第1章　计算机控制外设的原理

从1981年美国IBM公司推出第一台PC机至今，个人计算机已深入我们生活的各个角落，给我们的生活带来了方便增添了乐趣，许多人为自己拥有一部电脑或会用电脑而自豪，更有许多人为学习好电脑而在发奋努力。为了配合广大读者学习电脑，各种各样的计算机书籍奉献在读者面前，使读者受益匪浅。人们在学习之余就想实践，想看到自己亲自制作的计算机产品，如果是纯软件方面的内容学习实践起来就相对容易一些，因为不需再花费很多财力(但也要费很大的脑力)，编程中即使错了也不会造成什么损失，但是硬件方面的学习就相对难一点了，它要求读者除了有软件方面的知识外还要有各种电子线路、计算机硬件结构等方面的知识，对广大的计算机爱好者来讲确实增加了难度。这是不是说初学者就不能开发计算机硬件了呢？应该说不是的，我们完全可以为初学者找到一条适合他们的硬件开发之路。

首先让我们从一个例子开始看一下开发计算机硬件的实质是什么。在工业应用上对步进电机的控制是常见的事，步进电机的控制需要三个参数：1.启动电机工作与停止电机工作信号；2.步进电机走向信号；3.电机移动距离与运动快慢的控制。其中，前两个信号分别是用电平的高低来实现，而第三个参数是通过高低电平的变化次数和频率来实现，高低电平变化的次数决定了电机的移动距离，而频率则决定了运动快慢。如果我们欲通过计算机对步进电机进行控制，就必须向步进电机输送三路信号，通过这三路信号分别实现上述三个参数的传送。而要实现这些工作就需要我们设置硬件电路作为连接计算机与步进电机的接口。从中我们不难看出计算机对外设的控制实际上是实现计算机与外部设备之间的通讯，这种通迅既需要开发硬件设备，在计算机与外设之间架起一座桥，又需要由软件制定通讯协议加以管理。

在着手开发硬件前，让我们看一下有哪些途径可走。总的来说有两条途径，首先想到的是自己动手开发标准的计算机接口卡，但是不是也可以考虑一下在现有计算机硬件设备的基础上，稍加改进而利用已有的标准接口卡做为第二条途径呢？下面我们针对这两种情况进行分析。

1.1　开发新的计算机接口卡

1.1.1　计算机扩展插槽

打开计算机的机箱，您会发现有一块横卧着的电路板，在该电路板上面有许多插槽，有的插槽中已插入了其它电路板。这些插槽是供计算机扩展其它设备用的，被称做扩展槽。不同的计算机插槽有所不同，但是，只要是IBM PC兼容机就都有几个具有62条插针的插槽。它是IBM PC机早期机型XT定义的标准插槽，以后的机器虽然有所发展和改进，但是为了与以前的机型保持兼容性，此插槽仍然给予保留。如图1-1所示为一块

PC/XT 计算机主板，其扩展槽已被标出。对于一些传送数据量不大或传送速度不高的外部设备而言这种插槽就足够用了。我们开发硬件接口卡时就要根据扩展插槽上不同信号的关系进行电路设计，做成的电路板尺寸也要符合扩展槽的标准，故我们称之为标准计算机接口卡。图 1-2 将 XT 扩展插槽中 62 条插针所代表的信号一一列出。

