第 3 章　可编程时间间隔定时器芯片 82C54

3.1　概述

82C54 是专为 Intel 系列微处理器而设计的一种可编程时间间隔定时器/计数器，它是一种通用芯片，在系统软件中，可以把多级定时元素当成 I/O 端口中的一个阵列看待。

82C54 是一种在微处理器系统中实现定时和计数功能的外围电路，其内拥有三个独立的 16 位的计数器，每个计数器都允许输入的时钟速率高达 10MHz。各种操作方式都可以通过程序设计的方法来实现。82C54 与早期的 HMOS 的 8254 以及 8253 都兼容。

82C54 所配备的 6 个可编程的计时器方式，可以把 82C54 作为事件计数器、操作所用时间指示器，也可以将它们设置成单脉冲发生器、方波发生器等多种用途。可编程时间间隔定时器芯片 82C54 拥有以下特点：

- 与所有 Intel 系列微处理器兼容。
- 较高的操作速度。与 8MHz 的 8086、80186 一起可实现“零等待状态”操作。
- 可以处理从 DC-10MHz 这么大范围的输入。
- 具有较强的适应性。既可以在标准的温度范围下运行，也可在较宽的温度范围下工作。
- 三个独立的 16 位的计数器。
- 低功耗的 CHMOS。
- 与 TTL 完全兼容。
- 有 6 种可编程的计数模式。
- 以二进制或 BCD 计数。
- 状态读返回命令。

82C54 有效地解决了微机系统内常见的一个问题，像在系统控制之下精确的时间延迟问题。摒弃了以往那种通过软件设置定时循环的笨办法。程序设计人员利用 82C54，可根据需要将所需的延迟时间通过程序对其中一个计数器进行设置，延迟时间一旦设置妥当，82C54 将会根据设置参数去中断 CPU 的执行。这样就可使软件的额外开销减至最小，延迟时间长短也更加容易调节。

82C54 能实现的其他类的计数器/定时器功能还有：

- 实时时钟
- 平滑计数器
- 数字一次通过
- 可编程速率生成器
- 方波发生器
- 二进制速率乘法器

- 复杂波形发生器
- 复杂马达控制器

3.2　82C54 的体系结构

3.2.1　82C54 的方框图

图 3-1 展示了 82C54 芯片的信号接口。在微处理器系统中，82C54 被当成一个外围电路来对待的。另外，它能以存储器映像方式存在于存储器地址空间，或以 I/O 映像方式存在于 I/O 地址空间。82C54 的微处理器接口允许微处理器对它的内部寄存器进行读或写操作。所以，可将它设置为各种不同的操作方式。

首先介绍 82C54 的微处理器接口信号。该微处理器接口包括一个 8 位的双向数据总线 D0～D7。在微处理器和 82C54 芯片之间进行数据传送时就要使用这些数据信号线。寄存器地址输入信号 A0 和 A1 用来选择要访问的寄存器。读控制信号(RD＃)和写控制信号(WR＃)用来指明要对哪些寄存器地址进行读出或写入操作。另外，它还提供片选(CS＃)输入信号，以开启 82C54 的微处理器接口。此输入信号可以使设计人员把 82C54 定位在特定的存储器或 I/O 地址。

图 3-1 所示方框图右边，可以看到每个计数器都用的三个信号。例如，计数器 0 有两个输入信号，它们分别标为 CLK0 和 GATE0。其中时钟输入端的脉冲信号(CLK0)用来使计数器 0 减 1。门控输入信号(GATE0)用来允许或禁止该计数器工作。要允许计数器 0 工作，GATE0 信号必须为 1(高电平)。比如，在方波操作方式中，由于该计数器是连续运行的，所以必须把 GATE0 信号固定为 1，并把一个连续的时钟信号加到 CLK0 输入端。82C54 的最大时钟频率为 10MHz。计数器 0 还有一个标识为 OUT0 的输出信号线。依据所选择的操作方式，该计数器在 OUT0 端产生连续的时钟信号或单个的脉冲信号。例如，若设置的是方波操作方式，这个输出就是一个连续的时钟信号。

