计算机接口技术

于英民　孙全　莫玮　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

计算机接口技术

于英民　孙全　莫玮　编著

電子工業出版社

内 容 简 介

本书全面阐述计算机的接口技术。全书共分 10 章。1～2 章是接口技术的基础部分;3～7 章是芯片级接口技术部分,即并行接口、串行接口、定时器接口和模拟器件接口;8～10 章是系统级接口技术的内容。

本书的特点是突出原理、兼顾硬软件并结合实际。1～2 章的基础部分是同类书中所没有的,或少有的。8～10 章也是本书有特色的部分,因为接口技术向系统级和商业机的扩充发展,需要相应发展系统级的设计技术。本书内容的安排,从始至终贯彻既为以单片机为核心的专用小系统设计打基础,也为以商品机 PC 为基础的系统扩充服务。在芯片级接口技术部分,本书安排有并行接口标准和串行通信差错检验等实用性较强问题的介绍。

本书是科技书,也是一本好教材,在多数章后附有思考题和习题,适合计算机应用、通信与系统、仪表与测量、监测与控制等学科的科技工作者、本科生和研究生使用。

计算机接口技术

于英民　孙全　莫玮　编著

责任编辑:张荣琴

*

电子工业出版社出版(北京市万寿路)

电子工业出版社发行　各地新华书店经销

北京科技印刷厂印刷

*

开本:787×1092 毫米　1/16　印张:20.75　字数:540 千字

1996 年 6 月第 1 版 1997 年 5 月第 2 次印刷

印数:5000－10000 册　定价:27.00 元

ISBN 7-5053-3503-0/TP·1404

前　言

本书的第1版名为《接口技术》,是按电子工业部工科电子类专业教材1986～1990年编审出版规划,由无线电技术与信息系统教材编审委员会仪表与测量编审小组征稿、评选、推荐出版的。本书的对象是宽口径的,适合计算机应用、通信与系统、仪表与测量、信号与电路、监测与控制的研究生和本科生。本书的取材主要来自我们的教学和科研工作,所以,内容结合实际也考虑学科发展的需要,对正在从事接口技术的科技人员也是一本有益的科技参考书。

经过几年的应用,普遍反映较好。1992年还荣获电子工业部电子类专业优秀教材二等奖。因为第1版作为教材发行,发行数量有限,未能满足社会的需求。另一方面,因为计算机技术飞速发展,虽然才经历了短短几年时间,第1版已不完全符合发展的要求。为了满足广大读者需要,进行修编,作为科技书出版,并改名为《计算机接口技术》。修编版虽是科技书,也不失作为教材的特色,也是一本好教材。本书有如下特点:

1. 全面阐述计算机中的接口技术

本书内容比较全面,包含了接口技术的各个环节,共有10章:

(1) 计算机基本接口原理;
(2) 微处理器系统和微计算机系统总线;
(3) 并行接口;
(4) 标准并行接口(打印机接口和GPIB接口);
(5) 串行接口(异步通信接口、通信差错检验和同步通信);
(6) 定时器接口及CRT显示器接口;
(7) 模拟器件接口;
(8) 信号处理器接口和多微处理器系统;
(9) 底板总线;
(10) 接口软件设计。

2. 结合实际,具有实用性

当前计算机接口技术主要要解决的问题有两类:一是以单片微机为核心的专用小系统设计,另一则是以商品机PC/XT/AT为基础的系统扩充。本次修编的重点就是要解决这两类问题。为此,在修编版中增加了"微处理器系统和微计算机系统总线",作为第2章,介绍单片微机和PC/XT/AT微机的扩充总线、总线时序及其系统特性,起承上启下的作用。在以后的章节中又以单片机和PC/XT/AT机为对象进行举例和设计等。

3. 突出原理,兼顾硬软件

计算机技术高度发展的今天,普遍认为,要开发一个系统,接口技术是重要的。但是,接口技术方面的新著作越来越少见。原因有二:一是原来以介绍硬件接口电路设计为主的接口技术已不能满足当前的需要,因为软件的比重加强了;二是系统机扩展的工作增多了,虽有多种PC级的硬件组成介绍,但缺少在系统机上进行接口的原理方面的参考资料。本书

第1版编写时曾拟定了“突出计算机接口原理、兼顾硬软件”的原则，当我们写修订版时更感此原则的重要性。在这一原则的指导下，结合我们的科研和教学，完成了本次修订版。本修订版，第1章以总线为中心讲述计算机的基本接口原理；第2章讲述单片微机和系统机PC/XT/AT机的总线、时序和特性等；第3章～第7章讲述各类接口的原理、编程和设计举例；第10章又专门讲述接口软件设计，特别是较全面地介绍了新发展起来的C语言接口软件技术。

4.适应面广

本书内容全面、丰富，应用面广。作为一般微机接口技术教学，可选用1～7章的基本内容。考虑这一应用，每章后面附有思考题和习题。当作为科技工作者的参考书时，1～10章都是需要的，特别是8～10章，因为同类书中这方面的内容很少。考虑使用方便和自学的需要，本次修编又在相应章节中增加了基础部分，并在书后安排了多种附录。

电子科技大学张世箕教授和北方交通大学蒋焕文教授领导的编审小组对本书修编提出了方向性的意见，并对具体工作也给予很大帮助；承蒙蒋焕文教授和王化深副教授主审，提出许多极宝贵的意见，在此表示衷心地感谢。

由于作者水平所限，书中错误和疏漏之处在所难免，恳请读者批评指正。

作者

1995年6月

第 1 章　计算机基本接口原理

接口技术可以用 3 种方法进行讲述:(1)以 CPU 为中心,(2)以接口芯片为中心,(3)以总线为中心。本书选择以总线为中心的方法，把接口技术中的共性问题作重点,忽略具体 CPU 或接口芯片的细节,目的是为读者打下良好的理论基础，以适应今天计算机飞速发展和多种集成电路应时而生的形势。

1.1　计算机的总线结构

计算机系统是由中央处理器(以下简称 CPU)、存储器和输入输出系统三大功能模块组成。在发展的初期,是面向 CPU 各模块之间点对点的直接连接。由于集成电路的发展,从 70 年代开始采用以总线为中心的标准结构(总线的定义见下节),图 1.1 给出典型的三总线结构。这是小型机和微型机的典型结构，且已形成多种总线标准。总线结构有如下优点:

(1)简化了硬、软件的设计。由于总线是严格定义的,所以这种结构只需将具体的 CPU 板、存储器板和 I/O 接口板，以插件的形式挂接上总线,并辅以软件即可工作。由于插件在系统中仅仅与总线打交道,因而使硬件的设计和调试简单,节省工时;软件也容易规范化,因而软件成本也降低了。

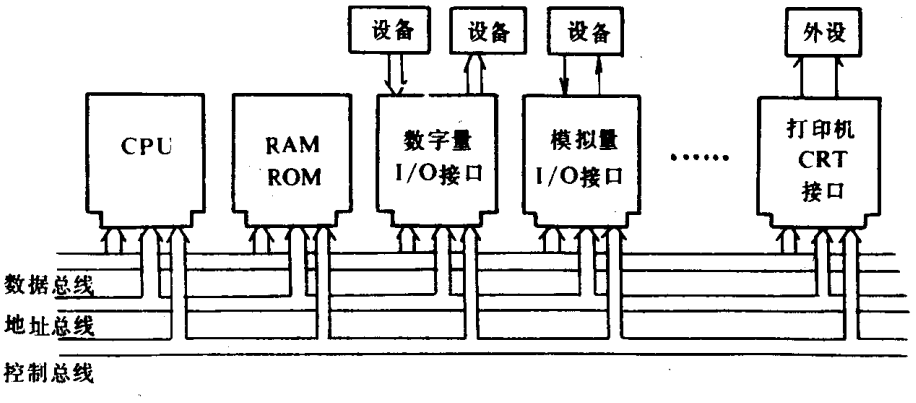

图 1.1　系统的总线结构形式

(2)简化了系统结构。这种系统节省连线,清晰明了,插座和总线甚至可采用印制板工艺,做成一块总线底板,这样,制造、安装都可简化。

(3)便于系统的扩充和更新。这种系统只需多准备插座,便可扩充;更新就更容易，换插板(带插头的印制板)即可。

在本章讲基本原理时我们选择 Z-80 总线作代表。Z-80 总线是前一段时期流行的一种工业标准,虽然现在应用已减少,但是它比较简单,适合作入门的总线模型。所谓 Z-80 总线就是以 Z-80CPU 的引脚信号为定义的总线，因为它的引脚信号没有复用功能,所以是典型

的三总线形式。第2章我们再讲当前更实用的总线，但是也复杂得多了。图1.2给出Z-80CPU的引脚信号，它有8条数据线，16条地址线和14条控制线(省略电源线)。数据总线传送数据，它是双向的，可输出也可输入。地址总线传送地址码，是单向的。控制总线有的输出，也有的输入，每条线有自己的功能，比较复杂。现叙述各控制引脚(线)的功能如下：

图1.2示出Z-80微处理器芯片的引脚信号图，为了突出重点，数据总线和地址总线采用了简化标注方法，只有控制线逐条给出。

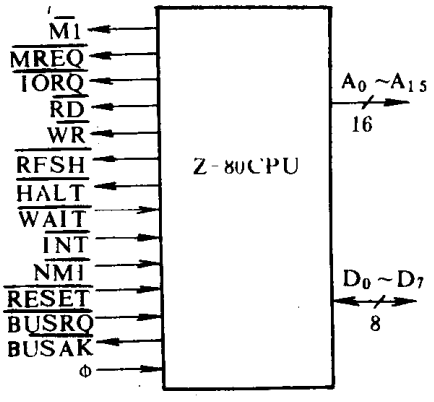

图1.2 Z-80 CPU引脚信号

$\overline{M1}$(机器周期1)信号低电平有效，表示一条指令的执行处于取指周期期间。

$\overline{MREQ}$(存储器请求)信号低电平有效，表示CPU要读或写存储器，此刻地址总线上已存在用于存储器读写的有效地址码(见图1.19)。

$\overline{IORQ}$(输入/输出请求)信号低电平有效，表示CPU要读或写外部设备(I/O端口)，此刻地址线低8位上已存在要读写I/O端口的有效地址码(参见图1.12)。另外，Z-80以$\overline{IORQ}$和$\overline{M1}$同时有效作为响应中断请求的标志(参见图1.18)。

$\overline{RD}$(读)信号低电平有效，表示CPU要从存储器或I/O端口读数据。考虑与$\overline{IORQ}$或$\overline{MREQ}$的配合作用，CPU读I/O端口的控制信号是$\overline{IOR} = \overline{RD} + \overline{IORQ}$；CPU读存储器的控制信号是$\overline{MEMR} = \overline{MERQ} + \overline{RD}$。

$\overline{WR}$(写)信号低电平有效，表示CPU要向存储器或I/O端口写数据。CPU控制传送数据的过程，就是数据读、写的过程，所以，$\overline{RD}$和$\overline{WR}$变化最频繁。考虑与$\overline{IORQ}$或$\overline{MREQ}$的配合作用，CPU写I/O端口的控制信号是$\overline{IOW} = \overline{WR} + \overline{IORQ}$；CPU写存储器的控制信号是$\overline{MEMW} = \overline{MERQ} + \overline{WR}$。

$\overline{RFSH}$(刷新)低电平有效。读写存储器分静态和动态两种。使用动态存储器时，由于电容性存储单元放电，信号要随之消失，为此必须不断给电容充电，以维持信号电平，这就叫做刷新。Z-80CPU的优点之一是自备刷新逻辑电路。在$\overline{MREQ}$和$\overline{RFSH}$都为低电平时，地址总线的低7位($A_0 \sim A_6$)给出动态存储器的刷新地址，用以刷新动态存储器。

$\overline{HALT}$(暂停)低电平有效。在计算机执行一条HALT指令后，HALT输出低电平，指示CPU进入暂停状态。所谓暂停状态，实际是连续执行NOP指令(空操作指令)，不中断对动态存储器的刷新，一直要到CPU接受非屏蔽或屏蔽中断操作后才结束。

$\overline{WAIT}$(等待)，输入，低电平有效。$\overline{WAIT}$告诉CPU所寻址的存储器或I/O端口，尚未准备好交换数据，只要这个信号有效，CPU继续插入等待周期。这个信号能使CPU与任何慢速的I/O或存储器配合工作。

$\overline{INT}$(可屏蔽中断请求，常简称中断请求)，输入，低电平有效。此线是供外设向CPU发中断请求用。CPU是否响应外设的中断请求，决定Z-80内部的中断允许触发器IFF的状态和是否存在总线请求。若内部IFF在开中断状态，又无总线请求，则响应中断请求($\overline{M1}$和$\overline{IORQ}$变低)。

