第3章

芯片级接口电路设计的基本方法

本章导学：

本章介绍了接口设计的基本方法。特别从接口的两个界面出发，分析了接口与CPU连接界面以及与I/O设备连接界面的基本设计方法，并对中断接口技术进行了讨论。本章内容对以后各章的学习尤为重要，通过本章的学习，读者应对接口设计方法有一个初步的认识。

微型计算机系统可通过系统总线与外设相连，进行系统的扩展与开发。而外设必须通过接口才能与CPU交换信息。也就是说，系统的扩展就是将所需外设通过接口连于系统总线而完成的。因此I/O接口电路的设计技术是计算机应用开发人员必须掌握的技术。

从硬件角度看，接口部件可谓是计算机与外设之间的连接通道。如图3-1所示，I/O接口存在着两个界面，一是接口与计算机的连接界面；二是接口与外设的连接界面。

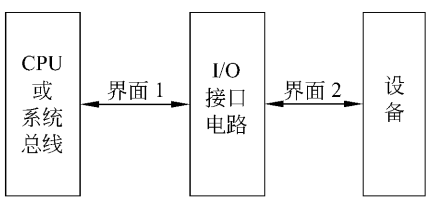

图3-1　I/O接口与计算机、外设之间的关系

接口与计算机的连接视界面的不同，有以下两种情况。

(1) 与CPU外部引脚连接界面

这是一种芯片级的接口技术，设计时只需掌握微处理器的各引脚信号及相应工作时序。

(2) 与系统总线连接的界面

这是一种系统级的接口技术，问题要复杂得多，它必须遵守微机系统设计时给出的关于系统资源使用的约定，以及关于系统总线标准的约定。

接口与外设的连接界面则根据所连接的外部设备功能的不同，亦可分为两类：

(1) 与系统外设连接的界面

与系统连接的外设往往是系统的基本配置，如显示器、磁盘驱动器、扬声器等。

(2) 与应用需要的专用外设连接的界面

为应用需要而连接的外设应根据用户的特定要求而配置，如图像处理、数据采集、语

音识别等。

但不管是两个界面中的哪一种情况,其接口电路的基本设计方法是类似的。下面将分别从接口的两个界面着手,讨论接口电路的基本设计方法。

3.1 I/O接口与计算机连接界面的设计

计算机访问外部设备时,首先要由地址总线给出I/O端口地址信号,以选中该单元。然后在相应的控制总线上发出有关进行读操作或写操作的控制信号,最后才能在数据总线上进行信息交流。所有这些操作都是通过总线与接口进行的。因此,接口电路与计算机的界面首先要完成总线的连接,包括地址总线、数据总线、控制总线的连接。在总线连接时,要考虑两个方面:即静态特性和动态特性。静态特性包括总线各信号相应的连接方法、逻辑电平的配合以及各信号线的静态负载能力,动态特性则指操作时序的约束条件。

3.1.1 总线连接的静态特性

1. 总线的连接

(1) 地址总线的连接

不管计算机系统采用哪种I/O端口寻址方式,接口电路中均应设有地址译码器,以便根据CPU提供的地址值来确定被访问的I/O端口。译码有多种方法,可采用简单的与非门译码,也可采用译码器或比较器进行译码,关键是要对接口所采用的I/O端口地址进行正确的选择。必须注意:端口地址只能选择那些系统保留的口地址,以避免产生冲突。

例 3-1 PC系列机采用的是独立I/O端口编址方式,其系统总线提供地址线A_9～A_0来寻址I/O端口,地址码200H～23FH为系统保留口地址。设要产生200H～203H;204H～207H;208H～20BH;214H～217H;218H～21BH这5个口地址来完成接口电路设计,请给出地址译码电路。

