第 5 章　微型计算机和外设的数据传输

5.1　为什么要用接口

要构成一个实际的微型计算机系统，除了微处理器以外，还需要各种接口。接口按功能分为两类：一类是使 CPU 正常工作所需要的辅助电路，通过这些辅助电路，使 CPU 得到时钟信号或接收外部的多个中断请求等；另一类是输入/输出接口，利用这些接口，CPU 可接收外设送来的信息或将信息发送给外设。

最常用的外设如键盘、显示器、打印机、磁盘驱动器等都是通过输入/输出接口和总线相连，完成检测和控制的仪器仪表也属于外设之列，通过接口和主机相连。

外设为什么一定要通过接口和主机总线相连呢？能不能将外设和 CPU 的数据总线、地址总线及控制总线直接连接呢？

从时序上看，CPU 对外设的输入/输出操作和对存储器的读/写操作很类似，那么，是什么原因决定了存储器不需要接口，可以直接连在总线上，而输入/输出设备却一定要通过接口与总线相连呢？

为回答这两个问题，需要分析外设的输入/输出操作和存储器读/写操作的不同之处。

所有存储器都是用来保存信息的，功能单一；存储器品种也有限，只有只读类型和可读/可写类型；此外，存储器的存取速度基本上可和 CPU 的工作速度匹配。这些决定了存储器可通过总线和 CPU 相连，即通常说的直接将存储器挂在 CPU 总线上。

但是，外设的功能却是多种多样的。有些外设作为输入设备，有些外设作为输出设备，也有些外设既作为输入设备又作为输出设备，还有一些外设作为检测设备或控制设备，而每一类设备可能又包括了多种工作原理不同的具体设备。对于一个具体设备来说，它所使用的信息可能是数字式的，也可能是模拟式的，而非数字式信号必须经过转换，使其成为数字信号才能送到计算机总线。这种将模拟信号变为数字信号或者反过来将数字信号变为模拟信号的功能是 A/D、D/A 接口来完成的。

大多数外设所用的信息是数字式的，但其中，有些外设的信息是并行的，有些外设的信息是串行的。串行设备只能接收和发送串行的数字信息，而 CPU 却只能接收和发送并行信息。这样，串行设备必须通过接口将串行信息变为并行信息，才能送给 CPU；反过来，要将 CPU 送出的并行信息变为串行信息，才能送给串行设备。这种双向的转换都由串行接口来完成。可见接口也起到并行数据和串行数据的转换作用。

这么说，如果一个微型机系统中连接的是并行设备，是否可不用接口了呢？也不是。因为 CPU 通过总线要和多个外设打交道，而在同一时刻 CPU 通常只和一个外设交换信息，也就是说，一个外设不能长期和 CPU 相连，只有被 CPU 选中的外设，才接收数据总线上的数据或将外部信息送到数据总线上。所以，即使是并行设备，也同样要通过接口与总线相连。这种接口就是 6.2.6 节要介绍的并行接口。

除了上面这些原因外，外设的工作速度通常比 CPU 的速度低得多，而且各种外设的工

作速度互不相同，这就要求接口对输入/输出过程能起一个缓冲和联络的作用。

对于输入设备来说，接口通常起信息转换和缓冲作用。转换的含义包括模拟量到数字量的转换、串行数据往并行数据的转换以及电平转换等，总之，目的是将输入设备送来的信息转换成 CPU 能接收的格式，并将其放在缓冲器中让 CPU 接收。对于输出设备来说，接口要将 CPU 送来的并行数据放到缓冲器中，并将它变成外设所需要的信息形式，这种形式可能是串行数据，也可能是模拟量等。

可见，输入/输出接口是为了解决计算机和外设之间的信息转换问题而提出来的，输入/输出接口是计算机和外设之间传送信息的部件，每个外设都要通过接口和主机系统相连。接口技术就是专门研究 CPU 和外设之间的数据传送方式、接口的工作原理和使用方法的，以下将逐步讨论这些问题。