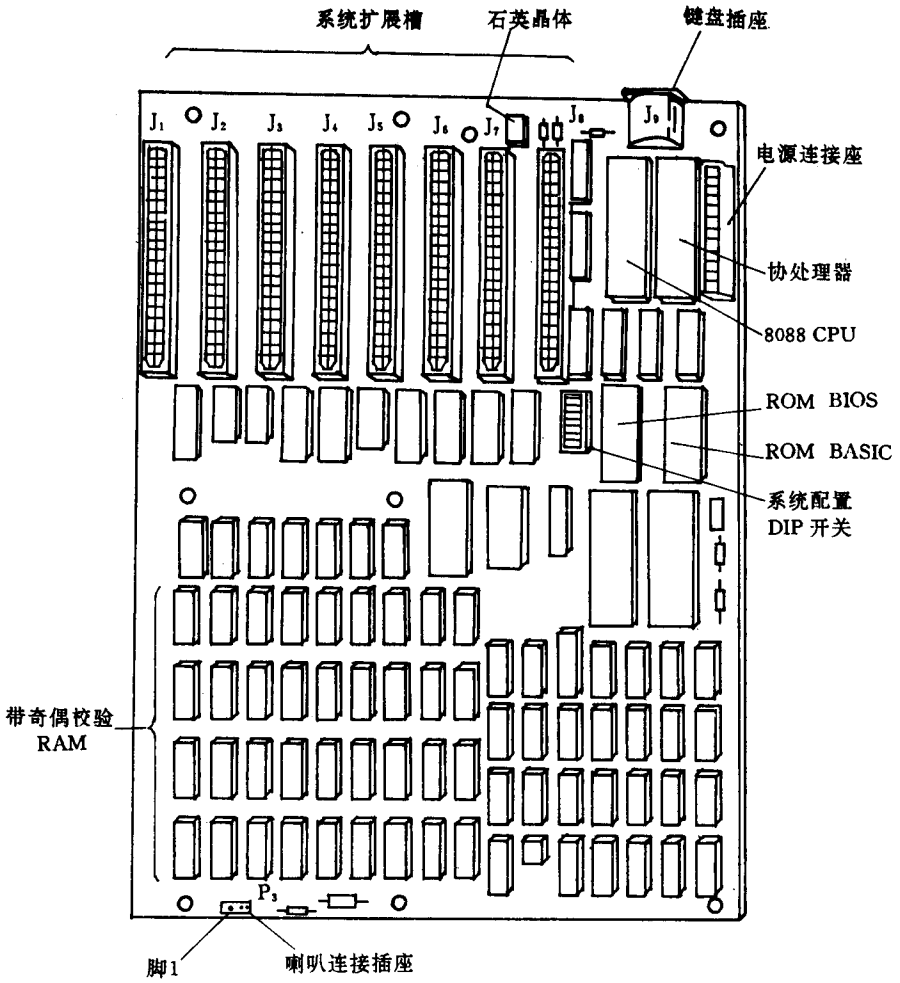


图 1-1　PC/XT 计算机主板

图 1-2 中的诸多信号线给人的第一印象是非常复杂，更不要提它们之间的相互关系了。在本书中我们不准备对所有信号的意义进行详细解释，这里只把这些信号按其功能进行大体分类，有关的信号线在各章涉及到时再做详细解释。PC/XT 扩展插槽上的 62 个信号按功能可划分为五类：

