微型计算机原理及应用

下

微型计算机原理及其应用

下　册

南京航空学院

目　录

第七章　输入输出接口

本章主要介绍微型计算机I/O接口的功能及结构特点，计算机与外设之间进行数据传输的工作方式，以及Z80系列典型接口芯片的工作原理、编程方法及应用。

第一节　微型计算机接口的作用与特点

一、接口的结构与功能

接口是构成微型计算机系统最重要的部件之一。它是沟通CPU和外围设备之间的桥梁。见图7—1，微型计算机配有各种外围设备，如键盘，打字机，读带机，穿孔机，磁带机，模数转换器(ADC)和数模转换器(DAC)，调制解调器(MODEM)，CRT显示器以及软磁盘等等。接口的作用就是使CPU和外围设备能够协调地完成输入输出工作。外围设备为什么不能直接与CPU相连接进行工作呢？这是因为：①所使用外围设备的速度各不相同，有快速、慢速、中速之分，不可能与主机的工作速度相匹配。②微型计算机目前普遍采用总线方式与外围设备交换数据，即通过数据总线与外围设备进行数据交换。CPU输入或输出的数据往往取决本身的最小处理单位(常常是一个字节)，而且大都是并行传输的。外围设备对数据格式的要求却是各式各样的，例如有的A/D转换接口是12位的，有的要求串行传送等等。③外围设备的结构各不相同，有电子式，机械式，电磁式等等，使用的电路元件也有MOS器件与TTL器件之分。因此，它们的信号也要经过转换才能与CPU要求的信号相一致。

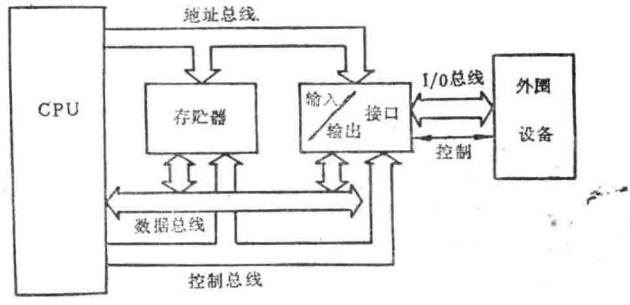


图7—1　输入/输出接口与CPU和外设的连接

总之，输入输出接口应该具有以下功能：

(1) 把外围设备送往CPU的信息转换为与计算机相容的格式。如串行转换为并行，配奇偶位等。

(2) 把CPU送往外围设备的信息转换为与外围设备相容的格式，例如并行转换为串行等等。

(3) 为计算机提供有关外围设备状态的信息，如设备“准备好”、“忙”或“闲”、缓冲器“满”或“空”等等。

(4) 协调CPU与外围设备在“定时”或数据处理速度上的差异。通常为使CPU与外围设备之间数据交换取得同步或速度匹配，常把传输数据存放在缓冲寄存器中，以便于等待或缓冲。

(5) 应有地址译码和设备选择能力。在有许多台外围设备的系统中，由输入输出接口提

供地址译码或确定设备码的功能，以选择其中的一台。对某些输入/输出设备，虽不执行数据传送，而是完成某种操作，例如磁带（或纸带）驱动装置，也需要有命令译码的功能。

(6) 应当有电平转换功能。微处理器，特别是绝大多数用 MOS 工艺制造的微处理器的输出电流或电压，不能与外围设备相匹配。必须由接口加以转换和匹配，才能工作。

(7) 外围接口电路还提供时序控制功能。有的接口电路具有自己的时钟发生器，以满足 CPU 和各种外设在时序方面的要求。

(8) 完善的外围接口电路具有可编程序能力。

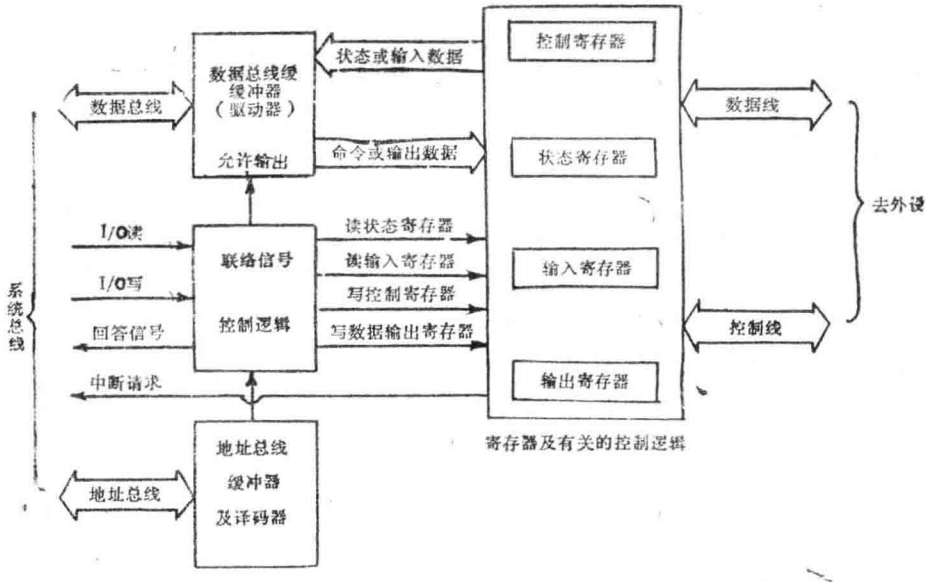


图7—2 典型接口的内部主要逻辑结构

典型外围接口的内部结构如图 7—2 所示。它主要包含有四种寄存器，即控制寄存器，状态寄存器，数据输入寄存器，数据输出寄存器。此外，还有数据总线缓冲器（驱动器）和地址总线缓存器，译码器，以及联络控制逻辑。它们的作用说明如下。