5.2 CPU 和输入/输出设备之间的信号

为了说明 CPU 和外设之间的数据传送方式，应该先了解 CPU 和输入/输出设备之间传输信号的分类。通常，CPU 和输入/输出设备之间有以下几类信号。

5.2.1 数据信息

CPU 和外设交换的基本信息就是数据。数据信息大致分为如下 3 种类型。

1. 数字量

数字量是指从键盘、磁盘驱动器等读入的信息，或主机送给打印机、磁盘驱动器、显示器及绘图仪的信息，它们是二进制形式的数据或是以 ASCII 码表示的数据及字符。

2. 模拟量

如果一个微型机系统是用于控制的，那么，多数情况下的输入信息是现场的连续变化的物理量，如温度、湿度、位移、压力、流量等，这些物理量一般通过传感器先变成电压或电流，再放大。电压和电流仍是连续变化的模拟量，而计算机无法直接接收和处理模拟量，要经过模拟量往数字量(A/D)的转换，变成数字量，才能送入计算机。反过来，计算机输出的数字量要经过数字量往模拟量(D/A)的转换，变成模拟量，才能控制现场。

3. 开关量

开关量可表示两个状态，如开关的闭合和断开、电机的运转和停止、阀门的打开和关闭等，这样的量只要用 1 位二进制数表示即可。

上面这些数据信息，一般由外设通过接口传递给系统。在输入过程中，数据信息由外设经过外设和接口之间的数据线进入接口，再到达系统的数据总线，从而送给 CPU。在输出过程中，数据信息从 CPU 经过数据总线进入接口，再通过接口和外设之间的数据线送到外设。外设和接口之间的数据信息可以是串行的，也可以是并行的，相应地要使用串行接口或并行接口。

5.2.2 状态信息

状态信息反映了当前外设所处的工作状态，是外设通过接口往 CPU 传送的。对于输入设备来说，通常用“准备好”(READY)信号来表明输入的数据是否准备就绪；对于输出设

备来说，通常用忙(BUSY)信号表示输出设备是否处于空闲状态，如为空闲状态，则可接收CPU送来的信息，否则CPU要等待。

5.2.3 控制信息

控制信息是CPU通过接口传送给外设的，以此控制外设的工作，外设的启动信号和停止信号就是常见的控制信息。控制信息往往随着外设的具体工作原理不同而含义不同。

从含义上说，数据信息、状态信息和控制信息各不相同，应该分别传送。但在微型机系统中，CPU通过接口和外设交换信息时，状态信息、控制信息也被广义地看成是一种数据信息，即状态信息作为一种输入数据，而控制信息作为一种输出数据。这样，状态信息和控制信息也通过数据总线来传送。但在接口中，这3种信息进入不同的寄存器。具体说，CPU送往外设的数据或外设送往CPU的数据放在接口的数据缓冲器中，从外设送往CPU的状态信息放在接口的状态寄存器中，而CPU送往外设的控制信息送到接口的控制寄存器中。

5.3 接口部件的I/O端口

每个接口部件都包含一组寄存器，如图5.1所示，CPU和外设进行数据传输时，各类信息在接口中进入不同的寄存器，一般称这些寄存器为I/O端口，每个端口有一个端口地址。