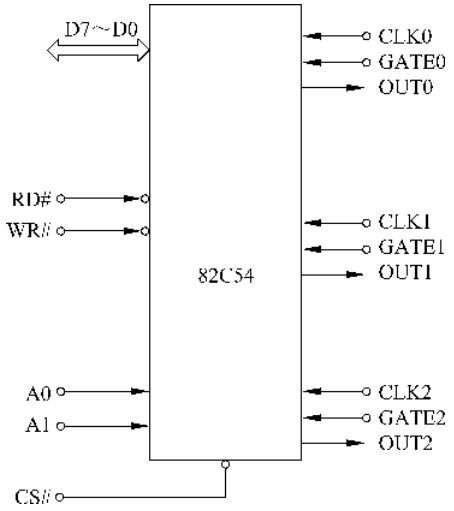

图 3-1　可编程时间间隔定时器芯片 82C54 方框图

图 3-2 展示了 82C54 芯片每个管脚上的信号。表 3-1 列出了 82C54 芯片每个管脚信号的功能说明。

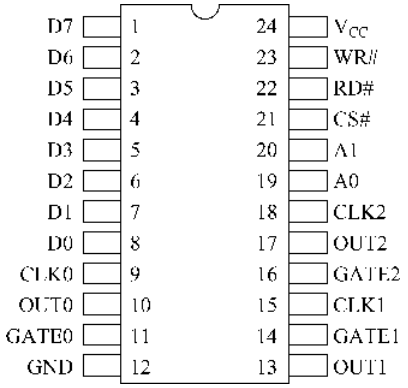

图 3-2 82C54 管脚

表 3-1 82C54 芯片管脚说明

符号	管脚号		类型	功能
	DIP	PLCC		
D7～D0	1～8	2～9	I/O	数据：双向三态数据总线，连到了系统的数据总线上
CLK0	9	10	I	时钟 0：计数器 0 的时钟输入
OUT0	10	12	O	输出 0：计数器 0 的输出
GATE0	11	13	I	门 0：计数器 0 的门输入
GND	12	14		地
OUT1	13	16	O	输出 1：计数器 1 的输出
GATE1	14	17	I	门 1：计数器 1 的门输入
CLK1	15	18	I	时钟 1：计数器 1 的时钟输入
GATE2	16	19	I	门 2：计数器 2 的门输入
OUT2	17	20	O	输出 2：计数器 2 的输出
CLK2	18	21	I	时钟 2：计数器 2 的时钟输入
A0，A1	19，20	22，23	I	地址：用于从三个计数器中选出其中的一个，或用来选择控制字寄存器进行读写操作。通常是连到系统地址总线上 A1 A0 选择 0 0 计数器 0 0 1 计数器 1 1 0 计数器 2 1 1 控制字寄存器
CS#	21	24	I	芯片选择：若这个输入信号为低电平，则允许 82C54 芯片去响应 RD# 和 WR# 信号；否则不理睬 RD# 和 WR# 信号
RD#	22	26	I	读控制：在 CPU 进行读操作期间，这个输入信号为低电平
WR#	23	27	I	写控制：在 CPU 进行写操作期间，这个输入信号为低电平
V_{CC}	24	28		电源：接到 5V 电源
NC		1，11 15，25		不连接

说明：DIP 与 PLCC 是 82C54 两种不同的封装结构。就其功能而言，塑封的 28 管脚的 82C54 与 24 管脚的 DIP 是一样的。

3.2.2　82C54 的体系结构

图 3-3 展示了 82C54 芯片的体系结构。其内配备有数据总线缓冲器、读/写逻辑、控制字寄存器和三个计数器等部件。其中数据总线缓冲器和读/写控制逻辑就是前面介绍过的微处理器接口。

1. 数据总线缓冲器

图 3-3 展示的数据总线缓冲器，实际上是 82C54 与系统总线的接口，它是一种 3 态、双向 8 位的缓冲器。

2. 读/写逻辑部件

82C54 的读/写逻辑功能模块接受来自系统总线的输入信息，并且还会产生控制 82C54 其他功能模块的信号。用 A1 和 A0 从三个计数器中选择其中的一个计数器或者是控制字寄存器进行读/写操作。若输入信号 RD＃为低电平，则是告知 82C54，CPU 将从其中一个计数器进行读操作。若输入信号 WR＃为低电平，则是通知 82C54，CPU 或者是在写一个控制字，或者是在写一个初始计数值。而 RD＃和 WR＃均是受 CS＃的控制，若 82C54 一旦被 CS＃的低电平所控制，RD＃和 WR＃将不被理睬。