数据寄存器（缓冲器）：

包括数据输入寄存器和数据输出寄存器。输入寄存器的作用是将外设来的数据暂时存放一段时间，以便CPU将它取走。输出寄存器的作用是将CPU输出到外设的数据保存足够长的时间，以满足输出设备的需要。因为CPU送到数据总线上的数据，一般只能维持数百毫微秒（例如 8080 是 500ns）时间，对大多数外部设备来说，这段时间是不够长的。

控制寄存器：

它的作用是存放CPU发来的控制命令（控制字）和其它信息。这些控制命令的内容包括设置接口的工作方式，指定某些参数及功能，例如设定接口的某个通路作输入通路等等。控制寄存器只能写入。

状态寄存器：

它的作用是保存设备的现行状态信息，提供给CPU判断使用。例如输出数据准备好，

输出数据寄存器空，错误检测等等。状态寄存器只能读出。

以上介绍的数据寄存器、控制寄存器、状态寄存器，仅仅是接口中最主要的寄存器或缓冲器，它们是CPU与外部设备进行信息传输所必不可少的部分。通常所说的端口或口子(port)，数据口，控制口或状态口等，大都是指的这些寄存器。为了便于程序对它们进行访问，每个寄存器都分配给一个地址代码，即端口地址（或口地址）。典型的外设接口中除了上述的寄存器之外，还包括有数据总线和地址总线缓冲（或驱动）逻辑电路，以及实现CPU与接口、接口与外部设备之间通讯的联络控制电路，中断控制电路等等。一个功能很强的外围接口芯片，其内部电路的复杂程度往往不亚于CPU。但作为使用人员来说，主要关心的应当是与编写控制程序有关的那些部分，即所谓“程序模型”。

二、微型计算机接口的特点

与中小型计算机比较，微型计算机接口有以下两个特点：

(1) 通用性。在小型计算机系统中，对每种外设都配置一个相应的接口电路。如光电机接口，打字机接口，穿孔机接口等等。在微型计算机系统中，采用的是标准化的通用接口。也就是说，把功能复杂的接口电路制作在一块大规模集成电路芯片中，使一个接口芯片具有多种用途。同一个接口芯片，既可作为输入接口使用，又可作为输出接口用。它可以连接各种不同的外围设备而不必增加特殊的附加电路。因此，当微型计算机系统要增加新的外围设备时，不需要另行设计接口，只要选用已有的标准接口就行了。这就为系统的扩充带来很大的方便。

(2) 可编程序的工作方式。所谓可编程（programmable），指的是可以用编制程序的方法来指定接口的工作方式、功能、和工作状态，以适应各种不同外部设备所提出的要求。这意味着标准接口内部的线路结构具有各种形式，能适应各种不同工作方式的需要。它们的选择是通过CPU向控制端口送入方式字或命令字来控制的。例如指定接口工作在输入方式或输出方式、控制发送或接收操作的进行、向设备发出特定的控制信号、清除内部一些触发器或寄存器、判断信息传输中的错误并发出中断请求信号等。如果说小型计算机使用的中小规模集成电路输入/输出接口，是以硬件功能来分别满足各种外部设备的不同要求的话，那么微型计算机使用的大规模集成电路接口芯片，则是以增加软件功能为其特点。这并不意味着硬件功能削弱了，而只能说是随着标准接口硬件功能的加强，要求其软件功能也必须相应加强。从使用者的角度来看，接口电路硬件设计的工作量已大大减少，而灵活使用和控制标准接口的软件编程技术，就变得更加重要了。

目前微型计算机使用的大规模集成电路接口芯片，数目繁多，作用各异。但大体说来，可分为两大类：专用接口和通用接口。专用接口，即为某种用途或某类外围设备而专门设计的接口电路。目前这类接口芯片很多，例如专门为CRT显示器、软磁盘机，键盘等设备设计的接口有：

CRT显示器控制器

软磁盘控制器

键盘控制器

DMA控制器等等。

通用接口，即可供几种外部设备使用的标准接口。生产微处理器(CPU)的各个厂家，

一般都同时生产该系列的接口电路，以便配套使用。表 7—1 列出三种 8 位微处理器生产厂家所配备的两种标准接口（通用接口）。

表7—1　三种 8 位微处理器及其标准接口

公　司	主　机	并行 I/O 接口	串行I/O接口
Intel	8080	I 8255 并行I/O接口	I 8251 串行I/O接口
Motorola	MC 6800	MC 6820 可编程序接口适配器	MC 6850 异步通讯接口适配器
Zilog	Z80	Z80—PIO 并行 I/O接口	Z80—SIO 串行I/O接口

第二节　CPU与外设数据传送的方式

一、输入/输出接口与CPU 之间的连接

输入/输出接口电路与 CPU 进行数据传送时，CPU 如何与接口中的各种寄存器（即前面所说的端口）进行联系呢？一般说，CPU 对外设接口寻址通常有两种方式，即端口寻址的输入—输出指令方式和存贮器映象输入/输出方式。

1．端口寻址的输入/输出指令方式

在这种工作方式中，CPU 有专门的输入/输出指令和外设交换信息。指令中用地址来选择外设的各个端口，实际上就是选择接口中的各个寄存器，如数据寄存器，状态寄存器，命令寄存器等。在这种方式中，存贮器的地址空间和 I/O 接口的地址空间是相互独立的。通常存贮器的全部地址空间为 64K，而 I/O 接口的寻址空间为（0—256），两者互不影响。这种工作方式中，I/O 接口与 CPU 之间的联系仅能靠输入输出指令来沟通，操作是在 CPU 和 I/O 接口之间进行的。例如：