根据题意,可采用如图3-2所示译码电路。

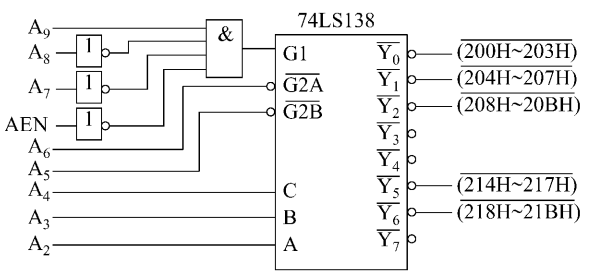

图 3-2 端口地址译码电路

A_9～A_2为地址信号,与相应地址总线相连。由于A_1、A_0没参加译码,因此译码器的每个输出值实际对应的是一段地址编码,这些编码对于同一个端口是完全等同的。在实

际使用中，往往用译码输出信号去选择某一个可编程接口芯片，使该芯片含有多个端口地址，然后由低位地址 A_1、A_0选择芯片中的某一端口电路。

电路将 A_9～A_5接 74LS138 的使能端，只有使能时译码器才工作。其输入与输出之间的译码关系如表 3-1 所示。

表 3-1　译码器输入输出关系

G_1			$\overline{G_2A}$	$\overline{G_2B}$	C	B	A			译码输出	口地址范围
A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		
1	0	0	0	0	0	0	0	X	X	Y_0	200H～203H
1	0	0	0	0	0	0	1	X	X	Y_1	204H～207H
1	0	0	0	0	0	1	0	X	X	Y_2	208H～20BH
1	0	0	0	0	0	1	1	X	X	Y_3	
1	0	0	0	0	1	0	0	X	X	Y_4	
1	0	0	0	0	1	0	1	X	X	Y_5	214H～217H
1	0	0	0	0	1	1	0	X	X	Y_6	218H～21BH
1	0	0	0	0	1	1	1	X	X	Y_7	

利用不同的地址输入连接，可完成不同的端口地址译码。同时译码时还应注意几点问题。

① DMA 传送时，应封锁端口地址译码。因为无论是 DMA 输出（存储器读、I/O 写）周期，或是 DMA 输入（存储器写、I/O 读）周期，控制总线上 I/O 读或写命令均有效。但此时地址总线送出的却是存储单元地址，这个地址可能和不是 DMA 操作的 I/O 端口地址一样，从而引起和内存地址相重合的端口地址译码选中，使设备发生误动作。为此可将 DMA 控制器发出的 AEN 信号加入译码，DMA 操作时该信号有效，封锁 I/O 端口地址译码输出。

② 如果接口电路中需要两个口地址，一个用于输入，一个用于输出，则可将 I/O 读、写命令加入译码，以实现读、写分别访问。这样，一个端口实际上就等效于两个端口地址了。

③ 译码时要注意门的延迟时间，若太长，有可能使读、写命令先有效，然后端口地址译码才有效，这样可能导致对别的端口地址进行读、写操作，从而产生误动作。

（2）数据总线的连接

接口的数据线应与系统总线各位对应连接。同时还应注意输出的数据是否要求具有锁存能力，挂接系统总线的还需要工作于三态等。

（3）控制总线的连接

系统总线提供的各类控制信号应根据接口电路的要求，正确地对应连接。

2. 总线的逻辑电平和负载能力

（1）逻辑电平的一致性

接口设计时须充分了解总线界面的输入输出电气特性。首先要满足逻辑电平的一致性要求，连接双方的工作电平必须吻合，否则，应配备相应的电平转换电路，进行电平

转换。

(2) 总线的静态负载能力

当用户设计 I/O 接口电路时,需考虑系统总线的负载能力。静态负载能力包括 TTL 及 MOS 电路对负载的扇出和扇入的能力,以及电路的长线驱动能力。若接口电路加到系统总线上的负载过载时,或传输速率较高、传输距离较远时,传输信号就会发生畸变甚至于丢失数据,这将使整个系统工作极为不稳定。

一般在设计中,需了解系统总线输出驱动器实际可驱动的能力,并扣去常用接口设备已占用的总线驱动功率,剩下的才是提供给用户设计接口的可用功率。这里特别注意的是 I_{OL} 和 I_{OH} 这两个主要负载指标,I_{OL} 是指在逻辑"0"状态时,流入驱动器的最大电流;I_{OH} 是指在逻辑"1"状态时,由该输出驱动器提供的最大电流。另外,系统总线输入的信号对接口电路而言,也是一个负载,该负载是建立在正常逻辑"0"电平和逻辑"1"电平的情况下,由驱动电路提供的电流 I_{IL} 或 I_{IH}。设计时同样要注意参考系统总线提供的输入信号负载能力指标。

