第3章 微型计算机输入输出接口

CHAPTER

外部设备是构成微型计算机系统的重要组成部分。程序、数据和各种外部信息要通过外部设备输入到计算机内，计算机内的各种信息和处理的结果要通过外部设备进行输出。微型计算机和外部设备的数据传输，在硬件线路与软件实现上都有其特定的要求和方法。本章重点讨论连接微型计算机系统总线和外部设备的硬件电路——输入输出接口(Input/Output Interface，I/O接口)的结构和组成方法，微型计算机和外部设备数据传输的方法，以及I/O接口的程序设计。

3.1 输入输出接口

3.1.1 外部设备及其信号

1. 外部设备(Peripheral Device)

微型计算机使用的外部设备种类很多，它们的内部结构，工作原理，使用方法各异，按照它们与CPU数据传输的方向，可以划分为以下三类。

(1) 输入设备。例如常见的键盘、鼠标、光笔，输入图形信息的扫描仪、数码相机，检测现场信息的数字化测试仪表，模拟量采集和模数转换装置等。

(2) 输出设备。显示器、打印机是最常见的输出设备，绘图仪，用于现场控制的数模转换装置和执行元件也属于这一范畴。

(3) 复合输入输出设备。外存储设备是典型的复合输入输出设备。它可以接收来自CPU和内存储器的程序、数据并储存在磁带、磁盘上，也可以根据CPU的指令把所储存的程序、数据送往系统总线。

常用的外存储设备有磁带机(Tape Driver)，软磁盘驱动器(Floppy Driver)，硬磁盘驱动器(Hard Disk Driver)和光盘驱动器(Compact Disk Driver)。磁带、软磁盘、硬磁盘、光盘分别是它们的存储介质(Medium)，许多光盘只能读出信息，称为CD-ROM(Compact Disk—Read Only Memory)，只读光盘驱动器只能用作输入设备。

2. 外部设备的信号

外部设备传输的信号有以下三种类型。

(1) 数据信号。数据信号是外部设备信号的主要部分。按照信号的物理形态，可分为以下几种。

① 数字量。以二进制形式表述的数据、图形或文字信息。例如，由磁盘驱动器读出的程序代码，送往打印机的字符 ASCII 代码等。

② 模拟量。现场的物理量(温度、压力、流量、位移等)通过传感器件，转换为大小与之对应的电压或电流信号。这些量呈连续变化的形态，称为模拟量(Analog)。模拟量必须经过模数转换(Analog/Digital Convert 简称 ADC)才能送往 CPU。反之，由 CPU 送出的数字量，可以经过数模转换(Digital/Analog Convert，简称 DAC)变换为大小与数字量对应的电压或电流，由放大器放大后通过执行元件(例如电磁阀)来控制现场的设备。

③ 开关量。开关量是只有两种状态(0，1)的量，如开关的接通(ON)与断开(OFF)，电动机的启动与停止等。

④ 脉冲量。计数脉冲、定时脉冲和控制脉冲在计算机控制系统中也很常见，它们统称为脉冲量。

对于输入设备，数据信号从外部设备送往 CPU，对于输出设备，数据信号从 CPU 发往外部设备。

(2) 状态信号。状态信号表明外部设备当前的工作状态，用来协调 CPU 与外部设备之间的操作。输入设备在完成一次输入操作之后，发出就绪信号(READY)，等待 CPU 给予处理。输出设备在接收一批数据信息进行输出的过程中，发出忙信号(BUSY)，表明目前不能接收新的数据信息。有的设备有指示出错状态的信号，如打印机的纸尽(Paper Out)，故障(Fault)。不同的外部设备可以有不同的状态信号。

状态信号总是从外部设备发往 CPU。

(3) 控制信号。控制信号是 CPU 向外部设备发出的命令，它指定设备的工作方式，启动或停止设备。控制信号的格式因设备而异。控制信号从 CPU 发往外部设备。

数据信号、状态信号、控制信号都是以“数据”的形式，通过数据总线与 CPU 进行传输的。

3.1.2 I/O 接口的功能

接口是计算机一个部件与另一个部件之间的连接界面。作为连接计算机系统总线与外部设备的 I/O 接口，可以有如下的功能。

1. 设备选择功能

CPU 通过地址代码来标识和选择不同的外部设备。接口应能对系统总线上传输的外部设备地址进行译码，在检测到本设备地址代码时，产生相应的“选中”信号. 并按 CPU 的要求进行信号传输。