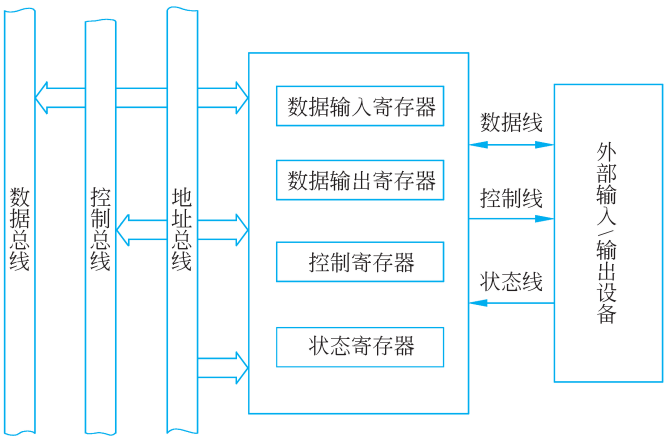

图5.1 外设通过接口和系统的连接

有些端口是用于对来自CPU和内存的数据或对送往CPU和内存的数据起缓冲作用的，这些端口称为数据端口。还有一些端口用来存放外设或接口部件本身的状态，称为状态端口。CPU通过对状态端口的访问可检测外设当前的状态。第三类端口用来存放CPU发出的命令，以便控制设备的动作，这类端口称为控制端口或命令端口。可以说，计算机主机和外设之间都是通过接口部件的I/O端口来沟通的。

对I/O端口有两种编址方式：与存储器统一编址方式、I/O端口独立编址方式。

对内存和I/O端口统一编址时，系统中只有一个统一的地址空间，这样，所有访问内存的指令也都能访问I/O端口。在I/O端口独立编址的系统中，要建立两个地址空间：一个为内存地址空间；另一个为I/O地址空间。通过控制总线来确定CPU到底要访问内存还是I/O端口。为确保控制总线发出正确的信号，系统提供了专用于和I/O端口通信的

输入/输出指令。

应该指出，不管是输入还是输出，所用到的地址总是对端口而言的，而不是对接口部件而言的。一个双向工作的接口芯片通常有 4 个端口，即数据输入端口、数据输出端口、状态端口和控制端口。因为数据输入端口和状态端口是“只读”的，数据输出端口和控制端口是“只写”的，所以，系统为了节省地址空间，往往将数据输入端口和数据输出端口对应一个端口地址，CPU 用此地址进行读操作时，实际上是从数据输入端口读取数据，而当 CPU 用此地址进行写操作时，实际上是往数据输出端口写入数据。同样，状态端口和控制端口也用同一个端口地址。

可见，有了端口地址，CPU 对外设的输入/输出操作归结为对接口芯片各端口的读/写操作。

5.4 接口的功能以及在系统中的连接

5.4.1 接口的功能

简单地说，一个接口的基本功能是在系统总线和 I/O 设备之间传输信号，提供缓冲作用，以满足接口两边的时序要求。下面是从广义的角度概括出来的接口功能，对于一个具体的接口来说，未必全部具备，但必定具备其中的几个。

1. 寻址功能

首先，接口要能够识别选择存储器和 I/O 端口的信号；此外，要能够识别片选信号，以便判断当前本接口是否被访问；如果受到访问，还要决定是接口中哪个寄存器被访问。

2. 输入/输出功能

接口要根据送来的读/写信号决定当前进行的是输入操作还是输出操作，并且随之能从总线上接收来自 CPU 的数据和控制信息，或将数据或状态信息送到总线上。

3. 数据转换功能

接口不但要从外设输入数据或将数据送往外设，并且要把 CPU 输出的并行数据转换成所连的外设可接收的格式（如串行格式）；或者反过来，把从外设输入的信息转换成并行数据送往 CPU。

4. 联络功能

当接口从总线上接收一个数据后，或者在把一个数据送到总线上后，能发一个就绪信号，以通知 CPU，数据传输已经完成，从而可准备进行下一次传输。

5. 中断管理功能

作为中断控制器的接口应该具有往 CPU 发送中断请求信号和从 CPU 接收中断响应信号的功能，而且还有往 CPU 发送中断类型号的功能。此外，一些接口还应该具有优先级管理功能。