当实际情况超出其负载能力时,在设计接口电路中,就必须要加相应的驱动电路,以增加信号的驱动能力,保证系统稳定工作。另外,应注意凡是引到印刷板外的各电信号,均应加驱动电路,以保证信号不变形。

3.1.2 总线连接的动态特性

接口电路与系统总线相连接时,其总线时序与接口电路的工作时序配合是相当重要的。这是一个动态匹配问题。设计时首先要对系统总线的各信号波形加以分析、总结,列出各自作用的时间片段,并以此作为逻辑变量加以开发应用。

当接口电路连接的是慢速的外部设备时,系统总线的读写周期与外设的读写时间就会不相配合,即外设不可能在系统要求的时间内完成读写操作。如第 2 章所述,为同时兼顾快速设备,总线数据通信通常采用半同步通信方式。那么在接口电路设计时,需加入一个产生等待信号的电路。当地址译码选中该外设时,在 I/O 读或写命令控制下,该电路输出一个低电平有效信号,送往系统总线的 READY 控制端。CPU 在规定的时刻检测到 READY 信号为低,则在相应的 I/O 读写总线周期内插入若干等待(Tw)周期,通过延长总线周期来满足外设的读写要求。但必须注意等待的时间不易过长,否则将影响整个系统正常工作。

图 3-3 给出的是一个通用插等电路,通过跨接开关的连接可产生 1～5 个附加的等待周期。

从图中可看出,当译码选中某个端口时,其有效译码信号(高电平有效) 即输出到触发器的 D 端,然后由读写控制信号将触发器置位。经反向,I/O CH $\overline{\text{RDY}}$信号为低,该信号送往系统总线的 READY 输入端,以控制插入等待时钟周期。同时,触发器 Q 端的"1"信号传到 74LS174 的 D1 端。由系统时钟周期 CLK 的上升沿打入,Q1 端为"1"又传到 D2 端,再由下一个 CLK 的上升沿打入,使 Q2 端为"1" ……如果只需插入一个等待时钟周期,则可合上开关 1 ,这时 Q2 信号通过开关 1 传到"或非"门的输入端,控制触发器复位,以恢复 I/O CH $\overline{\text{RDY}}$信号为高电平,结束插等。同理可知,分别接通开关 2、3、4、5