2. 信息传输功能

在设备被选中时，接口应能从 CPU 接收数据或控制信息，或者将数据或状态信息发往数据总线。

3. 数据格式转换功能

外部设备使用的数据格式与 CPU 数据格式不同时，接口要进行两种数据格式之间的相互转换。

4. 联络功能

接口从系统总线或外部设备接收一个数据，发出“数据到”联络信号，通知外部设备或 CPU 取走数据，数据传输完成，向对方发出信号，准备进行下一次传输。

5. 中断管理功能

中断管理功能主要如下：向 CPU 申请中断；向 CPU 发中断类型号；中断优先权的管理等。在以 8086 为 CPU 的系统中，这些功能大部分可以由专门的中断控制器实现。

6. 复位功能

接口在接收系统的复位信号后，将接口电路及其所连接的外部设备置成初始状态。

7. 可编程功能

有些接口具有可编程特性，可以用指令来设定接口的工作方式、工作参数和信号的极性。可编程功能扩大了接口的适用范围。

8. 错误检测功能

许多数据传输量大，传输速率高的接口具有信号传输错误的检测功能。常见的信号传输错误有以下两种。

(1) 物理信道上的传输错误。信号在线路上传输时，如果遇到干扰信号，可能发生传输错误。检测传输错误的常见方法是奇偶检验。以偶校验为例. 发送方在发送正常位数据信息的同时，增加一位“校验位”，通过对校验位设置为“0”或“1”，使信息位连同校验位中“1”的个数为偶数。接收方核对接收到的信息位、校验位中“1”的个数。若“1”的个数是奇数，则可以断定产生了传输错误。需要说明的是，奇偶校验是一种比较简单的检验方法，它能够确定某次数据传输是错误的，但却不能确定某次数据传输一定是正确的。

(2) 数据传输中的覆盖错误。输入设备完成一次输入操作后，把所获得的数据暂存在接口内。如果在该设备完成下一次输入操作之前，CPU 没有从接口取走数据，那么，在新的数据送入接口后，上一次的数据被覆盖，从而导致数据的丢失。输出操作中也可能产生类似的错误。

覆盖错误导致数据的丢失，常发生在高速数据传输的场合，例如硬磁盘驱动器的数据输入输出。

3.1.3　I/O 端口的编址方法

I/O 端口的编址有两种不同的方式。

1. I/O 端口与内存统一编址

这种编址方式也称为存储器映射编址方式，它把内存的一部分地址分配给 I/O 端口，一个 8 位端口占用一个内存字节单元地址。已经用于 I/O 端口的地址，存储器不能再使用。

I/O 端口与内存统一编址后，访问内存储器单元和 I/O 端口使用相同的指令，这有助于降低 CPU 电路的复杂性，并给使用者提供方便。但是，I/O 端口占用内存地址，相对减少了内存可用范围。而且，由于难以区分访问内存和 I/O 的指令，降低了程序的可读性

和可维护性。

2. I/O端口与内存独立编址

这种编址方法中，内存储器和I/O端口各自有自己独立的地址空间。访问I/O端口需要专门的I/O指令。

8086/8088 CPU采用这种方式：

(1) 访问内存储器使用20根地址线 A_0～A_{19}，同时使 $M/\overline{IO}=1$，内存地址范围为00000～0FFFFFH共1MB；

(2) 访问I/O端口使用低16根地址线 A_0～A_{15}，同时使 $M/\overline{IO}=0$，I/O端口地址范围为0000～0FFFFH。

因此，两个地址空间相互独立，互不影响。

8086/8088 CPU中，I/O指令的端口寻址方式有直接寻址和间接寻址两种方式，直接寻址方式时地址范围为0～255，间接寻址方式时地址范围为0～0FFFFH。

IBM PC微型计算机I/O端口地址分配如下：在IBM PC系列微型计算机中，仅使用 A_0～A_9共10条地址线定义I/O端口(设 A_{11}～$A_{15}=0$)，寻址范围为0～3FFH。其中前256个端口地址供主板上寻址I/O接口芯片使用，后768个供扩展槽接口卡使用，部分I/O端口分配情况列于表3-1中。