WR＃和 CLK 被当成同步信号使用，是通过把 CLK 信号输入到 82C54 的计数器这一途径来完成的。

3. 控制字寄存器

当 A1A0＝11，由读/写逻辑部件来选择控制字寄存器，如图 3-3 所示。若 CPU 此时正在对 82C54 进行一次写操作，在这种情况下，则是把数据存放在控制字寄存器内，且被当成控制字去规定各计数器的操作。

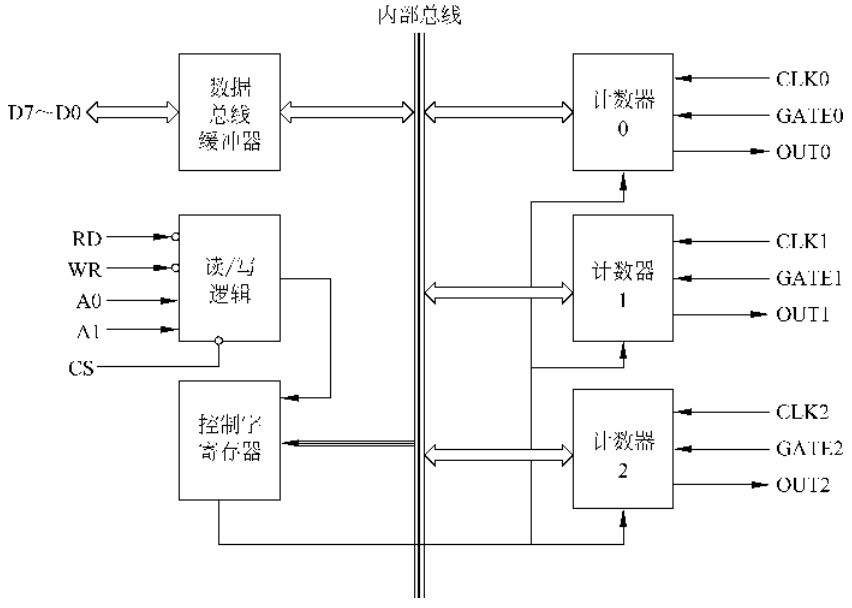

图 3-3　82C54 芯片的体系结构

控制字寄存器只能写,随着读回命令变化,状态信息也随之变化。

控制字寄存器部分实际上是由三个 8 位的寄存器构成,用它们来设置计数器 0、1 和 2 的操作。如图 3-4 所示的控制字寄存器,其中最高 2 位是用来分配该控制字到某一个计数器的编码。例如,若将这两位设置成 01,则选择的是计数器 1; D1～D3 这三位是方式选择码 M0 M1 M2,用这三位来选择计数器的 6 种操作方式之一; 最低位(D0)标记为 BCD,用它来选择二进制计数方式或 BCD 计数方式。假如将这一位设置为 0,那么该计数器按二进制 16 位进行计数操作。最后 2 位 D5、D4 被标记为 RW1、RW0 用来设置将字节读出或写入 16 位计数寄存器的操作顺序。

例 3-1 如果 82C54 经数据总线接收到一个其内容为 10010000B 的控制字,在这种情况下它的配置将如何进行设置?

由于此控制字中的 SC1SC0 = 10,所以控制字中的其余各位将要进行对计数器 2 的设置。再用控制字的内容来对照 82C54 控制字的格式,由于 RW1RW0=01,把计数器 2 设置成了仅读/写最低有效字节。这就意味着在对计数器 2 进行写操作时,是把数据装入它的计数寄存器的低序字节,其方式码为 000,意味着该计数器选择的是方式 0 操作; 最后一位(BCD 位)为 0,选择的是二进制计数。

控制字格式

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC—选择计数器

SC1	SC0	
0	0	选择计数器 0
0	1	选择计数器 1
1	0	选择计数器 2
1	1	读回命令

RW—读/写

RW1	RW0	
0	0	计数器锁存命令
0	1	仅读/写最低有效位字节
1	0	仅读/写最高有效位字节
1	1	先读/写最低有效位字节,然后再读/写最高有效位字节

M—方式

M2	M1	M0	
0	0	0	方式 0
0	0	1	方式 1
X	1	0	方式 2
X	1	1	方式 3
1	0	0	方式 4
1	0	1	方式 5

BCD

0	为 16 位的二进制计数器
1	二进制编码的十进制(BCD)计数器(4 位十进制)