可相应延长 I/O CH $\overline{\text{RDY}}$信号为低，从而产生 2～5 个插等周期。

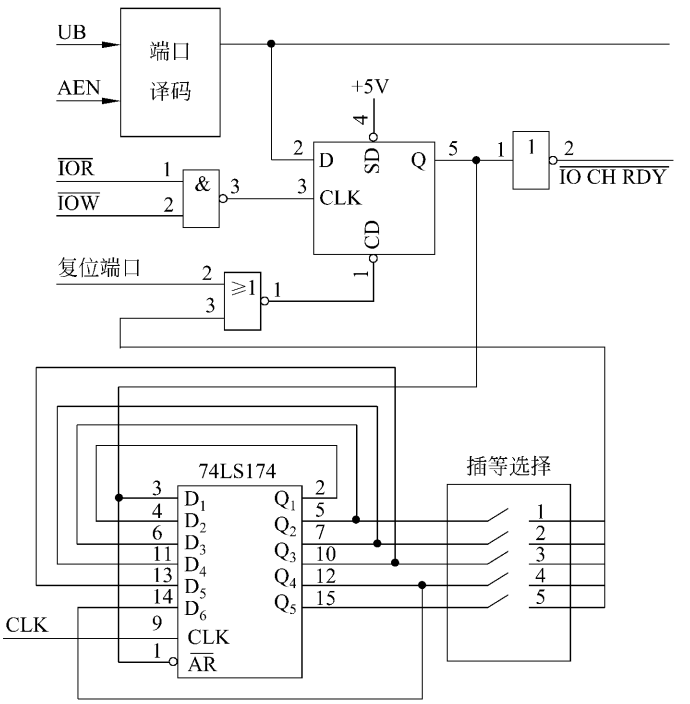

图 3-3　通用 I/O 插等电路

3.1.3　中断接口的设计

如前所述 CPU 与外设传送数据，通常可采用三种基本的方式，即程序传送方式、中断传送方式和 DMA 传送方式。设计接口时要根据所连外设的特点、数据传送的速率等因素多方面进行权衡。合理选择数据传送方式是保证系统可靠工作的关键。

程序传送方式接口电路较为简单，DMA 方式接口电路主要为 DMA 控制器，这两部分内容在第 1 章中都做了讨论，不再累述。本章主要讨论中断接口。

设计具有中断功能的接口电路，需考虑以下两个方面。

- 了解 CPU 的中断机构。
- 接口中断控制电路设计。

1. CPU 的中断机构

微处理器一般都具有中断功能，但不同的机型中断机构是不同的，设计时首先应对 CPU 的中断机构有所了解，才能完成正确的设计。

Pentium 系列机的中断分为三种类型：中断、异常和程序异常。中断通常也称为外部中断，由 CPU 外部硬件电路引发，包括不可屏蔽中断 NMI 和可屏蔽中断 INTR。NMI 不受“中断允许(IF＝1)”状态的影响，任何时候发生出错或故障都可请求 CPU 中断原程序转而为其进行处理。可屏蔽中断请求 INTR 由中断控制器驱动，完成对外部中断的控制。接口的中断请求信号可连于系统总线的相应中断请求输入端，向 CPU 发中断请求。

Pentium 系列机采用的是矢量中断方式,CPU 依据中断矢量号获取中断处理程序入口地址,但在实模式下和保护模式下采用不同的途径:实模式下使用中断向量表,而保护模式下使用中断描述符表。实模式下在内存储器的最低端设有 1KB 的中断矢量表,存放了 256 个中断矢量,表 3-2 列出了异常和中断矢量的分配情况。每个中断矢量占 4 个字节,前 2 个字节存放中断处理程序的偏移量 IP(低位在前;高位在后),后 2 个字节存放段地址 CS(低位在前;高位在后)。各中断矢量按中断类型号从 0 段 0 单元依次排列,见图 3-4。

表 3-2 中断矢量表

向量号	说 明	向量号	说 明
0	除法错	10	无效任务状态段
1	调试异常	11	段不存在
2	不可屏蔽中断	12	堆栈故障
3	断点	13	一般保护故障
4	被检测出的 INTO 上溢	14	页故障
5	超出了 BOUND 范围	15	保留
6	无效操作码	16	浮点错
7	设备不可用	17	对准检查
8	双故障	18～31	保留
9	协同处理器越段运行(保留)	32～255	可屏蔽中断

溢出中断、除法出错中断等属于内中断,是由微处理器内部产生的中断;DOS 系统调用和BIOS 中断调用是执行程序中断指令产生的程序中断;外部设备产生的中断则由可屏蔽中断输入端进入。

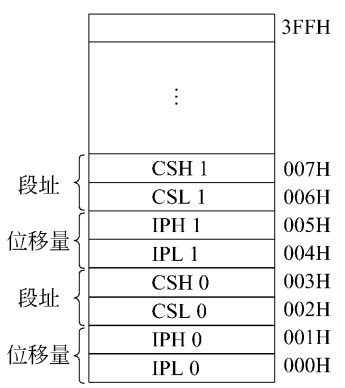

图 3-4 中断矢量表

发生中断时(中断源请求中断信号有效),若 CPU 允许中断(IF=1),在当前指令执行完后,CPU 响应中断。把从外设获得的中断类型号 4 倍乘后,依据此值从中断矢量表中找出中断处理程序的入口地址,然后把读得的值装入 IP 寄存器和 CS 寄存器中,从而完成程序的平滑切换。