IN A，(n)

指令的功能是将设备号为 n 的接口中的数据寄存器内容送入累加器。

OUT (n)，A

指令的功能是将 CPU 中累加器的内容送至设备号为 n 的 I/O 接口寄存器中。

在这种工作方式中，CPU 对 I/O 接口的寻址是利用地址总线的低 8 位，寻址方法有两种。

(1) 全译码方法

将地址总线的低 8 位 A_7—A_0 经译码器译出 256 根线，每次只有一根线选中一个 I/O 接口，作为 I/O 接口芯片的片选信号。此法的缺点是需要较多的译码电路硬件。对于接口电路少的较小微机系统，可用更简便的方法。

(2) 线性选择法

它是在低位地址总线中，指定某一位作为某个 I/O 接口的选择信号。这样 8 条地址总线最多可以选择 8 个外设。通常 I/O 接口电路中有 (2—4) 个数据或控制寄存器，所以往往用地址总线最低 2 位（A_1 A_0），作为每个接口电路中数据或控制寄存器的选择信号，而用其余 6 根地址线（A_7—A_2）来选择设备（接口），如图 7—3 所示。

图中最低两位地址线（A_1 A_0）用来对接口中的 4 个寄存器译码，此译码电路已设置在接口芯片之中。其余 6 位（A_7—A_2）可以连接 6 个接口电路。若每一个接口芯片有 4 个输入/输出端口，则可得到 24 个端口地址。这种寻址方法仅适用接口电路很少的小型微机系统。

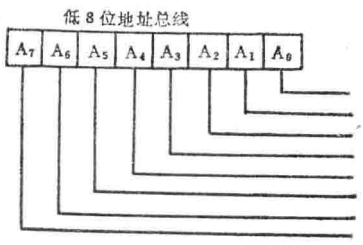


图7—3 接口电路线性选择法

2. 存贮器映象（Memory Mapped I/O）输入输出方式

这种方式又叫存贮器统一编址的寻址方式。在这种方式中，把一个外设作为存贮器的一个单元来对待，所以每个外设占有存贮器的一个地址。从外部设备向 CPU 输入一个数据，作为一次存贮器读的操作，而CPU 向外部设备输出一个数据，则作为一次存贮器写的操作。这样一来，对某个外部设备进行信息的输入或输出（通过接口）就相当于对某个存贮单元进行读或写一样。存贮器与输入输出接口所不同的，仅仅是地址不同。

这种连接方式的优点是可以简化系统的结构。因为接口的地址总线、数据总线、和控制总线可以和存贮器共用。在这种方式中，CPU 对外设的操作可使用全部存贮器指令（而不需要输入输出指令），所以编程灵活、方便。其缺点是外设地址要占去一部分内存单元，使内存容量减小。8 位微机中，M6800 采用这种寻址方式。

二、CPU 与外设之间的接口信号

外部设备通过接口电路与 CPU 交换信息，通常需要有这样一些信号，即数据，状态信息，控制信息（或控制命令）。

1. 数据（Data）

在微型机中，数据通常是 8 位或16位。它可大致分为三种基本类型：数字量，开关量和模拟量。

(1) 数字量。由键盘、光电输入机、卡片机等输入的信息，是以二进制形式表示的数或以 ASCII 码表示的数或字符。

(2) 模拟量。计算机用于控制时，大量的现场信息经过传感器把非电量（例如温度、压力、流量、位移等）转换为电量，经过放大即得到模拟电压或电流。这些模拟量必须先经过模拟/数字（A/D）转换之后，才能输入到计算机；计算机的控制输出大部份也必须经过数字/模拟（D/A）转换方能去控制执行机构。

(3) 开关量。指的是一些两个状态的量。如电机的运转与停止、开关的闭合与断开，阀门的打开与关闭等。这些量只要用一位二进制数即可表示。例如用二进制数 1 表示阀门打开；用 0 表示阀门关闭。所以字长 8 位的微机一次输入或输出可控制 8 个这样的开关量。

2. 状态信息（Status）

指的是表示外部设备接口当前状态的信息。例如，输入时，输入装置的信息是否准备好（Ready）；输出时，输出装置是否有空（Empty），若输出设备正在输出信息，则用Busy（忙）表示，等等。

3. 控制信息（Control）

通常指的是CPU向输入输出设备发出的控制命令。例如控制输入输出设备启动或停止的信号等。

三、CPU与外设数据传送的方式

CPU与外部设备之间进行数据传输，通常有四种传送方式，即同步方式，查询方式，中断方式和直接数据通道（DMA）传送方式。下面仅对这几种传送方式的特点加以说明。

1. 同步方式

又称无条件传送方式。这种方式只有当外部控制过程的各种动作时间是固定的，且是在已知的条件下才能使用。在传送信息时，由于能算出外部设备处于准备好状态的时间，所以不必询问外设的状态。这样，在输入时只需发出输入指令IN，而在输出时发出输出指令OUT即可。这种传送方式的优点是程序简单。但必须确信外设是处于准备好状态的情况下才能用。否则就容易出错。

2. 查询（Polling）传送方式

又称异步传送方式，或条件传送方式。其特点是在传送信息前必须查询一下外设的工作状态，当外设是处于准备好状态时才传送；如果外设未准备好，CPU就反复查询或执行其它一段程序后再查询，一直等待到外设处于准备好状态为止。