6. 复位功能

接口应该能接收复位信号，从而能使接口本身以及所连的外设重新启动。

7. 可编程功能

为了使一个接口可以工作于不同的方式，而且可用软件来决定到底工作于哪一种方式，

此外，还为了能够用软件来设置控制信号，一个接口应该有可编程功能，即可以通过程序控制其工作方式、修改其状态标志等。几乎所有大规模集成电路接口芯片都具有这个功能。

8. 错误检测功能

在接口设计中，常常要考虑对错误的检测问题。多数可编程接口芯片能检测下列两类错误。

一类是传输错误。因为接口和设备之间的连线常常受各种干扰，从而引起传输错误，所以一般接口采用奇/偶校验位对传输错误进行检测。传输时，如用奇校验，那么使信息中 1 的数目（包括校验位）为奇数。也就是说，所传输的数据中如 1 的个数为奇数，则校验位为 0；所传输的数据中如 1 的个数为偶数，则校验位为 1。这样，在传输一个数据时，连同校验位，1 的总数目总是为奇数。同样的道理，如用偶校验，那么，信息中 1 的数目（包括校验位）总是为偶数。如发现有奇/偶校验错误，则对状态寄存器中的相应位进行设置。而状态寄存器的内容可通过程序进行读取和检测。

另一类是覆盖错误。当计算机输入数据时，实际上是从接口的输入缓冲器中取数。如果 CPU 还没有取走数据，输入缓冲器由于某种原因又被装上了新的数据，那么，就会产生覆盖错误。在输出时，也会有类似的情况，即输出缓冲寄存器中的数据在被外设取走以前，如果 CPU 又往接口输出一个新的数据，那么，原来的数据就被覆盖了。在产生覆盖错误时，接口也会在状态寄存器中设置相应的状态位。

5.4.2 接口与系统的连接

图 5.2 是一个典型的 I/O 接口和外部电路连接图，右边的大框代表接口部件。各种具体接口的内部结构和功能随所连的 I/O 设备的不同而差别很大，但从结构上看，都可把一个接口分为两部分：第一部分用来和 I/O 设备相连；第二部分用来和系统总线相连。其中，前一部分的结构是和 I/O 设备的传输要求及数据格式有关的，所以，各接口之间互不相同，例如，对串行接口和并行接口来说，这部分的差别就很大。不过，所有接口与总线相连的那部分的结构非常类似，原因很简单，因为这些接口要连在同一总线上。

为了支持接口的工作，系统中通常有总线收发器和相应逻辑电路。逻辑电路把 CPU 的控制信号翻译成接口所需要的联络信号。联络信号随接口的不同而异。典型的外部逻辑电路应能接收 CPU 送来的读/写信号，以便决定数据传输方向。对于比较小的系统来说，可省去总线收发器，因为大多数接口部件内部都带有一定驱动能力的总线驱动电路。

系统中还必须有地址译码器，以便将总线提供的地址翻译成对接口的片选信号。地址译码器除了接收地址信号外，还需要把 CPU 提供的用来区分 I/O 地址空间和内存地址空间的信号 $M/\overline{IO}$用于译码过程。

如果地址译码器确定了某一个接口要被访问，那么，会使此接口得到有效的片选信号。一个接口通常有若干寄存器可读/写，由此，还要指出访问哪个寄存器。在实际使用时，一般用一两位低位地址结合读/写信号来实现对接口内部寄存器的寻址。

例如，一个接口内部有两个可读寄存器，称为 A、B，另外还有两个可写寄存器，称为 C、D，那么，在片选信号和 $M/\overline{IO}$信号有效后，用写信号、读信号和地址 A_0 就可以将 4 个内部寄存器加以区分，具体如表 5.1 所示。