2. 接口中断控制电路设计

若采用中断传送方式,则需在接口电路中配置相应的中断控制电路,包括中断请求触发器、中断屏蔽触发器等,当然如采用可编程接口芯片,那么其内部均已含有这些电路,可不必另外设计。一般来说,设计要做的工作可概括为以下三方面的内容。

(1) 中断源请求电路的设计

中断源的请求信号 INTR 是通过系统总线扩展槽的 IRQi 信号端送入的。有中断请求时,要求 INTR 信号由低变高,并保持高电平,直至处理机响应中断。包含中断屏蔽触发器的中断源请求电路如图 3-5 所示。

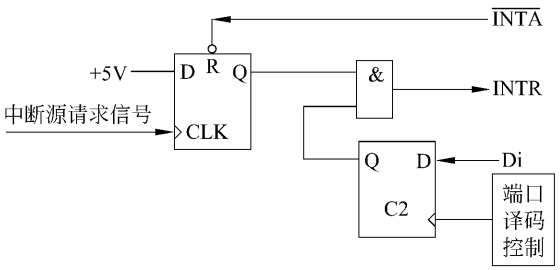

图 3-5 中断源请求电路

电路中用了两个触发器,C1 为中断请求触发器;C2 为中断屏蔽触发器。当中断源有请求时,该请求信号将中断请求触发器置"1",如此时中断屏蔽触发器为"1"(未屏蔽),则接口向 CPU 发中断请求(INTR 为高);反之,若中断屏蔽触发器为"0",则屏蔽接口的中断请求,封锁 INTR。CPU 响应中断请求后由响应信号将 C1 触发器清"0",复位中断请求信号,表示一次请求结束。中断屏蔽触发器的 D 端接系统数据总线的某一位,其内容可由 CPU 执行输出指令写入,通过编程控制接口是否屏蔽该中断源的中断请求信号。

(2) 中断处理程序的编制

中断处理程序可根据第 1 章所述流程按需要编制。

中断处理程序应具有良好的现场保护能力。因为中断往往是异步发生的,尤其是硬件中断的实时性很强。当它得到响应时,系统并不能自动地完全保护系统当前运行的各个状态。因此中断处理程序必须具备现场保护的能力,即应在中断处理程序中编写保护现场及恢复现场的程序段。

同时要注意堆栈存取的平衡性,中断响应后,其断点地址是保存在堆栈中的,而保护现场通常也是通过堆栈来实现的。在进行这一系列的堆栈操作时,要特别注意堆栈存取的平衡性。所谓堆栈存取的平衡性,是指 PUSH、POP 指令操作要相对应,否则在中断结束后,将无法返回正确的断点,严重的将会导致死机。

另外,中断处理程序的服务时间应尽可能短。也就是说,能在主程序做的工作尽量放在主程序做,以免影响其他同级或低级设备的中断。

(3) 中断处理程序的加载

对于像 Pentium 系列一类的微机,是利用中断矢量表方法来完成中断处理程序转移的。因此必须掌握怎么将用户中断处理程序的入口地址写入中断矢量表的相应位置,即中断处理程序的加载。修改矢量表要注意应采用 DOS 功能调用而不要采用 MOV 指令,否则会引起不必要的麻烦。具体加载部署如下:

① 保存原中断矢量内容。

可通过调用 DOS 系统功能 35H 来获取原中断矢量的内容,并置入堆栈或存储单元

保存。

```
MOV     AH,  35H            ; 读取中断矢量的功能调用
MOV     AL,  INT-TYPE       ; 中断类型号送 AL
INT     21H                 ; EX:BX=原中断矢量
MOV     KEEP-IP,  BX        ; 保存偏移地址
MOV     KEEP-CS,  ES        ; 保存段址
```

② 置入新的中断矢量。

置入新的中断矢量,使其指向自编的中断处理程序入口。该工作可通过调用 DOS 系统功能 25H 来完成。

```
PUSH   DS                   ; 保存 DS
MOV    DX, OFFEST INTI      ; 自编中断处理程序入口的偏移地址送 DX
MOV    AX, SEG INTI         ; 自编中断处理程序入口的段地址送 DS
MOV    DS, AX
MOV    AL, INT-TYPE         ; 中断类型号送 AL
MOV    AH,25H               ; 设置中断向量的功能调用
INT    21H                  ; 设置新的中断向量, DOS 调用
POP    DS                   ; 恢复 DS
```