查询输入方式的流程图如图7—4所示。

图7—4 查询式输入程序流程图

设用状态寄存器的D_7位表示某外设的状态，当$D_7=1$时，表示该外设已准备好，可以向CPU输入数据；$D_7=0$表示未准备好，不能输入数据。设状态寄存器的端口文字地址为STATUS；DATA是8位数据寄存器的的端口文字地址，则查询输入部分的源程序如下：

```
        ⋮
TEST: IN A, (STATUS)      ; 读入状态信息
      AND 80H             ; D7=1, 准备好?
      JR Z, TEST          ; D7=0, 未准备好,CPU 反复查询
      IN A, (DATA)        ; D7=1, 数据读入累加器
        ⋮
```

同样，当CPU向外设输出数据时，也要先判断外设的状态，如果外设正在进行输出操作（处于“忙”状态）或者外设接口中的数据寄存器已经“满”了，则CPU要等待，只有当外设接口为“空”状态时，CPU才执行输出指令。查询式输出的程序流程图如图7—5所示。查询输出部分的源程序如下：

```
        ……
TEST:  IN A, (STATUS)      ; 读入状态信息
       BIT  7, A           ; D7=0? 外设空?
       JR NZ, TEST         ; 不空，等待
       LD A, (STORE)       ; 数据取入累加器
       OUT  (DATA), A      ; 输出数据
        ……
```

3. 中断传送方式

在查询传送方式中，CPU 要不断地询问外设，当外设没有准备好时，CPU 要进行等待不能干别的工作，这样就浪费了CPU的时间。特别是对有些慢速的外设，如键盘，打印机等，CPU 要花费很多时间进行等待。当然，可以让 CPU 运行一段程序后再查询，但这无疑给程序设计者带来不便。为此，可采用前面介绍的中断方式传送信息。例如在输入时，若外设的数据已存入输入寄存器(准备好)，就可向 CPU 发出中断请求，使 CPU 响应中断转去执行为外设服务的输入操作，待输入操作完成后才返回去继续执行原来的程序。关于中断处理，前面第六章已有介绍，不再重复。

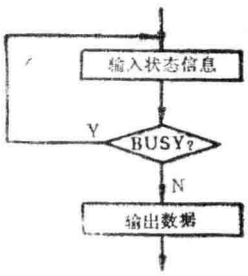


图7—5 查询式输出程序流程图

4. 直接数据通道传送（DMA）方式

利用上述的查询方式或中断方式传送信息都要多占用 CPU 的工作时间。例如某个外部设备 1 秒钟能传送 100 个字节，若用查询方式输入，则在这 1 秒钟内 CPU 全部用于查询和传送而不能再干别的工作；若用中断方式，设 CPU 每传送一个字节的服务程序是 100 微秒，则传送 100 个字节，CPU 只需用 10 毫秒，即只占 1 秒的 1/100，其它 99/100 的时间仍可用来执行主程序。但是以中断方式传送数据是由 CPU 来控制并通过累加器来实现的。也就是说，外部设备要与存贮器交换信息，必须经过累加器，而且送数和取数都是由 CPU 执行相应的指令进行软件控制完成的。在中断方式中，每次传送数据都要执行保存断点、保护现场、程序返回等许多条指令。因此，通常传送一个字节需要几十到几百微秒。这对于高速的 I/O 设备，以及成组交换数据的情况来说（例如磁盘与内存之间交换数据）就显得太慢了。为此，希望用硬件实现在外设与存贮器之间直接交换数据(即直接存贮器访问 DMA)。在 DMA 方式中，数据的传输是由控制器来实现控制的。也就是说，CPU 不必按取指令并译码的方式来完成数据的传输操作。DMA 方式的数据传输实际上只受存贮器存取时间快慢的限制。用硬件代替软件控制的过程，就意味着加快数据传送的过程。加上在 DMA 方式的数据传送过程中由于不用经过 CPU（累加器）和不受 CPU 的干预而在外围设备和存贮器之间直接进行数据交换，也加快了传送速度。其代价就是要增加比较复杂的硬件电路，即 DMA 控制器。已知在正常情况下，系统的地址和数据总线以及一些控制信号线(例如 $\overline{MREQ}$，$\overline{IORQ}$，$\overline{RD}$，$\overline{WR}$等）是由 CPU 控制管理的。但在进行 DMA 操作时，要求 CPU 把这些总线让出来（即 CPU 连到这些总线上的信号线处于第三状态—高阻抗状态），而由 DMA 控制器接管。

由此可见，DMA 控制器不仅要在必要时接管系统总线，而且还要能控制执行数据交换。所以它必须能发出地址信号，修改地址指针，控制传送的字节数，判断 DMA 是否结束，以及发出 DMA 结束的信号等。所有这些操作都要由硬件来实现。

一般地说，执行 DMA 数据传送的步骤如下：

(1) 外围设备通过其相应的 DMA 控制器，经 DMA 请求线向 CPU 发出 DMA 请求。对 Z80 CPU 来说，就是向 CPU 发出总线请求（$\overline{\text{BUSRQ}}$）信号。

(2) CPU 在接受到 DMA 请求信号之后，通过 DMA 响应线向 DMA 控制器给出“DMA 响应”的回答信号。对 Z80 CPU 来说，就是送出总线响应（$\overline{\text{BUSAK}}$）信号，然后由 DMA 控制器及相应的外围设备来驱动和控制系统总线，进入 DMA 工作方式。

(3) DMA 控制器把“DMA 响应”信号送到外围设备和存贮器，发出可传送数据的通知。

(4) 在接到 DMA 响应信号后，DMA 控制器把自己的逻辑电路接入系统总线，并在地址总线上发出所要访问的存贮器起始地址；在读/写控制线上发出“读出”或“写入”命令，开始实现 DMA 传送过程。