$\overline{NMI}$(非屏蔽中断请求)，输入，负边沿触发。非屏蔽中断请求的中断优先级高于$\overline{INT}$，但

仍不如总线请求$\overline{\text{BUSRQ}}$。

$\overline{\text{RESET}}$(复位),输入,低电平有效。复位信号迫使 PC 为零,且初始化 CPU。初始化包括以下内容:

(1) 中断允许触发器 IFF_1 和 IFF_2 清零;

(2) I 寄存器置 00H;

(3) R 寄存器置 00H;

(4) 置中断方式 0。

在复位期间,地址总线和数据总线处于高阻状态,且所有控制信号无效,也不产生刷新。

$\overline{\text{BUSRQ}}$(总线请求),输入,低电平有效,由要求使用微处理系统数据总线和地址总线的控制器使用,例如 DMA 控制器(存储器直接存取控制器)。当$\overline{\text{BUSRQ}}$有效时,Z-80 结束现行机器周期后立即响应,使数据总线、地址总线和部分控制总线($\overline{\text{MREQ}}$、$\overline{\text{IORQ}}$、$\overline{\text{WR}}$、$\overline{\text{RD}}$)引脚处于高阻状态。

$\overline{\text{BUSAK}}$(总线响应),输出,低电平有效。$\overline{\text{BUSAK}}$信号告诉总线请求装置,CPU 的地址总线、数据总线和部分控制总线引脚已处于高阻状态,请求装置(例如 DMA 控制器)可以利用系统总线了。

φ(时钟),Z-80 典型时钟频率为 2.5MHz,Z-80A 的典型时钟频率为 4.0MHz。φ 是由外部供给 Z-80 的时钟,系统中的其他部件,例如接口电路,也常利用它作为一种节拍脉冲或时间基准,所以,系统控制总线应该包括它。

1.2 总线的概念

什么是总线?总线是一组公用导线,一些数据源中的任何一个都可以利用它传送数据到另一个或多个目的。例如,在图 1.3 中,假定数据从器件 A 传送到器件 C ,此时器件 A 是数据源,器件 C 是目的;在另一时刻,数据从器件 D 传送给器件 A,此时器件 D 是源,器件 A 变成目的。要使数据传送无错误,要有一定的时序,第一个事件结束后才能开始第二个事件。此外,在给定时间周期内,源只能有一个,目的则可以有多个。换言之,总线是分时复用,在特定时间周期内,总线只能为一个源专用,从属于一个源。为了简单,这里只分析了一条导线,Z-80 微处理器有 8 条数据总线,还有 16 条和 32 条的 CPU,显然,其分析是一样的。

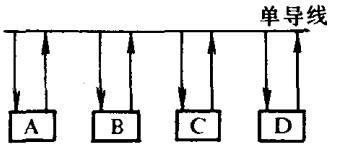

图 1.3 单导线总线

1.2.1 集电极开路电路

图 1.4 示出几个源的输出通过总线连在一起,标准 TTL 门电路不允许进行这种连接。假定作了这种连接,当一个门电路输出高电平而另一个输出低电平时,它们的输出将产生很大的电流,结果导致输出电平不确定。为了说明这个问题,图 1.4 (a)示出标准 TTL 门电路的原理图,图 1.4 (b) 示出这种电路连在一起的两个输出。假定 A 电路输出低电平,其 $Q3_A$ 导通,$Q4_A$ 截止;B 电路输出高电平,其 $Q3_B$ 截止 $Q4_B$ 导通。结果是 5V 电源将通过 B 电路导通的 $Q4_B$ 和 A 电路导通的 $Q3_A$ 产生一个大电流,近似短路。此电流值超过器件的额定值,

因此输出将不是标准的 TTL 高电平，也不是 TTL 低电平；此外，器件还可能烧坏。

上述问题的解决方法，是用集电极开路电路。集电极开路需外接一个上拉电阻，使输出达到高态。图 1.5 示出两个集电极开路输出连在一起，共用一个上拉电阻，因此最大电流将会被限制。假定两个晶体管或一个晶体管导通，输出将是低电平，相反，若两个晶体管均截止，则为高电平。

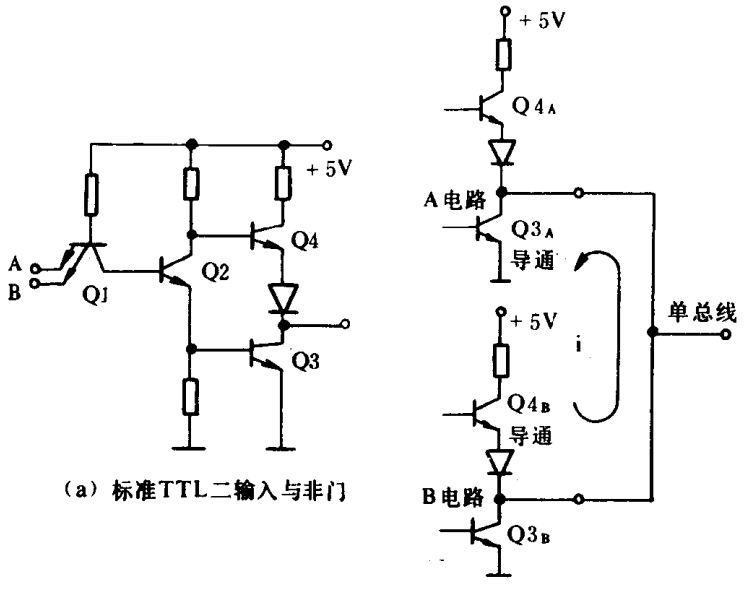

图 1.4 TTL 输出不能互连的说明

图 1.6 示出如何用集电极开路电路实现总线的概念。例如，要求信号从器件 A 传送到器件 C，$\overline{\text{TRAN A}}$必须有效而 $\overline{\text{TRAN B}}$、$\overline{\text{TRAN C}}$和 $\overline{\text{TRAN D}}$必须无效，DATA A 在门 1 输出，它的“非”出现在总线上。因为$\overline{\text{TRAN B}}$、$\overline{\text{TRAN C}}$和$\overline{\text{TRAN D}}$无效，所以 Q2、Q3 和 Q4 截止（与总线分离）。为了实现传输，$\overline{\text{RCV C}}$也必须有效，使门 6 的输入开，门 6 输出总线数据的“非”。门 6 的输出等于 DATA A。任何其他$\overline{\text{RCV}}$线也可以有效，数据传给多个目的。因为，在给定的时间周期内目的可以有多个，源仅能有一个。