注: 图中的 X 位,为保证与后续产品兼容其值应为 0,现在不用管它。

图 3-4 82C54 控制字格式

4. 三个计数器

在图 3-3 中所示的三个计数器均为 16 位长,可在减 1 计数方式下操作。即当被有效的门控信号允许后,每来一个时钟脉冲,均将使计数值减 1。每个计数器的 16 位计数寄存器的值必须在初始化时予以装入。保存在该计数寄存器中的值可在任何时刻通过软件读出。

为了对 82C54 的计数器进行读/写操作，或对它的控制字寄存器进行装入操作，微处理器必须执行相应的读/写操作命令。表 3-2 列出了读/写每个寄存器的总线操作控制信息。例如，为了写控制寄存器，寄存器地址信号线必须是 A1A0＝11，且控制信号线必须是 WR＃＝0，RD＃＝1 及 CS＃＝0。

表 3-2　读/写 82C54 寄存器操作

CS＃	RD＃	WR＃	A1	A0	描　　述
0	1	0	0	0	写计数器 0
0	1	0	0	1	写计数器 1
0	1	0	1	0	写计数器 2
0	1	0	1	1	写控制字
0	0	1	0	0	读计数器 0
0	0	1	0	1	读计数器 1
0	0	1	1	0	读计数器 2
0	0	1	1	1	无操作(三态)
1	X	X	X	X	无操作(三态)
0	1	1	X	X	无操作(三态)

例 3-2　试编写一段程序，按下列要求设置图 3-5 中 82C54 的三个计数器。

计数器 0：二进制计数，在操作方式 0 下操作，计数初值为 1234H。

计数器 1：BCD 计数，在操作方式 2 下操作，计数初值为 100H。

计数器 2：二进制计数，在操作方式 4 下操作，计数初值为 1FFFH。

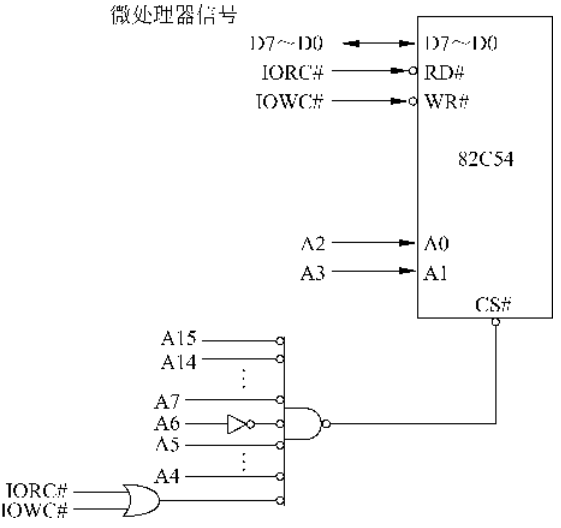

图 3-5　82C54 的微处理器接口举例

解决这个问题的步骤是首先确定 82C54 的基址。这个基址即计数器 0 的地址，是通过把 A3A2 设置成 00 来确定的。由图可以看出，为选择 82C54，需使信号 CS＃＝0，这样就必须让地址信号 A15A14A13…A6A5A4 ＝ 000000000100B。

将以上这部分地址与 A3A2A1A0＝0000 组合在一起，即形成基地址：

0000000001000000B = 40H

82C54 的基址确定为 40H 之后,在选择方式寄存器时又须使 A3A2=11,所以方式寄存器(即控制寄存器)的地址是 4CH。以此类推,计数器 0、计数器 1、计数器 2 的地址分别为 40H、44H、48H。

下面再根据 82C54 控制字中各位的定义,确定这三个计数器的方式字。应为:

计数器 0 的方式字=00110000B=30H

计数器 1 的方式字=01010101B=55H

计数器 2 的方式字=10111000B=B8H

下面这段程序就是设置这个 82C54 的方式和计数初值的:

```
MOV  AL, 30H            ;设置计数器 0 的操作方式
OUT  4CH, AL            ;
MOV  AL, 55H            ;设置计数器 1 的操作方式
OUT  4CH, AL            ;
MOV  AL, 0B8H           ;设置计数器 2 的操作方式
OUT  4CH, AL            ;
MOV  AL, 34H            ;对计数器 0 设置计数初值
OUT  40H, AL            ;
MOV  AL, 12H            ;
OUT  40H, AL            ;
MOV  AL, 00H            ;对计数器 1 设置计数初值
OUT  44H, AL            ;
MOV  AL, 01H            ;
OUT  44H, AL            ;
MOV  AL, 0FFH           ;对计数器 2 设置计数初值
OUT  48H, AL
MOV  AL, 1FH
OUT  48H, AL
```