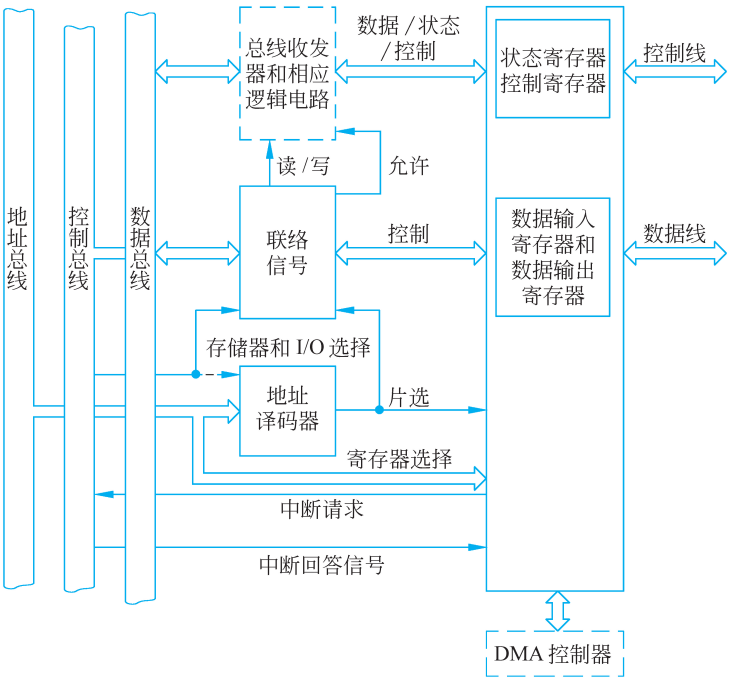

图 5.2　典型的 I/O 接口和外部电路连接

表 5.1　对 4 个内部寄存器的寻址

写信号	读信号	A_0	被访问的寄存器
0	1	0	A
0	1	1	B
1	0	0	C
1	0	1	D

5.5　CPU 和外设之间的数据传送方式

各种外设的工作速度相差很大，有些相当高，如硬盘的内部传输率达百兆字节每秒，而有些外设却由于机械和其他因素所致速度相当低，如键盘是用于人工输入数据的，通常速度为几十毫秒输入 1 字节。这样，CPU 何时从输入设备读取数据以及何时往输出设备写入数据，就成为较复杂的定时问题。

概括起来，有如下 3 种传送方式解决上述问题：程序方式、中断方式、DMA 方式。

5.5.1　程序方式

程序方式是指在程序控制下进行信息传送，又分为无条件传送方式和条件传送方式。

1. 无条件传送方式

如果 CPU 能够确信一个外设已经准备就绪，那就不必查询外设的状态而可直接进行信息传输，这称为无条件传送方式。

在无条件传送方式下，程序设计较简单。不过，名为无条件传送，实际上是有条件的，那

就是传送不能太频繁，以保证每次传送时，外设处于就绪状态。所以无条件传送方式用得较少，只用在对一些简单外设的操作，如开关、七段显示管等。

由于简单外设作为输入设备时，输入数据保持时间相对于 CPU 的处理速度要长得多，所以可直接使用输入缓冲器和数据总线相连，如图 5.3 所示。当 CPU 执行输入指令时，读信号$\overline{RD}$有效，选择信号 M/$\overline{IO}$处于低电平，因而输入缓冲器被选通，使其中早已准备好的输入数据进入数据总线，再到达 CPU。可见，要求 CPU 在执行输入指令时，外设的数据是准备好的，即已经存在输入缓冲器中，否则出错。

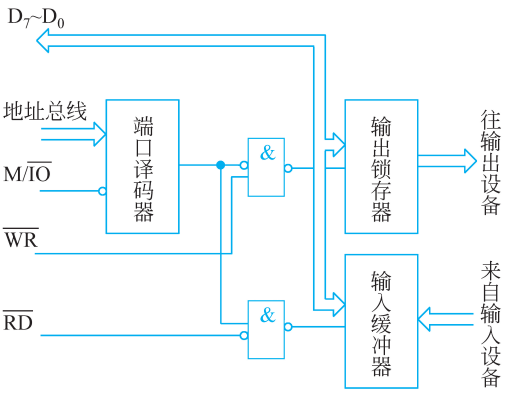

图 5.3　无条件传送方式的接口的工作原理

简单外设作为输出设备时，一般都需要锁存器，也就是说，要求 CPU 送出的数据在接口的输出端保持一段时间。其原因仍然是由于外设的速度比较慢，所以，要求 CPU 送到接口的数据能保持和外设动作相应的时间。如图 5.3 所示，CPU 执行输出指令时，M/$\overline{IO}$和$\overline{WR}$信号有效，于是，接口中的输出锁存器被选中，CPU 输出的信息经过数据总线送到输出锁存器，输出锁存器保持这个数据，直到外设取走。显然，这里要求 CPU 在执行输出指令时，确信所选中的输出锁存器是空的。