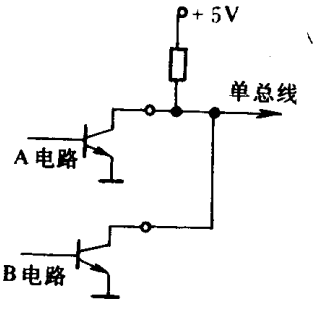

图 1.5 两集电极开路电路的输出电路

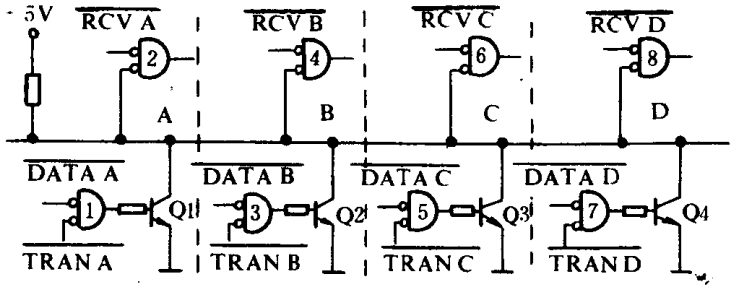

图 1.6 用集电极开路电路实现总线传输

1.2.2 三态电路

三态电路对利用总线传送数据提供了另一种手段。三态电路和集电极开路电路的主要区别是保留有源上拉电阻，使输出阻抗可控制。

三态的定义：(1)低阻抗高电平输出，(2) 低阻抗低电平输出，(3)高阻抗输出。图 1.7 示出一种三态非门，当禁止端$\overline{\text{DISABLE}}$高电平时，$D_1$ 截止，它和图 1.4(a)的电路相同。所以当输入端为高电平时，因 Q3 导通和 Q4 截止而输出低电平，并且是低阻抗的。当输入低电

平时，因 Q3 截止和 Q4 导通而输出高电平，输出阻抗因 Q4 导通，仍是低阻抗的。但当 $\overline{\text{DISABLE}}$端为低电平时，$D_1$ 导通将 Q4 的基极箝在低电平，使其截止。另一方面，Q3 也因输入端为低电平而截止，所以输出呈高阻态，这就是第三态。正因为有这第三态，才使它适合挂接在总线上使用。

三态反相器的符号示于图 1.8。它和一般反相器比，多一个三态控制端。此控制端有时用禁止端表示，如图 1.8(a)所示。也有时用使能端$\overline{\text{ENABLE}}$表示，如图 1.8(b)。对图 1.8(a)，当$\overline{\text{DISABLE}}=0$时，输出为高阻态，对图 1.8(b) 来说，当 $\overline{\text{ENABLE}}=0$ 时输出为低阻抗状态。

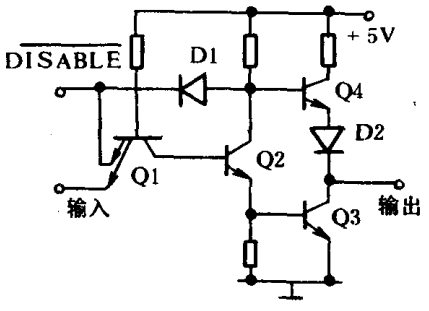

图 1.7　三态反相器

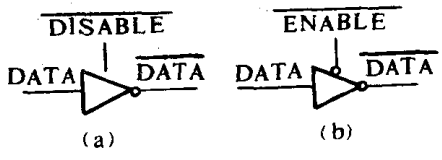

图 1.8　三态反相器的符号

用三态电路实现总线数据传输的说明示于图 1.9。假设要求把数据 A(DATA A)传送给器件 D，$\overline{\text{TRAN A}}$应当有效，使 DATA A 送上总线，$\overline{\text{RCV D}}$也应当有效，将总线上的数据选入 D 器件，使它在 D 输出出现，因为只允许有一个源利用总线，所以，在作上述传输时$\overline{\text{TRAN B}}$、$\overline{\text{TRAM C}}$和$\overline{\text{TRAN D}}$都必须无效，使不用电路呈高阻，与总线分离。

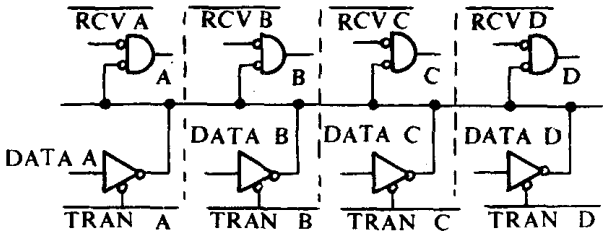

图 1.9　用三态电路实现总线数据传输

1.2.3　总线冲突

在分析总线传输的基础上，我们再引入“总线冲突”的概念。当有两个或两个以上的源同时要传送信息，若源 A 送出一个逻辑“0”，而源 B 在同一时刻打算传送一个逻辑“1”，就发生了总线冲突，实际上给出的信息取决于驱动总线的逻辑器件。若由集电开路器件驱动总线，在冲突期间传送一个“0”状态，则源 B 输出的逻辑“1”丢了。这样，在冲突的时候，一个或两个源会丢失数据，丢失的数据又是不能预料的。因此，冲突几乎一定会导致总线传输的故障。要保证可靠的传输，一般规定，每次在总线上只能有一个器件发送信息，可能接收数据的器件有多个。

请注意,三态电路使用不当,同样会发生总线冲突,因为三态电路被使能后就和一般TTL电路一样。

总线冲突可能造成比丢失数据更大的损害。在总线传输中,在总线上的两个器件一旦发生总线冲突,就建立了一条从 V_{CC} 通过两个冲突门的输出到地的一个低阻抗通路(见图1.4b),通过这个通路将产生一个大电流,此电流不仅造成输出逻辑电平不高不低,还可能烧坏这两个门电路。如果两个门之一损坏,那么这个故障可能和同一条信号线上的其他门再相冲突,又可能使它烧坏。如果总线冲突发生在取指周期,那么 CPU 的指令传送将被破坏,从而程序的正常运行被破坏。总之,无论总线冲突造成什么样的后果,都是不允许的,在接口设计中应当谨慎,避免产生。

1.3 输入口和输出口

输入口是外部设备通过总线向 CPU 输入数据的端口,它是 CPU 从外部设备输入数据的接口的电路。图 1.10(a) 示出简化的 8 位数据总线和 3 个输入口。