表3-1 系统板(主板)上I/O部分接口器件的端口地址

I/O接口器件名称	PC/XT	PC/AT
DMA控制器1	000～01FH	000～01FH
中断控制器1	020～021H	020～021H
计时器	040～043H	040～05FH
并行接口芯片	060～063H	—
键盘控制器	—	060～06FH
RT/CMOS RAM	—	070～07FH
DMA页面寄存器	080～083H	080～09FH
中断控制器2	—	0A0～0BFH
NMI屏蔽寄存器	0A0～0BFH	—
DMA控制器2	—	0C0～0DFH
协处理器	—	0F0～0FFH

用户设计I/O接口电路的时候，应使用系统尚未占用的端口地址。

3.1.4 简单I/O接口的组成

1. 端口

接口内通常设置有若干个寄存器，用来暂存CPU和外部设备之间传输的数据、状态和命令。这些寄存器被称为端口(Port)。根据寄存器内暂存信息的种类，可以有数据端

口、命令端口(也称控制端口)和状态端口。每一个端口有一个独立的地址,CPU 可以用地址代码来区别各个不同的端口,对它们进行读、写操作。CPU 对状态端口进行一次读操作,可以得到该端口暂存的状态代码,从而获得与这个接口相连接的外部设备的状态信息。CPU 对数据端口进行一次读或写操作,也就是与该外部设备进行一次数据传输。而 CPU 把若干位代码写入命令端口,则意味着对该外部设备发出一个控制命令,要求该设备按代码的要求进行工作。由此可见,CPU 与外部设备的输入输出操作,都是通过对相应端口的读写操作来完成的。所谓外部设备的地址,实际上是该设备接口内各端口的地址,一台外部设备可以拥有几个通常是相邻的端口地址。

2. 地址译码电路

地址译码是接口的基本功能之一。CPU 在执行输入输出指令时,向地址总线发送 16 位外部设备的端口地址。在接收到与本接口相关的地址后,译码电路应能产生相应的选通信号,使相关端口寄存器进行数据、命令或状态的传输,完成一次 I/O 操作。由于一个接口上的几个端口地址通常是连续排列的,可以把 16 位地址码分解为两个部分:高位地址码用作对接口的选择,低位地址码用来选择接口内不同的端口。为了简化电路,IBM PC 微型计算机使用 $A_0 \sim A_9$ 作为外部设备端口的地址。

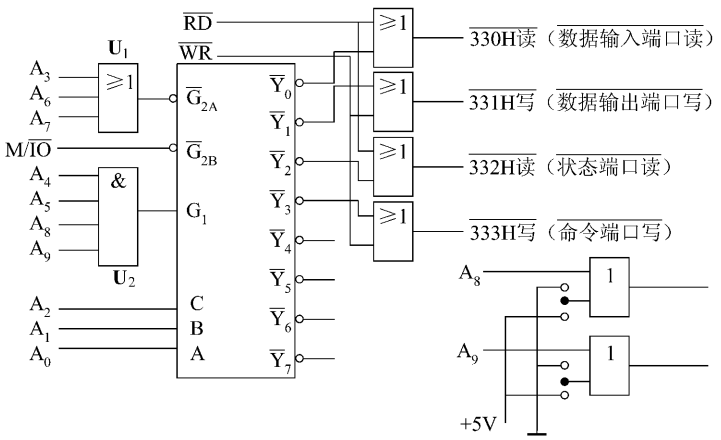
例如,某接口数据输入端口、数据输出端口、状态端口、命令端口的地址分别为 330H、331H、332H 和 333H。假设该系统也使用 10 位外部设备端口地址。那么,当 8 位高位地址码为 11001100 时,表明本接口被选中,2 位低位地址码的 4 种组合 00、01、10 和 11 分别表示选中了本接口的数据输入端口、数据输出端口、状态端口、命令端口。由此,在最小模式的系统中,可以设计如图 3-1 所示的译码电路。图中的 U_1 为或门,选中本接口时,地址码 A_7、A_6 和 A_3,均为“0”,于是 U_1 输出端为“0”。同时。选中本接口时,地址码 A_9、A_8、A_5 和 A_4 全为“1”,于是与门 U_2 输出“1”,它们连同 M/$\overline{IO}$的“0”使 3-8 译码器工作。根据 A_2 A_1 A_0 的 8 种组合,分别得到 8 个地址选择信号(本接口只使用其中的 4 个),与 $\overline{RD}$、$\overline{WR}$的进一步组合,就得到了本例中 4 个端口的读写选通信号。

