^{13}=8192$
0	1	方式 1	16 位计数器	$2^{16}=65536$
1	0	方式 2	自动再装入 8 位计数器	$2^{8}=256$
1	1	方式 3	T0 分成两个 8 位计数器，定时器停止计数	$2^{8}=256$