我们从 0# 号端口开始分析外设数据如何输入 CPU。端口由三态门组成,0# 号端口的三态门使能端连接门 A 的输出,由门 A 控制。当不进行数据交换时,门 A 输出低电平,0# 号端口高阻态与总线分离。当输入数据时,三态门被使能,外设的数据通过 0# 号端口送上总线。A 门输入受端口选择信号$\overline{PS_0}$和 I/O 读控制信号$\overline{IOR}$的控制。微处理器执行输入指令,$\overline{PS_0}$有效,接着$\overline{IOR}$有效,数据从 0# 号端口送上总线,CPU 从其总线引脚取入内部。在图 1.10 中,$\overline{IOR}$对各端口是公共的,但微处理器寻址时只有一个端口被选中。在上述情况下,应当是 0# 号端口选中,即$\overline{PS_0}=0$,$\overline{PS_1}$和$\overline{PS_2}$都应当为"1"。

以上分析中实际上已假定外设在 0# 号口提供的数据是长时间存在的。实际微处理器的读操作是在一瞬间完成的。有些外部设备提供数据的存在时间也是短暂的,不可能和微处理器的读瞬时同步,这就要求在输入端口中有锁存器,而且外设还必须提供一个时钟脉冲,将输入数据锁存住 ,以备微处理器按它自己的定时读入。图 1.10(b)示出带锁存器的输入口(图中 RESET 信号的作用见 1.5 节)。

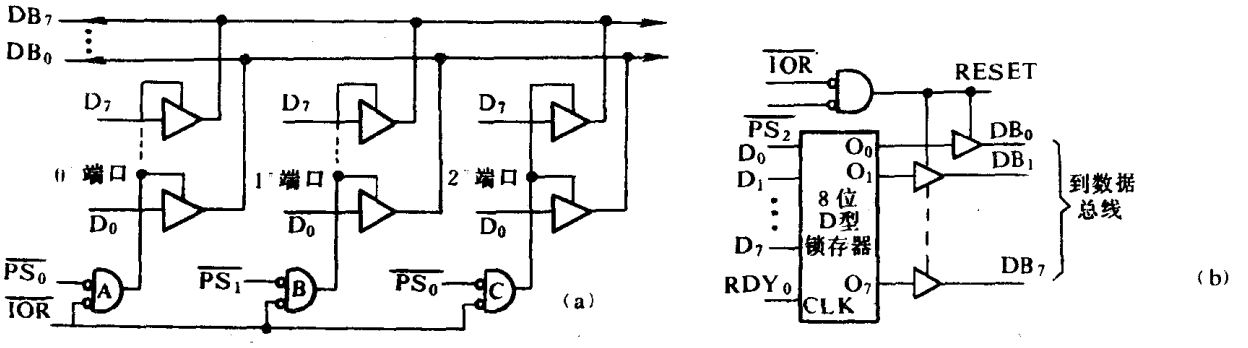

图 1.10 输入口

(a)三态门输入口 (b)锁存型输入口

输出口是微处理器通过总线向外部设备输出数据的端口，也是微处理器向外部设备输出数据的接口。

微处理器向外输出数据的时间间隔是很短的(约 0.5～1.0μs)。外部设备所需数据的持续期总是大于此间隔，所以输出口必须用锁存器。图 1.11 示出用 8 位 D锁存器组成的输出口。和输入口相反，这里输入是数据总线，输出引线和外设相连。微处理器执行输出指令时，先使端口选择信号$\overline{PS_0}$有效，接着写控制信号$\overline{IOW}$有效，此两信号"非与"将微处理器放到总线上的数据进行锁存。锁存的数据提供外设使用。输出口中不需要三态电路，因为总线允许同时有多个目的。图 1.11 和图 1.10(a)使用相同的端口选择信号$\overline{PS_0}$，虽然端口地址相同，是输出操作还是输入操作，由不同的控制信号(IOW 和 IOR)进行区别，不会造成传输上的混乱。

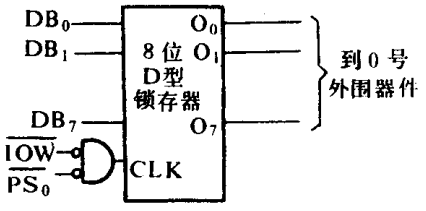

图 1.11　8位输出口

1.4　输入和输出时序

图 1.12 给出 Z-80 CPU 的输入和输出的定时关系，也称作输入和输出时序。(实际的时序曲线不是垂直变化或圆弧形变化，这里为了突出定时关系，进行了简化。以下相同，不再说明。)Z-80 有一条输入指令 IN A,(n)，它包括 3 个不同的机器周期：

(1) 取指周期，从存储器读操作码；

(2) 取操作数周期，从存储器读操作数；

(3) 输入周期，从端口 n 读数据到累加器 A。

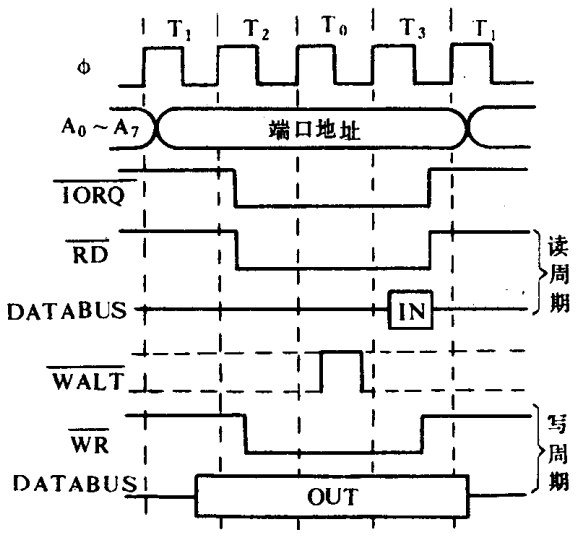

图 1.12　输入输出周期的定时关系

图 1.12 上部分是输入周期的定时关系，说明从端口 n 读数入累加器 A 的输入周期时序。从图中看出，在 T_1 上升沿后 CPU 将 I/O 端口的地址码 n(即上节的$\overline{PS_0}$)放到地址总线上。I/O 请求信号$\overline{IORQ}$有效时，地址码已经稳定(端口地址线已趋水平)。$\overline{RD}$几乎和$\overline{IORQ}$同时有效。从图中还可看出，输入时序要求在 T_3 前沿以后应将数据送上数据总线，以便 CPU 在 T_3 下降沿时采样数据总线，将数据取入 CPU 的累加器。所以$\overline{IORQ}$和$\overline{RD}$将作为控制三态门使能的控制信号，因此$\overline{IOR} = \overline{RD} + \overline{IORQ}$。若输入信号是短暂的，它不能维持到 T_3 时刻，CPU 将无法读入，因此，需要采用锁存型输入口。相反，若$\overline{IOR}$