当功能 25H 改变中断矢量表内容时,系统会自动禁止硬件中断,以防止在修改中断矢量表时产生硬件中断而引起错误。

③ 恢复原中断矢量内容。

应用程序执行完,在退出前,应从堆栈或存储单元中获取原中断矢量,并恢复到中断矢量表中。否则,后续程序将不能正确使用系统提供的例行程序。

```
PUSH   DS                   ; 保存 DS
MOV    DX, KEEP-IP          ; 取出保存的偏移地址
MOV    AX, KEEP-CS          ; 取出保存的段址
MOV    DS, AX
MOV    AL, INT-TYPE         ; 中断类型号送 AL
MOV    AH,25H               ; 设置中断向量的功能调用
INT    21H                  ; DOS 调用
POP    DS                   ; 恢复 DS
```

3.2 I/O 接口与外设连接界面的设计

计算机通过接口与外设相连,其接口与外设连接的界面同样不但要考虑各信号的静态连接,也要考虑动态的工作特性。由于外设种类繁多,其工作特性及工作时序有很大的差异。因此在设计接口电路时,首先要对被连接的外设的工作方式,特别是各输入输出引脚信号进行分析,并作出外设工作的控制时序图。然后,再根据应用需要,确定它们之间信息的交换方法,并合理地分配接口的软硬件功能,最后,整理归纳出接口电路。

3.2.1 并行通信方式与串行通信方式

有的外设采用并行通信的方式与主机交换信息，如打印机；有的则采用串行通信方式与主机交换信息，如鼠标。所谓并行通信就是把一个字符的各数位用几条数据线同时进行传输；而所谓串行通信就是把一个字符的各数位按通信规程约定的编码格式逐位地通过一根输入输出数据线进行传送。由于计算机的信息是以字节（8 位）或字（1 个～几个字节）为单位进行处理的。所以，以字节或字为单位传输比较方便，不必再进行任何转换。而采用串行通信，就必须使用移位寄存器进行并行→串行、串行→并行转换。因此在同样的传输速率下，并行通信的信息传输速度快，信息率高。当然，由于并行通信比串行通信所用的电缆要多，随着传输距离的增加，电缆的开销会成为突出的问题。所以，一般并行通信总是用在数据传输率要求较高，而传输距离较近的场合；串行通信则用于远距离传输。

采用并行通信时，需在主机与外设间配备相应的并行接口。并行接口可设计为只用来作为输出接口（如连打印机）；也可以设计为只用来作为输入接口（如连键盘），此外，还可以将它设计成既作为输入又作为输出的双向接口（如连磁盘驱动器）。

而采用串行通信方法时，则同样需要在主机与外设间配备相应的串行接口。串行接口可按其不同的串行通信方式进行设计。

实际上，无论是并行通信还是串行通信都是按时间顺序地逐个传输数据，传输过程中都要互相把自己的状态告诉对方，在相互确认对方操作的同时执行传送操作。因此，在接口电路中，除了应设计相应的输入缓冲寄存器和输出缓冲寄存器暂时存放数据；设计控制寄存器来接收 CPU 对它的控制命令外，还应设计状态寄存器，用来提供接口和设备的各种状态以便 CPU 查询，通过状态信息来同步双方的传输。

通常在进行系统扩展时，可根据外设特性选用并行或串行通信方式，并设计相应的接口。接口电路可采用中、小规模集成电路芯片或大规模可编程芯片进行设计。后面第 4 章和第 5 章将对并行接口和串行接口进行详细讨论。