2. 条件传送方式

条件传送也称查询式传送。用条件传送方式时，CPU 通过执行程序不断读取并测试外设的状态，如外设处于“准备好”状态（输入设备）或空闲状态（输出设备），则 CPU 执行输入指令或输出指令与外设交换信息。为此，接口部件除了有传送数据的端口以外，还有传送状态的端口。对于输入过程来说，当外设将数据准备好时，则使接口的状态端口中的“准备好”标志位置 1；对于输出过程来说，外设取走一个数据后，接口便将状态端口中的对应标志位置 1，表示当前输出寄存器已处于“空”状态，可接收下一个数据。

可见，对于条件传送来说，一个数据传送过程由以下 3 个环节组成：

（1）CPU 从接口中读取状态字。

（2）CPU 检测状态字对应位是否满足“就绪”条件，如不满足，则返回前一步读取状态字。

（3）如状态字表明外设已处于“就绪”状态，则传送数据。

图 5.4 表明了用查询式传送进行输入的接口的工作原理。输入设备在数据准备好后便往接口发一个选通信号。这个选通信号有两个作用：一方面将外设的数据送到接口的锁存器中；另一方面使接口中的一个 D 触发器输出 1，从而使接口中三态缓冲器的 READY 位置

1。数据信息和状态信息从不同的端口经过数据总线送到 CPU。按数据传送过程的 3 个步骤，CPU 从外设输入数据时先读取状态字，检查状态字看数据是否准备就绪，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，此时，状态位清 0，这样，便开始下一个数据传输过程。

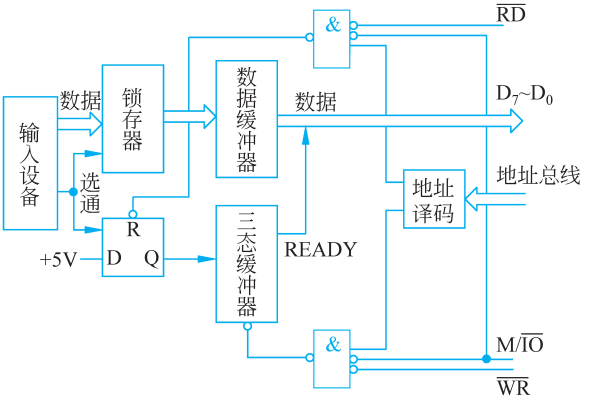

图 5.4　用查询式传送进行输入的接口的工作原理

图 5.5 表明了用查询式进行输出的接口的工作原理。当 CPU 要往一个外设输出数据时，先读取接口中的状态字，如状态字表明外设有空（或“不忙”），则说明可往外设输出数据，此时 CPU 执行输出指令，否则 CPU 必须等待。