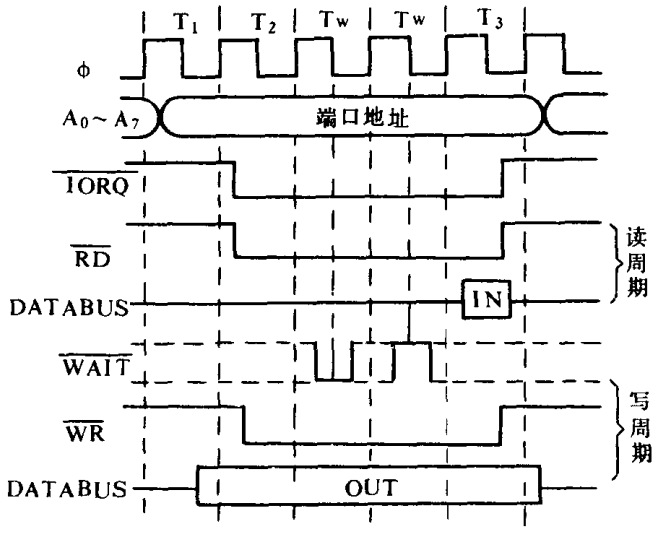

图 1.13　有插入状态的输入和输出周期定时关系

有效后，输入信号还不能出现在数据线上，这是慢速输入信号的情况，就需要使用 Z-80 CPU 的$\overline{\text{WAIT}}$控制信号，如图 1.13 所示。$\overline{\text{WAIT}}$改为低电平，可以插入等待周期 T_W，推迟 CPU 采样的时刻，保证输入数据可靠的输入。上节输入端口只提供最基本的原理，实际应用时，还要考虑输入信号的性质，保证满足 CPU 输入时序的要求，必要时加锁存器或设法插入等待状态 T_W。

为了节省篇幅，输出时序就绘在图 1.12 和图 1.13 的下半部分。从图中看出，输出数据存在的时间比要求输入数据存在的时间长得多，但是它与一般外设对输出数据的要求相比还不够长。就 Z-80 CPU 来说，图 1.12 示出输出数据存在的时间约 3 个时钟周期，每个时钟周期为 250ns，输出数据存在的时间还不到 1μs。外设要求输出数据存在的时间经常是 ms 级，所以输出口一般都是一个锁存器。

1.5　状态口和控制口

外设数据准备好，可以输入给微处理器，微处理器如何能知道呢？外设准备好接收微处理器的输出数据，微处理器又如何知道呢？要解决上述问题，在数据传送中还需进行状态信息的交换。图 1.14 示出状态输入端口，从外设向微处理器输入状态信息，状态口也是输入口，不过输入的是状态信息，而不是外设的数据。状态口（图 1.14）和输入口（图 1.10(b)）结合使用，可以解决较复杂的输入问题。现以 0# 端口为例说明全过程如下：

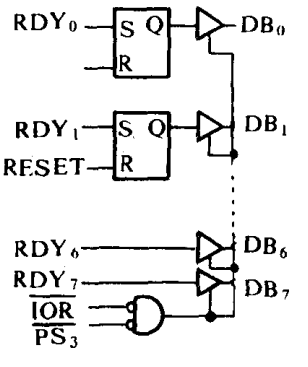

图 1.14　状态口

(1)外设提供的数据送到输入口输入（图 1.10(b)中的 D_0 ~ D_7）。

(2)外设提供钟控 RDY_0 脉冲，使数据锁存入 D 型锁存器（图 1.10(b)）。

(3)同一钟控脉冲 RDY_0 使状态口的 RS 触发器置位，记存外设的状态，在图 1.14 中用置“1”表示“准备好”。

(4)微处理器使$\overline{\text{IOR}}$和$\overline{PS_3}$有效，输入状态信息。

(5)微处理器识别输入的状态信息。

(6)微处理器使$\overline{\text{IOR}}$和$\overline{PS_2}$有效，从端口读入已锁存的外设数据（图 1.10(b)）。

(7)数据读入微处理器后，RESET（图 1.10 (b)）信号有效，使 RS 触发器复位（图 1.14），以准备下次记存状态信息。

假若外设提供的状态信号的持续期足够长，由微处理器直接检测不成问题，RS 触发器可以省去，如图 1.14 中的$RDY_6 \sim RDY_7$。

有些外围设备需要控制信号，微处理器可以输出它所要求的控制信号，输出控制信号给外设的接口电路称控制口。图 1.15(a)示出多数情况适合的控制口。从图看出，它和图 1.11

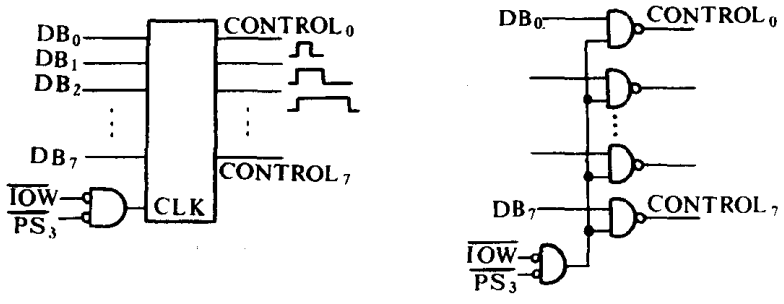

图 1.15　控制口

(a)锁存型　　(b)非锁存型

的输出口电路相同，只是输出连接 8 条控制线。控制和传输的着眼点不同，前者要求对每条控制线提供相应的时间函数，而后者要求 8 条线的编码(8 位码)。对一条控制线来说(例如 $CONTROL_0$)，微处理器先输出 $DB_0 = 1$，再输出 $DB_0 = 0$，则在 $CONTROL_0$ 上产生一个脉冲。若输出“1”后等待一段时间再输出“0”，或连续输出“1”再输出“0”，即能产生具有一定宽度的脉冲。对 8 条控制线来说，微处理器连续输出与 8 个时间函数对应的 8 位字，就能在控制口输出产生所要求的 8 个控制信号。若外设要求微秒级的控制脉冲，图 1.15(a)中的 D 型锁存器可用与门电路代替，如图 1.15(b)。现举一例，说明复杂的外设如何通过控制口，状态和数据口在微处理器控制下进行工作。图 1.16 示出微处理器控制的模 - 数转换器(A/D)电路图，它的工作顺序如下：