3.2.2 模拟量信号的处理

在现实世界中存在许多连续变化的物理量，比如温度、速度、流量、压力等。这些量有一个共同的特点，即变化是连续的，具有这一特征的物理量称为模拟量。

一般来说，计算机特别是微型计算机只能处理离散的数字量。那么在利用计算机进行工业控制和参数测量时，就需要进行模拟量与数字量的转换。通常，当模拟量输入时，要接一个模拟量到数字量的转换接口，该接口电路负责将模拟量信号转换为数字量信号，送入计算机进行处理；当需要把计算机的处理结果转换为模拟量，对外部现实世界进行控制时，则需要一个数字量到模拟量的转换接口。

模拟量信号种类繁多，首先需通过各类传感器将各种类型的模拟量转换为模拟电流或模拟电压。常用的传感器包括光电元件、压敏元件、热敏元件等。然后由接口电路对模拟电流或模拟电压进行采样，得到与模拟量相对应的离散的脉冲序列，最后，用模/数转换器将离散量脉冲变为离散的数字信号，这样就完成了模拟量到数字量的转换。具有模/数

转换功能的接口称为 A/D 转换接口。关于 A/D 转换接口的设计本书第 7 章将进行讨论。

计算机处理的结果是数字量,若需对现场进行反馈控制,则还需通过数/模转换器把数字量转换为模拟电流或模拟电压,然后才能对现场进行控制。这一转换工作则是由 D/A 转换接口完成的。关于 D/A 转换接口的设计也将在第 7 章中进行讨论。

可见,D/A 转换是 A/D 转换的逆过程。这两个互逆的转换过程通常出现在一个控制系统中。如图 3-6 所示为一个实时控制系统的框图,从图中可以看到 A/D 转换器和 D/A 转换器各自在控制系统中的位置与作用。因为传感器一般不能提供足够的模拟信号幅度,所以在 A/D 转换器前面加了一级放大器;同样,D/A 转换器的输出信号通常也不足以驱动执行部件,所以在 D/A 转换器和执行部件之间也加了一级功率放大器。

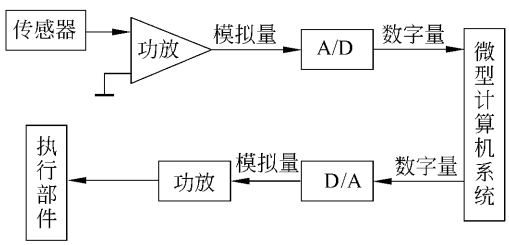

图 3-6　实时控制系统框图

通过 A/D 转换接口和 D/A 转换接口可构成各类计算机应用系统。如在图 3-6 中去掉 D/A 转换、功放及执行部件,仅保存 A/D 转换功能。这样就成了一个将现场模拟信号转换为数字信号、并送计算机进行处理的系统,这样的系统即为采样系统或测量系统。如保留 D/A 转换部分,而去掉 A/D 转换部分,那么,就构成了一个程序控制系统,可通过执行程序对外部进行控制。

3.2.3　利用标准接口扩展外设

一般通用计算机系统为方便用户,特别是不具备一定专业技术的用户进行系统扩充,通常在主机的设计中已设置一些常用接口电路,向外以标准接口的方式提供用户连接外设使用,如并行口(LPT1)、串行口(COM1、COM2)等。用户只需选择合适的外设,通过外设提供的连接线连于相应的接口插座上,并进行必要的软件安装,就可完成系统扩充了。这些标准接口通常位于主机后面板上,如图 3-7 所示。

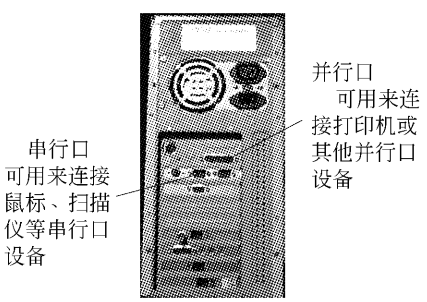

图 3-7　主机后面板

关于标准接口的应用将在第 9 章中介绍。

I/O 接口与外设连接界面的设计视所接的外设不同而不同,接口电路就是要将外设的数据信息转换为计算机能够进行处理的形式;将外设的工作信号转换为计算机能够匹配的信号,完成应有的转换是接口的主要功能。接口的具体设计方法在本书的以后各章中将详细讨论。