(5) 每传送一个字节的数据，使 DMA 控制器中的地址寄存器加 1，从而指明下次要传送数据的地址；同时使字节计数器减 1，如此进行下去，直至字节计数器的值减到零为止，说明数据已传送完毕，用这个零信号产生一个 DMA 结束信号，使 CPU 恢复正常的工作。

通常 DMA 传送的原理框图和相应的流程如图 7—6 和图 7—7 所示。

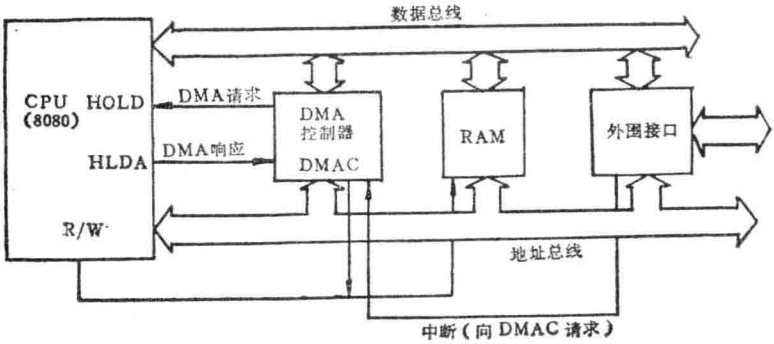


图7—6 DMA 传送原理框图

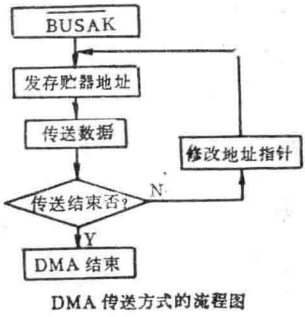


图7—7 DMA 送方式的流程图

第三节 计数器定时器电路（Z80-CTC）

一、结构特点

计数器/定时器电路 (Counter Timer Circuit，简称 CTC)，是一个具有四个独立通道的可编程序芯片。它具有定时和对外界事件计数的功能。Z80—CTC采用N 沟道硅栅耗尽型负载工艺、28 条引脚双列直插式封装。所有输入与输出电平都与 TTL 电路兼容。通过编程的办法，可以把 CTC 每个通道安排为计数器工作方式或定时器工作方式。所谓计数器

工作方式，是指CTC能对外界事件进行计数，当计数达到程序规定的数值时，便向CPU请求中断。而定时器工作方式，是指CTC能定时地发出脉冲并同时请求中断。和Z80系列的其它芯片一样，不必增加外部电路CTC就能形成中断优先级顺序链排队电路。CTC中的每个通道都可以设定自己的中断向量，并规定0号通道的中断优先级最高，3号通道优先级最低。

图7—8是Z80-CTC的内部结构框图。图7—9是它每个通道的电路结构框图。

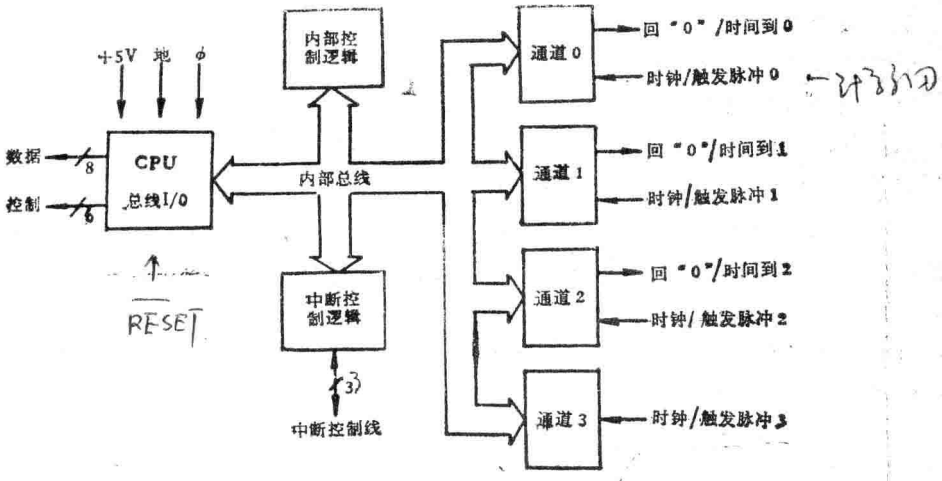


图7—8　Z80—CTC内部结构框图

图中每个通道包括两个寄存器（通道控制寄存器和时间常数寄存器）和两个计数器（减1计数器和定标器）。0号通道还有一个中断向量寄存器。它们的功能说明如下：

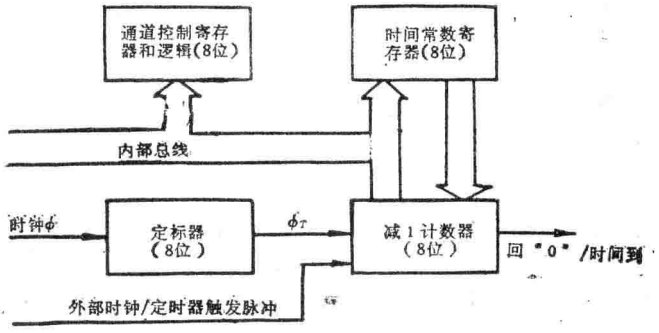


图7—9　Z80—CTC通道的框图

1．通道控制寄存器

CTC的每个通道都有一个8位的通道控制寄存器，它用来存放CPU发来的工作方式控制命令，即控制字，以指定该通道的工作方式（定时器方式还是计数器方式）及参数。

2．定标器

定标器实际上是一个8位的分频器。它只有在定时器工作方式时才使用，按照通道控制字的规定，对其输入信号(即系统时钟φ)进行16分频或256分频。它的输出作为减1计数器的输入。