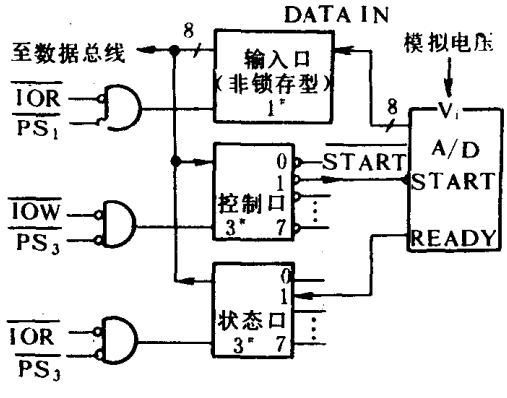

图 1.16　从 A/D 输入数据

(1)微处理器通过控制口输出，控制第一位线(图中 3# 端口的 1)。这使控制线$\overline{START}$成为有效，启动 A/D 开始转换。转换有一过程，长短取决于 A/D 的型号，总之，微处理器要等待。

(2)处理器读入状态位 READY，检查是否完成，若未完成，继续检测。

(3)当换转过程结束，A/D 发出 READY 有效信号。

(4)微处理器检到 READY 信号有效后，通过输入口 1 读入数据(对应输入模拟电压值的数字量)。

1.6 中断向量输入口和中断响应时序

还有一种和微处理器数据总线挂接的输入口如图 1.17 所示，它是按照 CPU 的中断操作时序，在中断响应信号$\overline{INTA}$有效时将一组数据送到总线上，由微处理器读入，作为中断向量或中断向量的地址或中断类型号。

中断向量输入口的控制信号不是 IOR 和地址信号而是$\overline{INTA}$。现用 Z-80 的中断响应时序进行说明。虽然各种 CPU 的中断响应时序会有差异，但都会产生一个中断响应信号 INTA。图 1.18 示出 Z-80 的中断请求和响应时序。从图中可看出，在每条指令的最后一个时

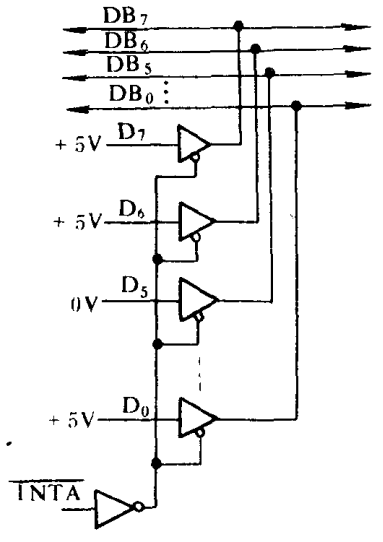

图 1.17 中断向量输入口

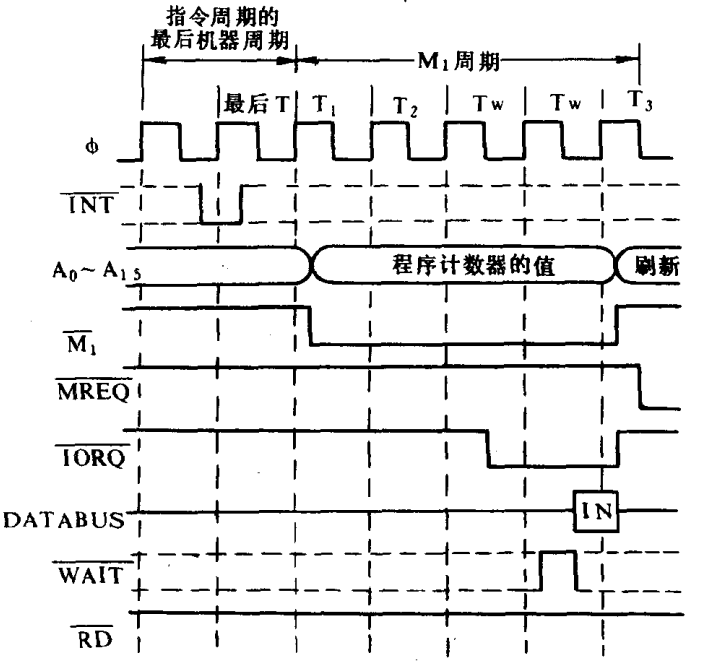

图 1.18 中断请求和响应周期定时关系

钟脉冲上沿，CPU 将采样中断请求信号$\overline{INT}$，若有中断请求（CPU 内部中断允许触发器假定在开中断状态）CPU 即响应中断请求，CPU 的工作从此进入响应中断周期，对 Z-80 来说，就是一个特殊的 M_1 周期。在此周期内用$\overline{IORQ}$有效代替取指周期的$\overline{MREQ}$（存储器请求）有效；此外从图中还可看出，CPU 自动插入两个 T_W 状态后，在 T_3 前沿采样数据总线，把数据读入 CPU，本章后几节将会讲到，读入的数据将用以决定中断服务子程序的入口地址。若数据不能准时送上数据线也可用 WAIT 信号来增加 T_W 状态，推迟 CPU 的采样，保证读入时序的配合。但是，这种情况是少有的。从响应中断时序分析看出，在 Z-80 情况，$\overline{INTA}$由 $\overline{M}$ 有效和$\overline{IORQ}$有效的“或”产生，即$\overline{INTA} = \overline{M_1} + \overline{IORQ}$。从中断响应时序还看出，读控制信号$\overline{RD}$始终无效，因此，在此周期内不会开启其他任何输入口，不会有总线冲突发生。

1.7 存储器读、写时序和存储器接口

存储器接口和输入/输出接口一样，首先应该满足 CPU 总线的时序要求。我们将从存储器的读、写时序着手研究。仍以 Z-80 为例，由此得出的结论对其他 CPU 也适合。图 1.19 示出 Z-80 的存储器读、写时序。图 1.19 和图 1.12 相比较，主要区别就是用存储器请求信号$\overline{MREQ}$代替了 I/O 请求信号$\overline{IORQ}$。此外在周期状态方面没有预先自动插入一个 T_W，因为现代计算机的存储器主要都是集成电路，工作速度快，时间要求容易满足。从图中还看出，CPU 的设计者也为慢速存储器接口安排了$\overline{WAIT}$信号，一般很少遇到这种情况。存储器读和 I/O 输入相似，CPU 先把存储器地址放到地址线上，然后$\overline{MREQ}$和$\overline{RD}$相继有效，在 T_3 脉冲下降沿采样数据线，把数据读入 CPU。因此，存储器读接口也像输入接口一样，只需用地址信号、$\overline{MREQ}$和$\overline{RD}$去控制存储器的三态门，把数据送上数据线即可。为了规范化，以后将用一个存储器读信号$\overline{MEMR}$作控制，对 Z-80 来说，$\overline{MEMR} = \overline{MREQ} + \overline{RD}$。