1）时钟信号：数字电路的基础是按时间顺序进行有序工作，因此时钟信号是不可缺少的。

2）地址总线：如果把计算机比作电话局，那么地址线好比电话号码分配器，它负责给每一个计算机外接设备分配一识别码，计算机根据这些地址线高低电平的不同组合找

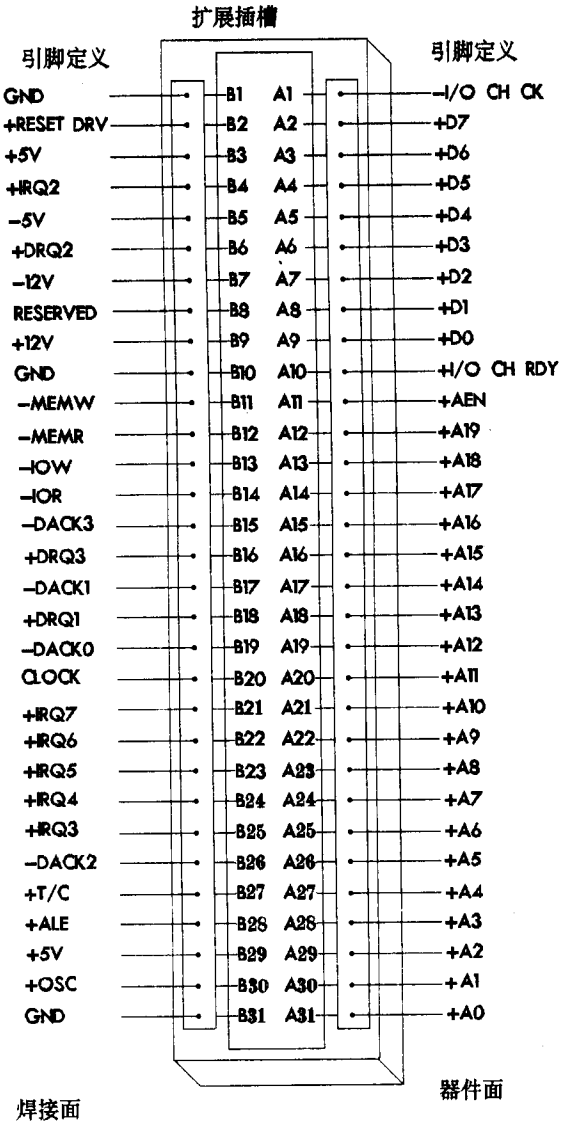


图 1-2 PC/XT 扩展插槽中各信号

到相应的设备。

3）数据总线：如果把地址总线比作电话号码分配器，那么数据总线就好比电话线，它负责传送有关的数据内容。但是与电话局不同的是，数据总线在某一时刻只能被一个计算机外接设备占用，如果其它设备也要使用数据总线，系统可根据它们的优先级来作判断，这就需要有一裁决线即下面提到的控制总线。

4）控制总线：控制总线主要负责数据总线中的数据流向，即计算机是向外部设备输送数据还是从外部设备接收数据，另外也负责判断哪一个外部设备主动申请与计算机通讯同时裁决外接设备的优先级别等。

5）电源与地：负责向扩展设备提供常用的几种电源。

当我们欲做一硬件接口电路板时，首先做的第一步是利用地址总线给电路板分配一识别码即地址译码，然后利用控制总线来控制数据总线的流向。如果还有其它要求则仍可利用控制总线进行相应的处理。对于地址总线的译码有两种处理方式，一种是把 I/O 设备(INPUT/OUTPUT，输入/输出设备)当作一存储单元而与计算机的存储体进行统一编址方式，另一种是把 I/O 设备进行单独编址方式。对于第二种方式，虽然 I/O 设备与计算机的存储体不统一编址，但是它们都使用相同的地址总线，所以为了不造成冲突必须用控制总线加以区别。当 I/O 设备与存储体统一编址时，计算机用控制总线中的 MEMR(MEMORY READ，读存储器)和 MEMW(MEMORY WRITE，写存储器)配合译码，单独编址用控制总线中的 IOR(INPUT/OUTPUT READ：读输入/ 输出设备)和 IOW(INPUT/OUTPUT WRITE，写输入/输出设备)配合译码。当然，在软件编程时也相应地使用不同的指令，前者利用 MOV 指令，而后者利用 IN 或者 OUT 指令。对于绝大多数接口电路板而言，一般都使用 I/O 设备单独编址方式。IBM PC/XT 计算机共给出了 1024 个 I/O 端口号(以后的新机型 I/O 可达 64K 个)，它们用 20 根地址线中的 10 根即可(2 的 10 次方为 1024)，用十六进制表示这些端口地址是 000H—3FFH。虽然系统给出了这么多 I/O 端口，但是有许多端口已被系统自身或常用的外围设备利用，例如显示器控制卡，打印机接口卡，串行通讯卡，软盘控制卡，硬盘控制卡等，用户必须有选择地利用余下的 I/O 端口，否则如果一个端口对应两个外部设备，此两设备皆不能正常工作，这好比两部电话使用相同的电话号码。图 1-3 给出了 1024 端口的分配情况(考虑到大多数机型已为 AT 兼容型机，故在此也列出了 AT 机的 I/O 端口分配情况)。