3．时间常数寄存器

这是一个8位寄存器。它可以在定时器方式下工作，也可以在计数器方式下工作。它用来存放计数的次数或预定的时间值。只要不对它写入新的内容，其数值就一直保持不变。

4．减1计数器

8位计数器。可以在计数器方式或定时器方式下工作。该计数器由时间常数寄存器预置初值，以后每当减1计数器到“0”时，由时间常数寄存器自动地重新装入初值。当CTC以计数器方式工作时，外部时钟脉冲边沿使减1计数器减1，当CTC以定时器方式工作时，则通过定标器的输出脉冲使减1计数器递减。换言之，此时每来16个或256个（由程序规定）Z80时钟脉冲，就使减1计数器减1。在任何时刻，CPU可以通过选择CTC通道的端口地址，用简单的I/O读指令来读取这个计数器的内容。每当计数器计到“0”时，便在CTC的ZC/TO（计数回零/时间到）输出端产生一个正脉冲，同时向CPU发出中断请求信号。注意，由于受引脚数目的限制，通道3没有ZC/TO引出端。因而通道3只可用于不需要这一输出脉冲的场合。

此外，CTC的0号通道比其它通道还多了一个中断向量寄存器，它向CPU提供低8位的中断向量地址。其内容是由程序员设定的。

二、引脚功能说明

Z80 CTC的引脚配置如图7—10所示。引脚的功能说明如下。

1. D_7—D_0：数据总线。双向，三态。用于在CPU和CTC之间传送数据及命令信息。

2. CS_1，CS_0：通道选择(输入，高电平有效)。通过对CS_1和CS_0的组合，可以选择四个通道之一。通常CS_1可以和CPU的地址总线A_1相连，CS_0和A_0相连，其真值表见表7—2。

表7—2　通道选择真值表

	CS_1	CS_0
通　道　0	0	0
通　道　1	0	1
通　道　2	1	0
通　道　3	1	1

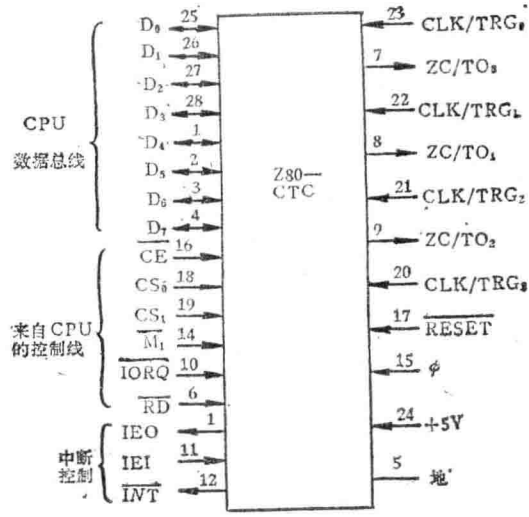


图7—10　Z80—CTC引脚图

3. $\overline{CE}$：片选端（输入，低电平有效）。当$\overline{CE}$为低电平时，才允许CTC与CPU之间进行信息的传输。一般用地址总线A_7—A_2译码后所取到的低电平信号接到CTC的$\overline{CE}$信号端。$\overline{CE}$与CS_1、CS_0相配合，以实现对指定的CTC中四个通道的选择。

4. ϕ：时钟（输入），从Z80 CPU来的系统时钟信号。CTC利用它实现与某些信号的内同步。

5. $\overline{M_1}$：来自 CPU 的取指令操作码周期信号（输入，低电平有效）。

6. $\overline{IORQ}$：来自 CPU 的输入/输出请求信号，低电平有效。它和$\overline{M_1}$信号配合构成中断响应信号。

7. $\overline{RD}$：来自 CPU 的读周期信号，低电平有效，以上三个信号在不同状态下的作用见表7—3。

表7—3 $\overline{M_1}$，$\overline{IORQ}$，$\overline{RD}$ 的功能

$\overline{M_1}$	$\overline{IORQ}$	$\overline{RD}$	功　　能
0	0	1	中断响应
0	1	0	检查中断服务程序的终止
1	0	0	读 CTC，将减 1 计数器的内容读出送至 CPU
1	0	1	写 CTC，将计数值或计时值、中断向量、控制字写入 CTC

8. IEI：中断允许输入（输入、高电平有效）。当有多个外设接口与 CPU 相连接时，用这个信号和 IEO（中断允许输出）信号一起，建立串行中断优先级顺序链。当这个引脚为高电平时，表明 CPU 没有为优先级高于本芯片的器件进行中断服务。此时本接口电路可向 CPU 请求中断。

9. IEO：中断允许输出（输出，高电平有效）。和 IEI 信号一起，建立中断优先级顺序链。只有当 IEI 为高电平，且 CPU 没有为本芯片进行中断服务时，IEO 信号才为高电平。若CPU 正在为本芯片中断服务或者为优先权比本芯片高的接口服务时（使 IEI 为低），IEO 信号均为低电平，从而阻止优先权较低的外设接口向 CPU 申请中断。

10. $\overline{INT}$：中断请求（输出、漏极开路，低电平有效）。任何一个已经编好程序的通道，在允许中断的情况下，如其减 1 计数器计到零，$\overline{INT}$ 就变为低电平，输出中断请求信号。

11. $\overline{RESET}$：复位信号，低电平有效。这个信号停止所有通道的工作，并使各控制寄存器中的通道中断允许位复位，从而禁止 CTC 产生中断请求，ZC/TO 和 $\overline{INT}$ 输出皆变为无效状态。IEO 和 IEI 变为相同状态，CTC 的D_7-D_0端输出全部变为高阻抗状态。