设定端口地址时,注意不能和已有设备的端口地址重复。为了避免重复的发生,许多接口电路允许用“跳线器(JUMPER)”改变端口地址。图 3-1(b)给出了一个“跳线器”的例子。异或门(半加器)的一个输入引脚来自跳线器的中间引脚,它可以由使用者选择连接“1”(+5V)或“0”(接地),另一个引脚分别连接 A_8 和 A_9。将异或门的输出代替图 3-1(a)中的 A_8 和 A_9 引脚连接到 U_2。两个跳线引脚均接地时,上面译码电路仍然产生 330H～333H 的端口译码信号,而当两个跳线引脚均接“1”时,则上面译码电路会产生 030H～033H 的端口译码信号。同理还可以产生 130H～133H,230H～233H 的译码信号。

由于读、写操作不会同时进行,一个输入端口和另一个输出端口可以使用同一个地址代码。例如,可安排数据输入端口、数据输出端口使用同一个地址 330H,命令端口和状态端口共同使用地址 331H,则图 3-1(a)中右端的信号组合电路可作如图 3-2 的修改。

图中,$\overline{Y}_0$ 和 $\overline{Y}_1$ 是译码器输出的两个地址选择信号。需要注意的是,数据输入端口和数据输出端口虽然使用相同的地址,但却是两个各自独立的不同的端口。

8086 工作于最大模式时,由 8288 总线控制器发出$\overline{IORC}$,$\overline{IOWC}$代替上面的 M/$\overline{IO}$、$\overline{WR}$和$\overline{RD}$。读者不妨自己设计一个相应的地址译码电路。



(a) 地址译码电路　　(b) 跳线器

图 3-1　端口的地址译码电路

3. 数据锁存器与缓冲器

在微型计算机的系统数据总线上，连接着许多能够向 CPU 发送数据的设备，如内存储器、外部设备的数据输入端口等。为了使系统数据总线能够正常地进行数据传送，要求所有的这些连接到系统数据总线的设备具有三态输出的功能。也就是说，在 CPU 选中该设备时，它能向系统数据总线发送数据信号。在其他时间，它的输出端必须呈高阻状态。为此，所有的输入端口必须通过三态缓冲器与系统总线相连。

图 3-3 中，输入设备在完成一次输入操作后，在输出数据的同时，产生数据选通信号，把数据打入 8 位锁存器 74LS273。锁存器的输出信号通过 8 位三态缓冲器 74LS244 连接到系统数据总线。数据端口读信号由地址译码电路产生。该信号为高电平（无效）时，缓冲器输出端呈高阻态。一旦该端口被选中，数据端口读变为低电平（有效），已锁存的数据就可以通过 74LS244 送往系统数据总线继而被 CPU 所接收。

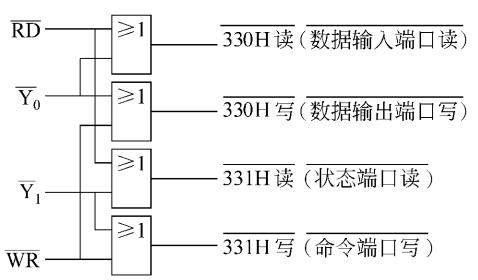


图 3-2　修改后的译码电路

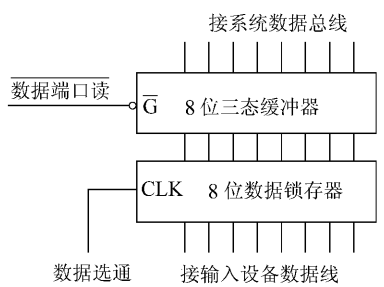


图 3-3　输入设备接口的数据锁存和缓冲电路

如果输入设备自身具有数据的锁存功能。输入接口内可以不使用锁存器。输入设备的数据线可通过三态缓冲器直接与系统数据总线相连接。但是，由于系统总线的工作特点，输入接口中的三态缓冲器是绝对不能省略的。