由于本书是围绕计算机打印口来进行讲述并开发产品的，故在此以打印机接口电路的设计为例介绍标准电路接口板的设计过程。

1.1.2 计算机打印卡电路设计

计算机打印口在许多用户看来它只能与打印机相连而不能挪作它用，实际上它是一并行通讯口，可以与任何符合该通讯标准的设备相连。并行通讯一词是相对于计算机的串行通讯而言的，在计算机的数字世界里，用来表示信息的最小单位是比特(BIT)，它或为 0 或为 1，比 BIT 大的单位是字节(BYTE)，它是常用单位，我们通常所讲的文件长度就是以字节为单位，每一字节由 8 个比特组成，由于串行通讯是按时间顺序一比特一比特地传送，一个字节需拆成 8 次传送，所以当按时间顺序一字节一字节传送时就称为并行通讯。

现在设想一下一台设备与另一台设备通讯需要什么条件：首先要有传送的具体内容，其次是要告知对方做好准备以便接收数据，最后应能了解到对方是否已接收了数据。从上面分析看来对于一个通讯口至少应有三个通道：

1）数据通道：它共有 8 根数据线组成(即一个字节)，计算机输出的打印内容就是通过它传送给打印机的。

2）控制打印机通道：该通道将计算机对打印机的控制信号传送给打印机，这些信号应包括对打印机初始化信号，告知打印机接收数据信号等。

端口地址	设备名称		
	XT机	AT机 (ISA或EISA)	说明
0-F	DMA控制器 (8237A-5)	DMA控制器1 (8237A-5)	
10-1F		DMA控制器2 (8237A-5)	
20-2F	中断控制器 (8259A)	中断控制器1 (8259A)	实际只用20,21
30-3F		中断控制器1 (8259A)	
40-4F	定时器 (8253-5)	定时器 (8254-2)	XT机用40-43
50-5F		定时器 (8254-2)	
60-6F	并行口 (8255A-5)	键盘 (8042)	XT机用60-63
70-7F		RTC, NMI	
80-9F	DMA页寄存器	DMA页寄存器 (74LS612)	XT机只用80-83
A0-AF	NMI屏蔽寄存器	中断控制器2 (8259A)	
B0-BF		中断控制器2 (8259A)	
C0-DF		DMA控制器2 (8237A-5)	
E0-EF			
F0-FF		协处理器 (80287)	只用F0,F1,F8-FF
100-1EF			
1F0-1FF		硬盘	
200-20F	游戏口	游戏口	
210-21F	扩展单元		XT用210-217
220-26F			
270-27F	打印机2	打印机2	只用278-27F
280-2AF			
2B0-2DF	EGA	EGA	
2E0-2EF	GABO, 数据采集0	GPIB, 数据采集0	只用2E1,2E2-2E3
2F0-2FF	串口2	串口2	只用2F8-2FF
300-31F	典型卡	典型卡	
320-32F	硬盘适配器		
330-33F			
340-35F	DCA 3278		XT用348-357
360-36F	PC 网		XT用360-367
370-37F	打印机1	打印机1	用378-37F
380-38F	SDLC	SDLC	XT用390-393
390-39F	群适配器	群适配器	
3A0-3AF	同步通讯	同步通讯	
3B0-3BF	单显及打印机适配器	单显及打印机适配器	
3C0-3CF	EGA	EGA	
3D0-3DF	CGA	CGA	
3E0-3EF			
3F0-3FF	磁盘控制器, 串口1	磁盘控制器, 串口1	

图 1-3 I/O 端口分配表

3）检测打印机状态通道：该通道将打印机的当前的状态情况传送给计算机，这些信号包括打印机是否可以接收数据，打印纸是否已经装好，打印机是否有错误等。

计算机的打印口正是由上面的三个数据通道组成。图 1-4 是早期的计算机打印机接口电路图，它是由最基本的数字电路搭接而成的。现在由于集成电路工业的发展，打印机接口大多与计算机软盘驱动控制器、串行通讯口等集成在一起构成了一块电路板，这就是所谓的多功能卡，但是，尽管如此其电气特性仍保持不变。