如前所述,可随时读出 82C54 计数寄存器的内容。但如何通过软件实现这一操作,一种方法是简单地通过一条输入指令来读取相应寄存器的内容。由表 3-2 可见,为要读出计数寄存器 0 的内容,输入控制信号必须是 CS#=0,RD#=0 及 WR#=1,且寄存器地址码 A1A0=00。为了确保读出的是计数寄存器 0 的有效计数值,必须在进行读操作之前禁止该计数器的计数操作。最有效的实现方法是,在必须形成读操作之前把信号 GATE0 输入转变成 0。在读取该计数值时,两个字节被分开读取,先读取低序字节,然后再读取高序字节。

当然,也可以不采用先禁止计数操作的方法来读取计数寄存器的内容。而是在计数操作进行过程中来读出计数值。为了用软件来实现这一操作,必须先向方式寄存器输出一条把计数器的当前值锁存到暂存寄存器的命令。由图 3-4 可见,将控制字中的 D5D4 设置为 00,就规定了这种锁存操作方式。一旦将这样的控制字写入 82C54,即可读出刚被锁存时的暂存寄存器的内容。

例 3-3 试编写一个在计数进行的过程中进行读计数器 2 内容的程序段,并把读取的值装入 AX 寄存器。假设 82C54 被定位在 I/O 地址 40H 上。

解决这一问题的途径是,先锁存计数器 2 的内容,然后从暂时寄存器读取这个值,下面

这段程序即可实现这一功能。

```
MOV  AL, 1000×××× B        ;锁存计数器 2,×××× 必须
                           ;是前面已经规定的方式和计数类型
OUT  4CH, AL               ;
IN   AL, 48H               ;读低序字节
MOV  BL, AL                ;
IN   AL, 48H               ;读高序字节
MOV  AH, AL                ;
MOV  AL, BL                ;(AX) = 计数器 2 的值
```

82C54 的三个计数器在操作上其功能都是一样的,所以我们仅介绍一个计数器的功能即可。图 3-6 展示了计数器内部结构图。

这三个计数器都是各自独立的,每一个计数器都可以在不同的方式下进行操作。

在图 3-6 中所展示的那个控制字寄存器,并不是计数器本身的一个部分,但控制字寄存器中的内容却决定了计数器如何进行操作。

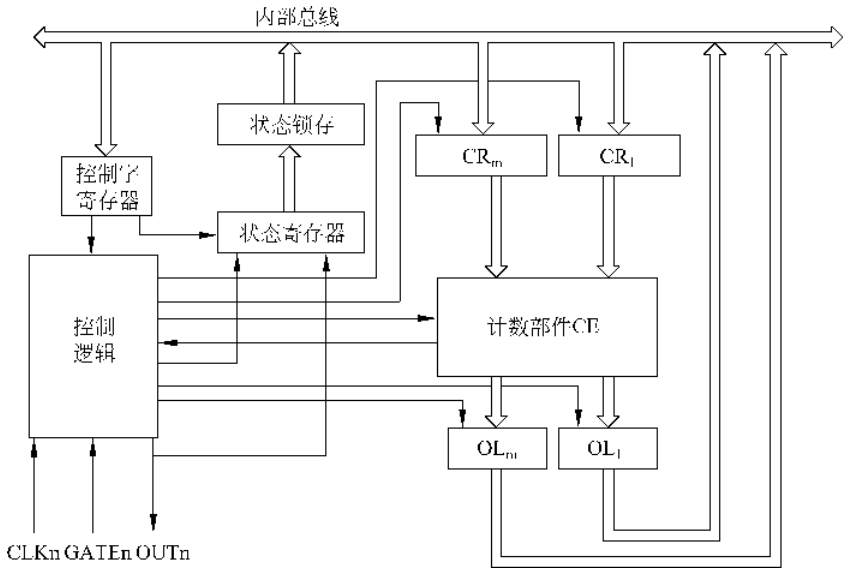

图 3-6　计数器内部结构图

从图 3-6 中可以看出,计数器内包括一个状态寄存器。一旦状态寄存器被锁存,其内保存的是控制字寄存器的当前内容和输出状态以及空计数标志。

在图 3-6 中把计数器标记成计数部件 CE(counting element),它是一个 16 位的同步计数器。

计数器中的 OL_m 和 OL_l 是两个 8 位的锁存器。OL 表示的是输出锁存,而下标 m 和 l 分别表示的是最高有效字节和最低有效字节。通常把二者看成是一个部件,统称为输出锁存。这两个锁存器通常是跟在计数部件 CE 的下面。但若在适当的时候把计数器锁存命令发送给 82C54,此时锁存器“锁存”的是 CPU 读之前的计数值,然后再返回到下面的计数部件 CE。通过计数器的控制逻辑驱动内部总线,一次仅允许进行一次锁存操作。由此可以看出,在 8 位的内部总线上是怎样对 16 位的计数器进行传送操作的。值得说明的一点是,计