CPU 送往外部设备的数据或命令。一般应由接口进行锁存，以便使外部设备有充分

的时间接收和处理。图 3-4 是一个 8 位输出锁存电路的例子。

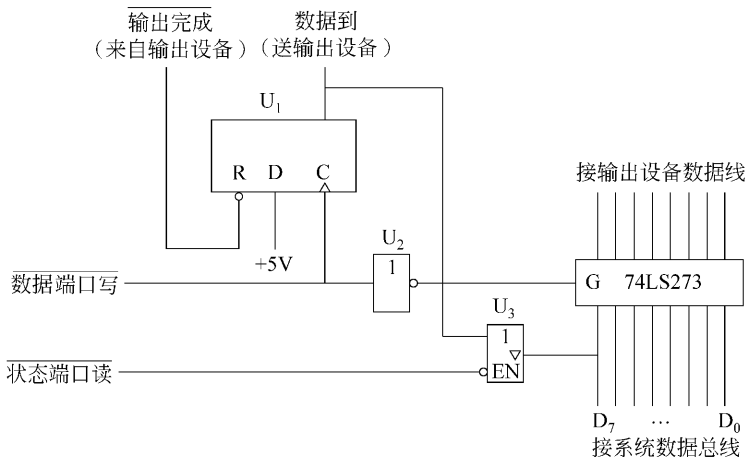


图 3-4　输出锁存电路

由地址译码电路产生的数据端口写选通信号是一个负脉冲，经 U_2 反相后把来自系统数据总线的 8 位数据 $D_0 \sim D_7$ 打入 8 位寄存器 74LS273，经输出端 $Q_0 \sim Q_7$，送往外部设备。数据端口写信号同时把 D 触发器（U_1）置“1”，通过 Q 端发出“数据到”信号，通知外部设备及时取走输出的一字节数据。外部设备在输出完成之后，向接口回送一个输出完成负脉冲. 将 D 触发器（U_1）清“0”，准备接收下一个数据。外部设备在接收和输出数据期间，D 触发器 Q 端保持为“1”。所以，它同时也成为该设备的状态信号 BUSY。图 3-4 中该信号通过三态缓冲器（U_3）连接到双向数据总线 D_7，三态缓冲器由地址译码获得的状态端口读信号控制。CPU 通过对状态端口的读指令，在 D_7 上可以获知它的状态。在该位为“1”时. CPU 不能向数据输出端口发送新的数据，否则将发生覆盖错误。

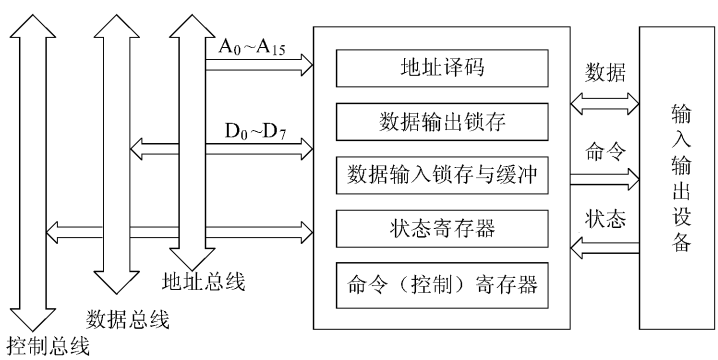


图 3-5　简单接口的组成

输出数据的锁存因设备而异。有的输出设备自身具有数据锁存功能，这时接口内也可以不设置锁存电路，把系统数据总线直接连接到输出设备的数据输入端，地址译码得到的数据端口写则用作输出数据的选通信号。

综上所述，把地址译码、数据锁存与缓冲、状态寄存器、命令寄存器各个电路组合起

来，就构成一个简单的输入输出接口。它一方面与系统地址总线 $A_0 \sim A_{15}$、数据总线 $D_0 \sim D_7$，控制总线 M/$\overline{IO}$、$\overline{RD}$、$\overline{WR}$（最小模式时）或$\overline{IOWC}$、$\overline{IORC}$（最大模式时）相连接，另一方面又与外部设备相连。由于常用的字符输入输出设备均使用8位数据，上述例子中均使用8位的数据输入输出端口。对于16位的I/O设备，其接口的基本原理是相同的。