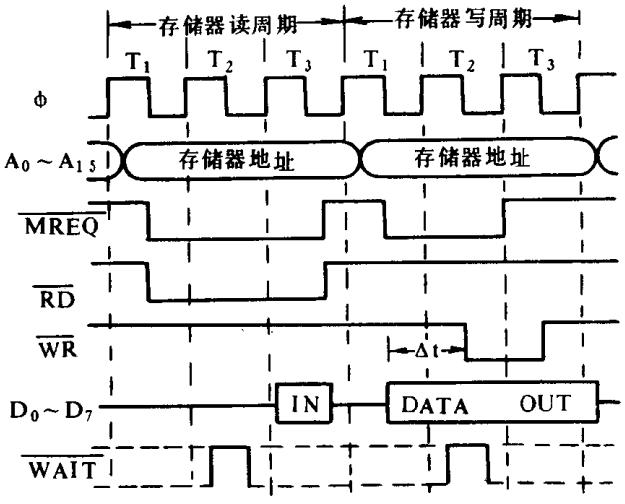

图 1.19 存储器读、写周期的定时关系

存储器写和 I/O 输出相似，CPU 先把存储器地址放到地址线上，然后又把要写的数据放到数据总线上，相继$\overline{MREQ}$和$\overline{WR}$有效。因此存储器写接口设计，只要用地址信号、$\overline{MREQ}$和$\overline{WR}$去控制把数据锁存入指定单元即可。为了规范化，今后用存储器写控制信号$\overline{MEMW}$，对 Z-80 来说，$\overline{MEMW} = \overline{MREQ} + \overline{WR}$。以下研究各种存储器的具体接口。存储器分读写存储器(RAM)和只读存储器(ROM)两大类。读写存储器又分静态读写存储器(SRAM)和动态读写存储器(DRAM)。只读存储器有多种，但常见到的也分两种：掩膜 ROM 和可擦可编程 ROM (EPROM)。

1.7.1 ROM 接口

掩膜 ROM 在工厂制造过程中已将信息存入，封装后不能改变，用户只能读出，不能修改。ROM 芯片对外的引脚有数据线、地址线和片选控制$\overline{CS}$。为了传送数据，数据线引脚和 CPU 数据总线连接。为了选取存储单元，地址线引脚和 CPU 地址总线连接，ROM 在数据传输中是源，不能总占据数据线，只有当 CPU 需要它将数据送上数据总线时它才占据数据总线，所以有片选控制端$\overline{CS}$。它控制 ROM 内部的输出三态门，当$\overline{CS}$有效时，三态门被使能；当$\overline{CS}$无效时它的数据总线引脚呈高阻态。在接口时，此$\overline{CS}$应该等于存储器选择信号$\overline{MS}$和存储器读控制信号$\overline{MEMR}$的“或”，如图 1.20(图中地址线省略)。$\overline{MS}$由地址线译码决定(参见下节)。图 1.20 的连接可以实现：当微处理器将存储器芯片选中并进行读操作时，存储器芯

片才将数据送上总线，其余时间它与数据总线分离（高阻态）。

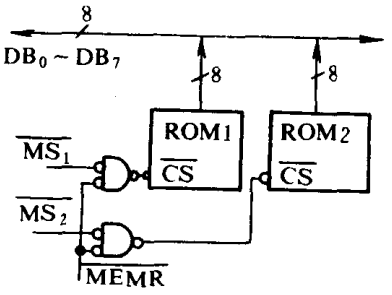

图 1.20　ROM 接口

1.7.2　EPROM 接口

EPROM 是这样一种存储器芯片，它在微处理器系统中完全起掩膜 ROM 的作用，但是它存储的信息可由用户编程器（或称 EPROM 写入器）写入。用户也可用一种紫外线擦抹器将已存的信息抹去，进行改写，使用比较灵活方便。表 1.1 给出 EPROM2716 的 4 种工作模式和引脚的关系，2716 有 3 个控制引脚$\overline{CE}$/PGM（$\overline{CS}$/PGM）、$\overline{OE}$和 V_{PP}。

表 1.1　EPROM 2716 模式选择

脚 / 模式	$\overline{CE}$/PGM（18 脚）	$\overline{OE}$（20 脚）	V_{PP}	输出
读	低	低	+5	输出
未选中	无关	高	+5	高阻
功率下降	高	无关	+5	高阻
编程	TTL 电平正脉冲 50ms	高	+25	输入

在微处理器系统中应用时 V_{PP}固定为 5V。当微处理器进行读操作时，$\overline{CE}=0$ 和 $\overline{OE}=0$，EPROM 输出数据；在其余时间它应当处于高阻状态，与数据总线分离。后一种状态有两种模式可选。一般选功率下降模式，功率下降 75%（从 525mW 降至 132mW），$\overline{CE}$ = TTL 高电平。按这种考虑的接口方法如图 1.21 所示。$\overline{CE}$引线接存储器选择信号$\overline{MS}$，$\overline{OE}$接存储器读控制信号$\overline{MEMR}$。

以上 EPROM 接口方法，仅提供考虑的原则，不同型号的 EPROM 的控制信号引脚可能不同，接口时还需要具体分析。

1.7.3　SRAM 接口

RAM 是读写存储器，当被微处理器读时，它是数据源，当被写时，它变成数据的目的。

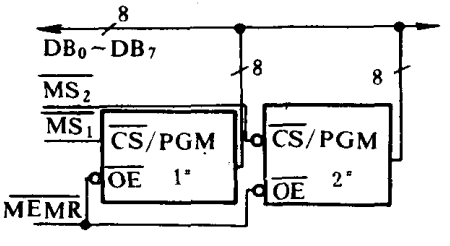

图 1.21　EPROM2716 接口图

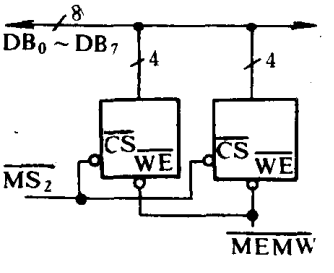

图 1.22　RAM2114 接口