数部件 CE 自己是不能进行读操作的;每当读计数值时,都是 OL 在进行读操作。

与此类似,计数器内的两个 8 位寄存器被分别称之为 CR_m 和 CR_l。通常也是把二者看成是一个部件,被标记以计数寄存器 CR(count register)。当把一个新计数值写入计数器时,总是先将计数值存放在计数寄存器 CR 内,然后才传送至计数部件 CE。传送逻辑允许通过内部总线一次仅给一个寄存器进行装载操作。两个字节信息被同时传送到计数部件 CE,计数器一旦受到程序控制,将对 CR_m 和 CR_l 进行清除操作。按照这种操作方式,如果计数器经由程序已经处理了一个字节计数值(或者是最高有效字节,或者是最低有效字节),那另一个字节将被清成 0。请注意:计数部件 CE 是不能写的,每当一个计数值被写时,总是将其写到控制寄存器 CR 内。

在图 3-6 中还展示了计数器的控制逻辑,通过控制逻辑将 CLKn、GATEn 以及 OUTn 连到了计数器的外部世界。

5. 82C54 系统接口

系统软件把 82C54 当成一个外围 I/O 端口的阵列。此阵列由三个计数器和一个用于 MODE 程序设计的方式控制寄存器组成。

图 3-7 展示了 82C54 系统接口,通常选择 A0 输入,A1 又被连到 A0,A1 是 CPU 的地址总线信号。利用线选方法,从地址总线可直接得到 CS# 信号。

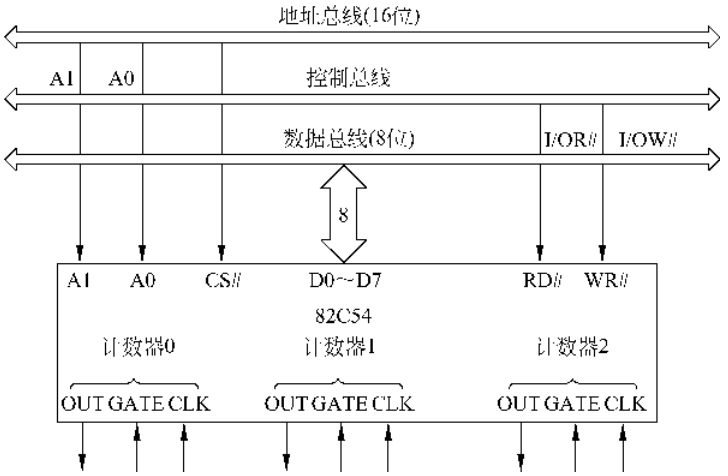

图 3-7 82C54 系统接口

3.3 程序设计基础

加电之后,82C54 的状态是不确定的。在这种情况下,计数值以及所有计数器的输出也是不确定的。

当用程序控制的方法对 82C54 进行操作时,又是怎样来确定每一计数器操作的呢?其实,在使用之前,每一个计数器必须用程序实施控制。没有使用的计数器则不必实施程序控制。

3.3.1　82C54 程序控制

通过给 82C54 写一个控制字，然后再初始化计数值的方法对计数器进行程序控制。控制字格式如图 3-4 所示。

当 A1 A0 ＝11 时，全部控制字信息都写入被选中的控制字寄存器内。由控制字自己来说明哪一个计数器实施了程序控制。

与以上相反，初始值被写进了计数器，而不是将其写入控制字寄存器。用 A1A0 的输入信号来选择被写的计数器，初始计数值格式由所用控制字确定。

3.3.2　写操作

对 82C54 进行程序设计的方法、手段都是非常灵活的。只需记住下面两点就可以了：

- 对每一个计数器而言，在写初始计数值之前，必须写控制字寄存器。
- 初始计数值必须根据控制字所规定的格式计数，要么只是最低有效字节，要么只是最高有效字节，或者是先最低有效字节，然后再最高有效字节。