从图 1-3 的计算机 I/O 端口地址分配表上看，打印机接口卡被分配的端口地址是 370～37F 以及 270～27F 共 32 个端口。之所以被分配两段地址，是因为每台计算机可以同时接两台打印机，而每台打印机就被分配给一段端口地址。大多数情况下计算机只带有一个打印口，即我们俗称的 LPT1 口(一些 COMPAQ 计算机用 LPT2 口)，它被分配给的端口地址是 370～37F。虽然每个打印口被分配给多达 16 个端口地址，但从前面的分析中可以看出，打印口用三个端口地址足矣：数据传送口，控制输出口，状态读入口，根本用不到所有的 16 个地址端口。实际情况的确是这样：PC 机只选用 378H、379H 和 37AH 作为 LPT1 的三个端口地址，其它地址端口可以留做它用。

还是先看一下这三个端口地址是怎样得到的。由于端口地址 378H、379H、37AH 是三个连续地址，所以在对 10 根地址线译码时它们的高 8 位是相同的，可以统一译码为 11011110B，即地址线中的 A9、A8、A6、A5、A4、A3 为高电平有效，A7、A2 为低电平有效。图 1-4 中的器件 U5、U9、U11 实现了这一译码功能，但是图中地址线 A2 并没有参加译码而另一信号线 AEN 却参加了译码，这是因为图 1-3 中端口地址 37CH、37DH、37EH 也被分配给了打印机接口 LPT1，但它并未被用到，因而可以不予考虑，在这种情况下端口地址 378H 与 37CH 是重合的，同样 379H 与 37DH，37AH 与 37EH 的地址也是重合的。AEN 信号表示地址有效，它的参与就保证了地址译码的有效性。另外，地址线 A8 经过了 U11 的预处理，这主要是为了方便用户的选择而设的，U11 是一异或门，它的输入端 4 与一跳线开关 J1 相连。当跳线开关 J1 断开时输入端 4 为高电平，此时只有 A8 地址线为低电平时 U11 的输出端才为高电平，这样打印机接口卡的端口地址被设为 278H、279H、27AH 也就是所说的 LPT2 口；当跳线开关 J1 合上时，U11 的输入端 4 与地相连，其电平为低，作为 U11 另一输入脚的 A8 为高电平时 U11 的输出端才为高电平，这种情况正好是 LPT1 的端口地址设置。接下去是对低两位地址线进行处理。先看一下三个端口的作用，然后根据它们的作用再作讨论。IBM 公司在设计打印机接口时对三个端口地址的作用是这样分配的：378H 端口用作传送打印数据，在通常情况下计算机只向该端口写数据，但是为了在计算机开机时能检查向该端口所写数据的正确性，计算机对该端口也必须有读取数据的功能，该端口是读写双向的；地址端口 379H 被用作检测打印机状态，计算机只需从此端口读取打印机的状态即可，该端口是单向的；地址端口 37AH 被用来控制打印机，计算机控制打印机时向端口写数据，而当计算机初始化时为检测控制端口的正确性也应能从该端口读数据，37AH 端口也是读写双向的。当计算机向某端口写数据时用到扩展插槽中的 IOW 信号线，它是低电平有效，当从某端口读取数据时要用到 IOR 信号线，它也是低电平有效。了解到了各端口的作用可以用 IOW 和 IOR 两信号线配合低两位地址线译码，图 1-4 中的 U6 和 U9 就起到了这样的作用。U6 是晶体管