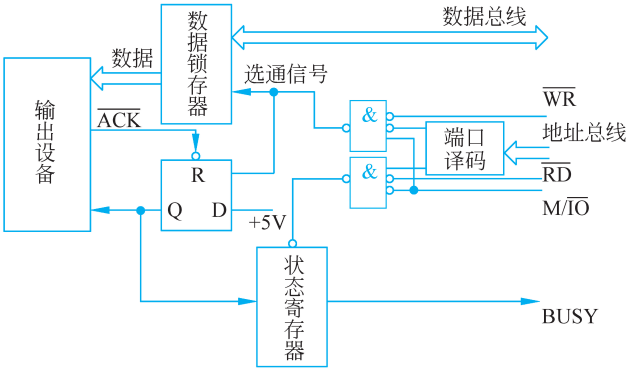

图 5.5　用查询式传送进行输出的接口的工作原理

CPU 执行输出指令时，由 M/$\overline{IO}$和$\overline{WR}$产生的选通信号将数据总线上的数据送到数据锁存器，同时使 D 触发器输出 1。D 触发器一方面为外设提供联络信号，告诉外设现在接口中已有数据可提取；另一方面，D 触发器使状态寄存器的对应标志位置 1（有些设备用“忙”标志表示状态，有些设备用“空”标志表示状态，两者有效电平相反）告诉 CPU，当前外设处于“忙”状态，从而阻止 CPU 输出新的数据。

当输出设备从接口中取走数据后，通常会送一个应答信号$\overline{ACK}$，$\overline{ACK}$使接口中的 D 触发器置 0，从而使状态寄存器中的对应标志位置 0，这样就可开始下一个输出过程。

图 5.6 是一个查询式输入过程的流程图。图中假定要输入 1 字节串或 1 个字串，每字节或字被送到 CPU 以后进行一定的处理，然后再送到内存缓冲区，当所有的数据都输入完毕并送到缓冲区后，再对缓冲区中的数据进行处理。

下面举一个使用查询式输入/输出方式的实例。

假设从键盘往缓冲区输入 1 个字符行，当遇到回车符(0DH)或字符行超过 80 个字符时，输入便结束，并自动加上 1 个换行符(0AH)。如在输入的 81 个字符中未见到回车符，则在终端上输出信息“BUFFER OVERFLOW”。

因为键盘往 CPU 输入的是 ASCII 码，而 ASCII 码采用 7 位二进制数据表示，所以，用第 7 位(即最高位)作为键盘往 CPU 传送时的校验位，这里假定用偶校验。如校验出错，则输出错误信息；如没有校验错误，则先清除校验位，再传送到缓冲区。