12. CLK/TRG 0—CLK/TRG3：外部时钟/定时器触发脉冲。这是通道 0 到通道 3 的外接时钟脉冲或外接计时开始的触发输入脉冲。究竟是用上升沿还是下降沿起触发作用，可由程序设定。

13. ZC/TO(0)—ZC/TO(2)：计数回零/时间到信号（输出，高电平有效）。当 CTC 通道的减 1 计数器减至 0 时，该引脚输出一个正跳变脉冲，当重新开始计数时，又恢复为低电平。这些信号可向外部电路提供精度较高和宽度一定的周期方波。

三、CTC 的工作方式

刚一接通电源时，CTC 的状态不确定。用复位信号$\overline{\text{RESET}}$将 CTC 置成某一个已知状态。在任何一个通道可以开始计数或定时之前，通道控制字和时间常数数据字必须先写入该通道中相应的寄存器中。如果某通道按程序安排是允许中断的，则必须向该通道的中断控制逻辑写入一个中断向量字。当这些指定 CTC 各通道功能和参数的控制字都写入CTC后，则相应的通道就按照程序设计的安排，进行工作。

1. 计数器方式

这种工作方式主要用来对 CLK/TRG 输入信号的边沿（由程序规定为上升边沿或下降边沿）进行计数（见图 7—11）。开始计数时，这个通道的控制逻辑检测通道外部来的一系列外部时钟/定时器触发脉冲(CLK/TRG)的边沿，在每一个触发脉冲的边沿后的第一个系统时钟（ϕ）的上升沿使减 1 计数器减 1。虽然对外部时钟脉冲触发边沿和 ϕ（时钟）上升沿之间，不要求严格的时间关系，但是直到紧接着的 ϕ 脉冲到来之前减 1 计数器将不进行减量。总之，每来一个外部触发脉冲 CLK/TRG，减 1 计数器就自动减 1，当减 1 计数器减到“0”时，通道的 ZC/TO 端输出一个正脉冲信号，表明一个计数周期已经结束。如果这时通道允许中断，则向 CPU 发出中断请求信号 $\overline{\text{INT}}$。此后，时间常数寄存器再次把计数值装入减 1 计数器，重复上述的计数操作。在计数到“0”以前，如果需要对时间常数寄存器写入新的内容，电路只是把新常数放入时间常数寄存器中，这样新的计数值必须等减 1 计数器完成当前的计数过程后，才能按新的时间常数计数。计数器工作方式的时序波形如图7—12所示。图中是对外部时钟/定时器触发脉冲（CLK/TRG）的上升沿计数。

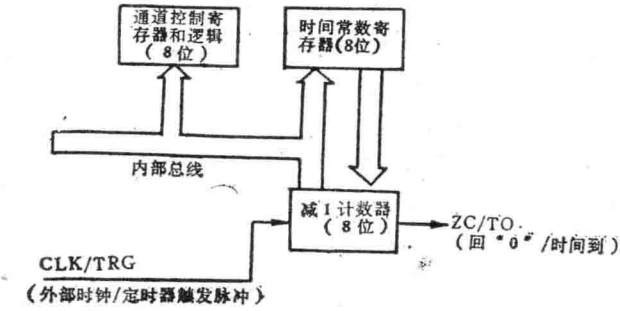


图7—11 CTC 通道计数器工作方式

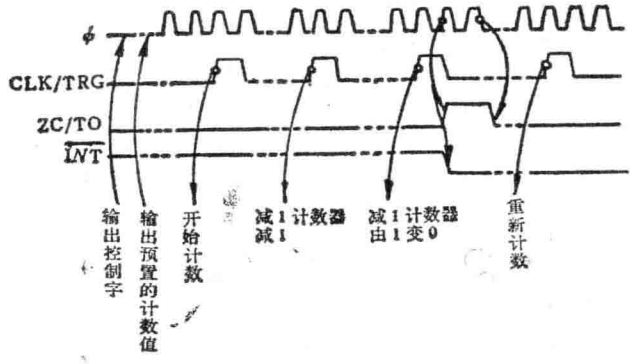


图7—12 计数器工作方式的时序波形

2. 定时器工作方式

使用这种方式工作时，CTC 的 ZC/TO（回零/时间到）端定时地发出脉冲，即得到以系统时钟为基础的指定宽度的信号。开始工作时，先要把时间常数寄存器中计时值装入减1计数器。系统时钟 ϕ 由定标器进行16或256分频（由程序规定）之后，输入到减1计数器。每来一个分频后的输入脉冲，减1计数器就自动减1；减1计数器减至零时，一个计时周期就结束了。这时在 ZC/TO 端输出一个正脉冲。显然 ZC/TO 脉冲和减1计数器的输入脉冲以及时间常数之间的关系是：

$$T_{ZC/TO}=t_C \cdot P \cdot T_c$$

其中 t_C 是系统时钟周期，P 是定标系数16或256，T_C 是设定的时间常数。$T_{ZC/TO}$ 是 ZC/TO 的脉冲周期。当减1计数器递减至零时，如果通道处于允许中断状态，则向CPU发出中断请求信号 $\overline{INT}$。此后，时间常数寄存器的内容重新自动装入减1计数器。CTC 再次开始上述的计时工作。