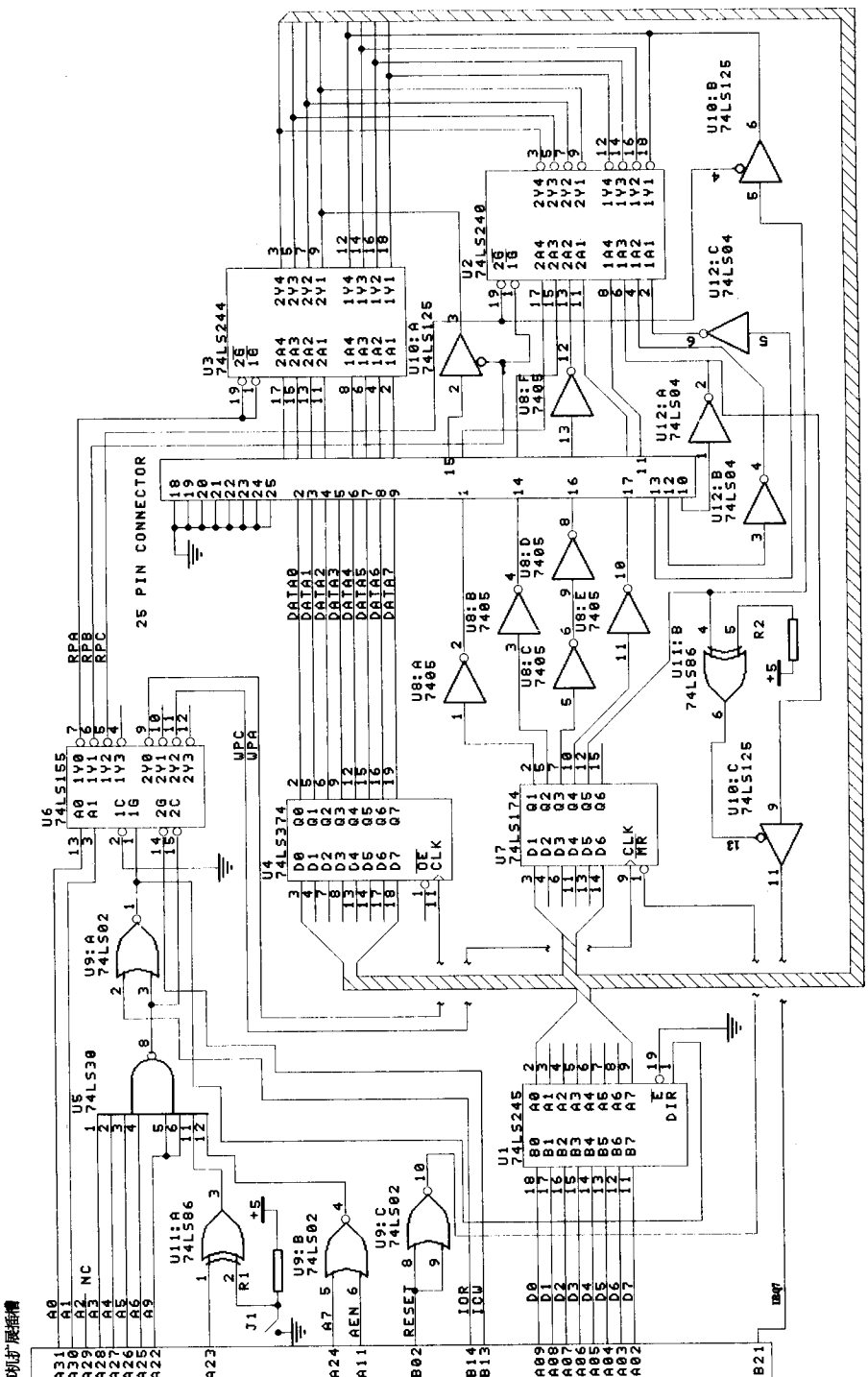
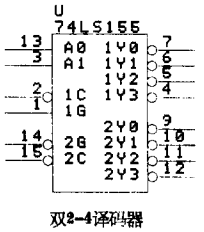