3.2 输入输出数据传输的控制方式

CPU与外界进行两种主要类型的数据传输：与内存储器的数据传输和与外部设备的数据传输。CPU使用一个总线周期就可以与内存储器进行一次数据传输，而且这个过程可以连续进行。CPU与外部设备的数据传输则要复杂得多。CPU从输入设备读入一个数据之后，要等到该设备完成了第二次数据输入之后，才能读入第二个数据。等待的时间不但与该设备的工作速度有关，有时也带有许多随机的成分。例如，用户在键盘输入过程中，两次击键的间隔时间往往是不确定的。因此，较之与内存储器的数据传输，CPU与外部设备的数据传输有着不同的特点，因而也有着不同的处理方式。概括起来有以下3种传送方式：程序方式、中断方式和DMA方式。

3.2.1 程序方式

程序方式传送是指在程序控制下进行信息传送，具体实现又可分为无条件传送和条件传送两种方式。

1. 无条件传送方式

一些简单的I/O设备，对它们的I/O操作可以随时进行。例如，一些设备常用一组开关指示设备的配置情况和操作人员设定的工作方式：每个开关只有两种不同状态（ON和OFF，对应于“1”和“0”），它与某个输入端口中的一位（bit，b）相对应。程序员可以随时用输入指令读取该端口内每个开关的状态，而无须考虑它的“状态”。这一类简单设备的输入信号一般不需要锁存，可以通过三态缓冲器与系统数据总线直接相连（图3-6）。

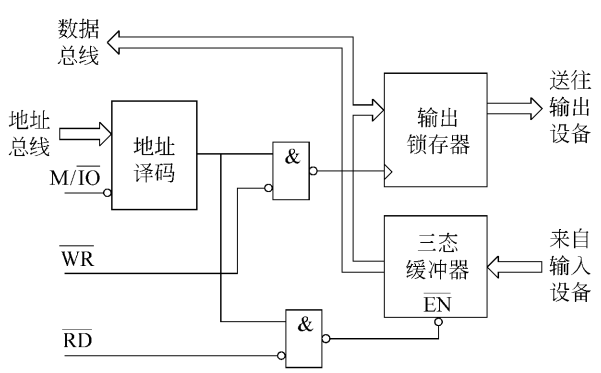


图3-6 无条件输入输出传送接口

一些简单的输出设备也有类似的情况。例如，常常用一组发光二极管（LED）来指示设备当前的工作状态。每一个LED对应于输出端口的一位（bit，b）。它的亮暗代表某个

设备两个特定的状态。例如，某 LED 亮表示 2＃电动机已通电运转，暗表示该电动机未通电等。这样的输出信号通常要在接口内进行锁存，以便在新的输出到来之前保持现在的输出状态。

图 3-6 给出了无条件传送时接口的组成方式，它是 3.1 节中介绍的几个部分的简单组合。

2. 条件传送方式

条件传送也称为查询式传送。使用条件传送方式时，CPU 通过程序不断读取并测试外部设备的状态。如果输入设备处于准备好状态，或者输出设备处于空闲状态，则 CPU 执行输入指令或输出指令与外部设备交换数据。为此，接口电路除了有传送数据的端口以外，还应有传送状态的端口。对于输入过程来说，外部设备将数据准备好时，将接口内状态端口的“准备好”标志位(READY)置“1”。对于输出过程来说，外部设备取走一个数据后，接口便将状态端口的对应标志位清“0”。表示当前输出寄存器已经处于“空”状态，可以接收下一个数据。