定时器工作的启动方法主要有二种：一种是由程序启动；另一种是由外部触发脉冲 CLK/TRG 的边沿(升沿或降沿，由程序设定)来启动。程序启动就是由系统时钟自动启动。当时间常数写入通道之后，定时器便在紧接着的下一个时钟周期起开始计时（见图7—13）。从图中可以看出，定时器在计时值装入时间常数寄存器时刻开始工作，每当有16个系统时钟脉冲（设定为16分频时）送到减1计数器时，减1计数器就自动减1。如果是由外部触发脉冲 CLK/TRG 的边沿（升沿或降沿）来启动定时器工作，则当时间常数装入时间常数寄存器之后，通道便等待外部触发脉冲 CLK/TRG 的到来。一旦出现CLK/TRG信号，在下个时钟信号 ϕ 的上升边时刻定时器便开始计数。其时序波形如图7—14所示。图中定标器的分频系数设定为256，用触发脉冲CLK/TRG的上升沿启动定时器工作。也可将

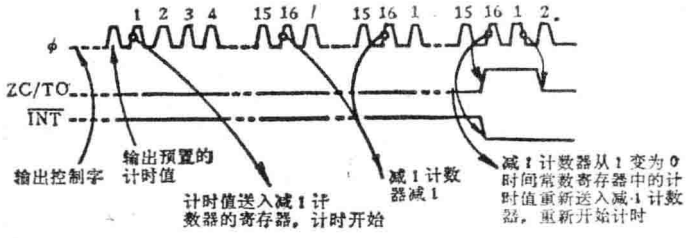


图7—13 CTC 定时器工作方式的时序波形（由程序启动定时器工作）

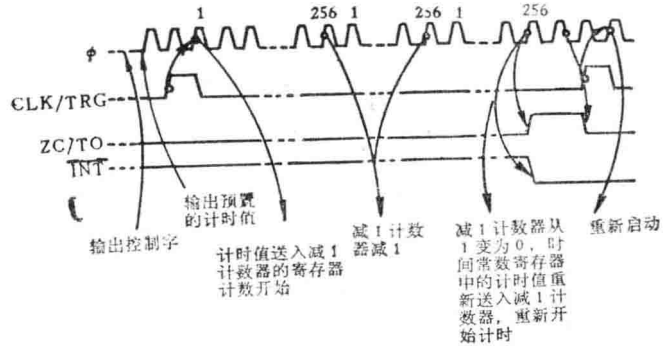


图7—14 CTC 定时器工作方式的时序波形(由外部触发脉冲 CLK/TRG 启动定时器工作)

触发脉冲改为负脉冲，用它的下降沿来启动定时器工作。

四、CTC 的编程

对 CTC 进行程序设计，就是编写控制 CTC 按不同方式进行工作的程序。比如规定通道按计数器方式还是按定时器方式工作，给定时间常数，定标系数等参数。

具体地说，CTC 的程序设计就是设定几个控制字的内容，其步骤如下：

(1) 首先要确定中断服务程序入口地址表在存贮器中的存放位置，以设定 CPU 中断向量寄存器 I 的值（即向量表地址指针的高 8 位字节）。

(2) 设定中断向量控制字，即中断向量低 8 位，以便中断响应后，能和 I 寄存器配合找到入口地址表。

(3) 设定通道工作方式控制字，以指定 CTC 的工作方式和内部相应逻辑。

(4) 设定时间常数控制字，以确定定时器方式时的时间常数和计数器方式时的计数值。

下面说明各个控制字的格式和编程的方法。

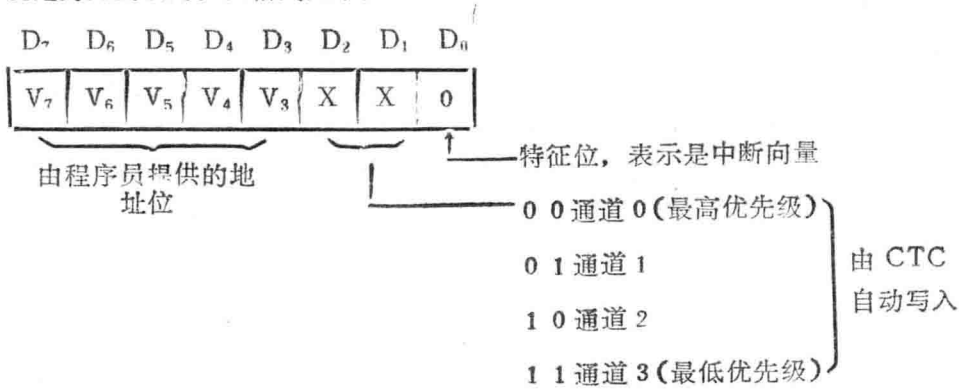
1. 设定 CPU 中断向量寄存器 I

假定中断向量地址表指针为 2400H，这时应把 24H 装入 I 寄存器，完成这一工作的源程序如下：

```
LD  A,  24H
LD  I,  A
```

2. 中断向量控制字

CTC 按照 Z80 CPU 的中断方式 2 进行中断处理。因此，每当 CTC 的某个通道向 CPU 请求中断并被响应时，都必须向 CPU 提供一个中断向量表地址指针的低 8 位字节。中断向量控制字就是为此而设的。其格式如下：



其中 D_0 位是特征位，为了使 CTC 认出这是一个中断向量字节，D_0位必须是 0 。D_7～D_3位由程序员设定。D_2 D_1 位则是由 CTC 的中断控制逻辑自动提供的。当某个请求中断的通道要把中断向量放到 CPU 的数据总线上时，CTC 的中断控制逻辑自动给出 D_2 D_1 位的数值，以便指明是哪一个通道要求中断服务。

中断向量控制字在 CTC 初始化时，由 CPU 通过输出指令送到通道 0 。应当指出的是，中断向量控制字只能送到通道 0，因为只有通道0内部才有中断向量寄存器。这在编程时要注意。由于中断控制逻辑能自动工作，所以编程时只要送入一个通道的中断向量后，其它通道