由于控制字寄存器和三个计数器都是各自独立地进行寻址（由 A1A0 输入选择），且由每个控制字来规定计数器（由 SC0 和 SC1 确定），不用规定指令顺序。任何程序设计的顺序均可根据以上所提及的两点就足够了。

在任何时候，都可以将一个新的初始计数值写到某个计数器中去，且不会影响到计数器的程序控制方式。但由于操作方式的规定，计数过程会受到一定的影响。新计数值必须依据程序控制计数的格式处理。

如果一个计数器在程序控制之下对两个字节的计数值进行读写操作，在如下应用时应倍加小心：当一个程序在向另一个程序写第一个字节和第二个字节之间，而另一个程序也在向同一个计数器进行写操作时，在这种情况下是不能传送控制的。否则，计数器内装载的会是一个不正确的计数值。

表 3-3 列出了一些可用的程序设计次序。

表 3-3　一些可用的程序设计次序

描　　述	A1	A0	描　　述	A1	A0
控制字—计数器 0	1	1	控制字—计数器 2	1	1
计数值的低字节—计数器 0	0	0	控制字—计数器 1	1	1
计数值的高字节—计数器 0	0	0	控制字—计数器 0	1	1
控制字—计数器 1	1	1	计数值的低字节—计数器 2	1	0
计数值的低字节—计数器 1	0	1	计数值的高字节—计数器 2	1	0
计数值的高字节—计数器 1	0	1	计数值的低字节—计数器 1	0	1
控制字—计数器 2	1	1	计数值的高字节—计数器 1	0	1
计数值的低字节—计数器 2	1	0	计数值的低字节—计数器 0	0	0
计数值的高字节—计数器 2	1	0	计数值的高字节—计数器 0	0	0
控制字—计数器 0	1	1	控制字—计数器 1	1	1

续表

描　　述	A1	A0	描　　述	A1	A0
计数器字—计数器 1	1	1	控制字—计数器 0	1	1
控制字—计数器 2	1	1	计数值的低字节—计数器 1	0	1
计数值的低字节—计数器 2	1	0	控制字—计数器 2	1	1
计数值的低字节—计数器 1	0	1	计数值的低字节—计数器 0	0	0
计数值的低字节—计数器 0	0	0	计数值的高字节—计数器 1	0	1
计数值的高字节—计数器 0	0	0	计数值的低字节—计数器 2	1	0
计数值的高字节—计数器 1	0	1	计数值的高字节—计数器 0	0	0
计数值的高字节—计数器 2	1	0	计数值的高字节—计数器 2	1	0

说明：

1. 在以上 4 种例子中，所有的计数器都是在程序的控制之下读/写两个字节计数值。

2. 在表中列出的仅是诸多可用的程序设计次序中的 4 种。

3.3.3 读操作

82C54 很容易做到：既可以经常去读计数器内的值而又不会干扰计数过程。

读计数器内的值有三种可行办法供选择：一是通过一次简单的读操作；二是通过计数器锁存命令；三是通过读回命令。下面分别给予解释。

1. 简单的读操作

简单的读操作，就是进行一次读计数器的操作。用输入信号 A1A0 来选择计数器，但又必须禁止使用被选中的计数器的 CLK 输入(通过 GATE 或者是外部逻辑)信号。否则，当在进行读操作时，计数值可能是过程中的一个值，从而得到一个不正确的结果。

2. 计数器锁存命令

读操作的第二个方法使用的是“计数器锁存命令”。它很像一个控制字，因为它是把命令写到控制字寄存器内的。它又是当 A1A0＝11 时被选中的控制字寄存器。说它像控制字，还有一点根据是，用 SC0 和 SC1 位选中的是三个计数器中的一个，而像计数器锁存命令中的 D5 和 D4 这两位则是把命令与控制字严格区别开来。图 3-8 展示了计数器锁存命令格式。

A1 A0 ＝ 1 1；CS＃＝0；RD＃＝1；WR＃＝0

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

SC1、SC0 表示计数器被锁存。D5 D4 ＝ 00 表示计数器锁存命令。

SC1	SC0	计数器
0	0	0
0	1	1
1	0	2
1	1	写回命令

注：图中的 X 位，为保证与后续产品兼容其值应为 0，现在不用管它。

图 3-8　计数器锁存命令格式