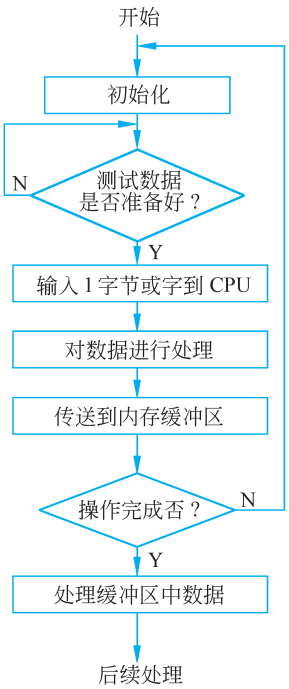

图 5.6　查询式输入过程的流程图

假定接口的数据输入端口地址为 0052H，数据输出端口地址为 0054H，状态端口地址为 0056H，并且假定状态寄存器中第 1 位为 1，表示输入缓冲器中已经有 1 字节准备好，可进行输入。此外，还假定状态寄存器的第 0 位为 1，表示输出缓冲器已经腾空，因而 CPU 可往终端输出数据。当然，后面两个假设都是有条件的，即在设计接口部件时，使状态寄存器的第 1 位在接口从设备输入 1 字节时，便自动置 1，而当 CPU 从接口读取 1 字节时，便自动置 0；相类似的，当 CPU 往接口输出 1 字节时，状态寄存器的第 0 位自动置 0，而当 1 字节从接口输出到设备时，则自动置 1。具体程序如下：

```
DATA_SEG    SEGMENT
MESSAGE     DB          'BUFFER OVERFLOW',0DH,0AH
            ⋮
DATA_SEG    ENDS
COM_SEG     SEGMENT
BUFFER      DB          82   DUP (?)                ;接收缓冲区
COUNT       DW          ?                           ;计数器
COM_SEG     ENDS
            ⋮
CODE        SEGMENT
ASSUME      DS:DATA_SEG,ES:COM_SEG,CS:CODE
STAT:       MOV         AX,DATA_SEG                 ;对 DS 作初始化
            MOV         DS,AX
            MOV         AX,COM_SEG                  ;对 ES 作初始化
            MOV         ES,AX
            MOV         DI,OFFSET BUFFER            ;计数器指向缓冲区首址
            MOV         COUNT,DI
            MOV         CX,81                       ;字符行长度
            CLD                                     ;清方向标志
NEXT_IN:    IN          AL,56H                      ;读入状态
            TEST        AL,02H                      ;测状态寄存器第 1 位
            JZ          NEXT_IN                     ;未准备好，则等待，再测
            IN          AL,52H                      ;准备好，则输入字符
```

```
                OR      AL,0                    ;校验
                JPE     NO_ERROR                ;校验正确,则转 NO_ERROR
                JMP     ERROR                   ;校验出错误,则转 ERROR 程序
NO_ERROR:       AND     AL,7FH                  ;清除校验位
                STOSB                           ;将字符送缓冲区
                CMP     AL,0DH                  ;是否为回车符
                LOOPNE  NEXT_IN                 ;不是回车,则再输入
                JNE     OVERF                   ;不是回车且溢出,则转 OVERF
                MOV     AL,0AH                  ;加一个换行符
                STOSB                           ;存入缓冲区
                SUB     DI,COUNT                ;计算输入字符数
                MOV     COUNT,DI                ;COUNT 中为输入字符数
                ⋮
OVERF:          MOV     SI,OFFSET MESSAGE       ;SI 中为字符串首址
                MOV     CX,17                   ;字符数
NEXT_OUT:       IN      AL,56H                  ;读状态寄存器
                TEST    AL, 01H                 ;测状态寄存器第 0 位
                JZ      NEXT_OUT                ;如没有就绪,则再测
                LODSB                           ;将字符取到 AL 中
                OUT     54H,AL                  ;输出字符
                LOOP    NEXT_OUT                ;输出下一个字符
                ⋮
```

对上面的程序作以下 5 点说明：

(1) 程序中用 ES 和 DI 作为段寄存器和变址寄存器指向输入缓冲区，CX 作为控制循环次数的寄存器，一开始，CX 寄存器中设置为字符行的最大长度。

(2) 方向标志 DF 清 0 是为了使 STOSB 指令和 LODSB 指令在执行时，地址值自动加 1，否则，地址会按减量修改。

(3) NEXT_IN 标号后面的 3 条指令是用来测试接口状态寄存器的，如状态寄存器的值表明端口未准备好，则用循环测试来等待，直到规定的状态位为 1 才退出循环。

(4) 奇/偶校验是通过把输入字符和 0 相“或”设置奇/偶标志 P，然后对 P 标志进行判断来实现的，这样做很简单，不需要另外设计子程序进行校验。校验完成后，将结果和 7F 相“与”，以清除奇/偶校验位，再送到输入缓冲区。

(5) 当用 CMP 指令判断遇到的字符为回车符 0DH 时，程序紧接着再附加一个换行符 0AH，并存入缓冲区，这样做是为了实现程序的设计要求。

利用查询式进行输入/输出操作时，如果系统中有多个利用查询式实现输入/输出的设备，那该怎么处理呢？通常是用轮流查询检测接口的状态位来解决此问题。

假定一个系统中有 3 个输入设备，那么，可用下面的程序实现轮流查询的输入操作。

```
TREE_IN:        MOV     FLAG,0                  ;清除标志
INPUT:          IN      AL,STAT1                ;读入第一个设备的状态
                TEST    AL,20H                  ;是否准备就绪
                JZ      DEV2                    ;如否,则转 DEV2
                CALL    PROC1                   ;如准备就绪,则调 PROC1
                CMP     FLAG,1                  ;如标志被清除,则输入另一个数
```