可见，对于条件传送来说，一个数据的传送过程由 3 个环节组成：

(1) CPU 从接口中读取状态字；

(2) CPU 检测状态字的对应位是否满足“就绪”条件，如果不满足，则回到前一步重新读取状态字；

(3) 如状态字表明外部设备已处于“就绪”状态，则传送数据。

图 3-7 展示了查询方式的输入接口电路。输入设备在数据准备好以后向接口发一个选通信号。这个选通信号有两个作用：一方面将外部设备的数据送到接口的锁存器中；另一方面使接口中的 D 触发器置“1”，它的输出端经过三态缓冲器和数据总线的某一根相连。数据信息和状态信息从不同端口经过数据总线送到 CPU。按照数据传送过程的 3 个步骤，CPU 从外部设备输入数据时先读取状态字(本例中状态字仅一位有效)，检查状态字是否表明数据准备就绪，即数据是否已进入接口的锁存器中。如准备就绪，则执行输入指令读取数据。读取数据的同时把 D 触发器清“0”，设备又恢复到“未就绪”状态，本次数据传输到此结束。

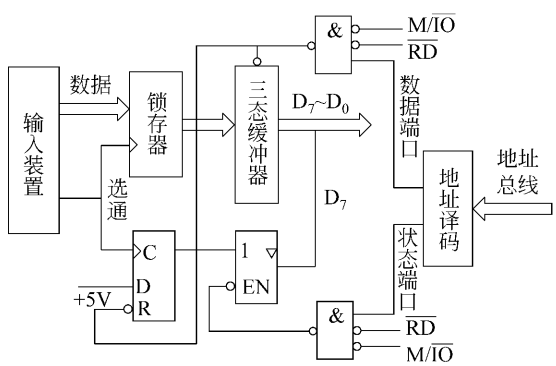


图 3-7　查询式输入接口电路

相应的汇编语言指令如下：

```
AGAIN: IN          AL, STATUS_PORT          ; 读状态端口，D7= 1 表示“数据就绪”
```

```
TEST        AL, 80H                ; 测试"数据就绪"位
JZ          AGAIN                  ; 未就绪,继续读状态端口
IN          AL, DATA_PORT          ; 已就绪,从数据端口读取数据
 ⋮
```

用C语言编写如上过程,则更为简便:

```
do { stat= inportb(status_port); }
    while (stat & 0x80==0);              /* 数据未准备好则反复读状态 */
data= inportb(data_port);                /* 数据已准备好则读取数据 */
```

图3-8展示了查询方式的输出接口电路。CPU需要向一个外部设备输出数据时,先读取接口中的状态字,如果状态字表明外部设备空闲(或"不忙"),说明可以向外部设备输出数据,此时CPU才执行数据输出指令,否则CPU必须等待。

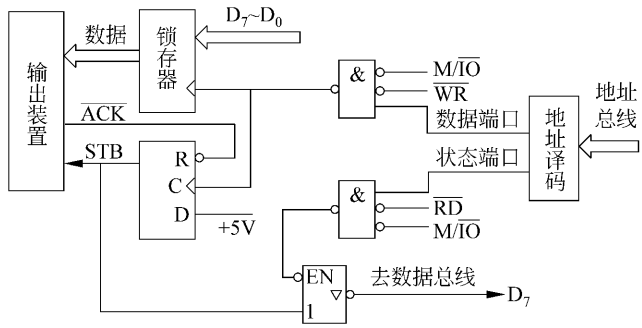


图3-8 查询式输出接口电路

CPU执行数据输出指令时,地址译码电路产生的数据锁存信号将数据总线上的数据打入接口数据锁存器,同时将D触发器置"1"。D触发器的输出信号一方面为外部设备提供一个联络信号STB,告诉外部设备现在接口中已有数据可供提取;另一方面,也用作该设备的状态标志"忙"(BUSY)。CPU读取状态端口后可以得知该外部设备处于"忙"状态,从而阻止输出新的数据。输出设备从接口取走数据,或者完成了本次输出后,通常会回送一个应答信号$\overline{ACK}$。该信号使接口内的D触发器清"0",也是把状态位清"0",这样就可以开始下一个输出过程。

相应的汇编语言程序如下:

```
ONE: IN     AL, STATUS_PORT         ; 读状态端口
     TEST   AL, 80H                 ; 测试"忙"位
     JNZ    ONE                     ; 忙,再读状态端口
     MOV    AL, DAT                 ; 不忙,取来数据
     OUT    DATA_PORT, AL           ; 数据送入数据端口
     ⋮
```

相应的C语言程序如下:

```
do { stat= inportb(status_port); }
```