图 1-4　打印机接口电路图

集成电路 74LS155，该芯片包含两个 2—4 线译码器，图 1-5 列出了各管脚之间的逻辑关系。



双2-4译码器

LOGIC TABLE

INPUTS				OUTPUTS			
A0	A1	1C	1G	1Y0	1Y1	1Y2	1Y3
X	X	H	X	H	H	H	H
L	L	L	H	L	H	H	H
H	L	L	H	H	L	H	H
L	H	L	H	H	H	L	H
H	H	L	H	H	H	H	L
X	X	X	L	H	H	H	H

"X" MEANS NO CARE

INPUTS				OUTPUTS			
A0	A1	2C	2G	2Y0	2Y1	2Y2	2Y3
X	X	X	H	H	H	H	H
L	L	L	L	L	H	H	H
H	L	L	L	H	L	H	H
L	H	L	L	H	H	L	H
H	H	L	L	H	H	H	L
X	X	H	X	H	H	H	H

图 1-5　74LS155 逻辑关系表

对照图 1-4 中 U6 管脚的接线关系以及图 1-5 中的逻辑关系表，我们可以看出：当地址线 A0、A1 与信号线 IOW 以及 U5 的输出端为低电平时 U6 的第 9 脚输出为低电平(图 1-4 中标为 WPA 即 Write Port A 的缩写)，此时是计算机向端口 378H 写数据；当地址线 A0 与写信号线 IOW 以及 U5 的输出端为低电平且地址线 A1 为高电平时 U6 的第 11 脚输出为低电平(图 1-4 中标为 WPC 即 Write Port C 的缩写)，此时计算机通过端口 37AH 向打印机发送控制命令；当地址线 A0、A1 及读信号线 IOR 以及 U5 的输出端为低电平时 U6 的第 7 脚输出为低电平(图 1-4 中标为 RPA 即 Read Port A 的缩写)，此时计算机通过端口 378H 读取数据；当地址线 A1 及读信号线 IOR 以及 U5 的输出端为低电平且地址线 A0 为高电平时 U6 的第 6 脚输出为低电平(图 1-4 中标为 RPB 即 Read Port B 的缩写)，此时计算机通过端口 379H 读取打印机状态数据；当地址线 A0 及读信号线 IOR 以及 U5 的输出端为低电平且地址线 A1 为高电平时 U6 的第 5 脚输出为低电平(图 1-4 中标为 RPC 即 Read Port C 的缩写)，此时计算机通过端口 37AH 读取数据。

对每个端口的地址译码及其读写控制已处理好，剩下的问题就是对各端口的数据线进行处理。在介绍计算机扩展插槽时，我们曾指出 PC/XT 的数据总线共 8 条，而每一个端口都要用到它，因此为了避免数据冲突必须对各端口的数据线加以隔离，只有当计算机欲通过该端口读数据时，该端口上的数据线才能与扩展插槽中的数据总线连通。图 1-4 中的 U1 是一个 8 位双向数据缓冲器，它的作用是把打印机接口卡中的数据线与扩展插槽中的数据总线隔离开并增加驱动能力，尽管该数据缓冲器始终被选中，但是由于通常情况下数据流向是从扩展插槽到该端口，因而不会使打印机接口卡与与其它外设发生冲

突。一般情况下，U1 的数据流向是由 B 到 A，打印机接口卡中的数据总线不会干扰扩展槽中的数据；只有当计算机欲从打印机接口卡中读取数据时 U9 的输出端为低电平，U9 输出低电平就会导致 U1 的数据流向由 A 向 B，使计算机从打印机接口卡上获取正确数据。U4 是一个 8 位 D 型触发器，它受控于 WPA 即端口 378H，当计算机通过端口 378H 向打印机传送数据时，数据总线上的 8 位数据被 WPA 锁存在 U4 中保持不变，只有当计算机再通过端口 378H 再次向打印机传送新数据时，U4 中锁存的数据才被更新。U4 的输出端有两路去向，一路去往 25 芯 D 型连接器，该连接器将数据送往打印机，另一路去往 U3 的输入端，U3 受控于 RPA，当计算机通过端口 378H 读取数据时 RPA 为低，这样计算机读取的数据实际上是 U3 中保存的数据，计算机将该数据与先前发出的数据进行比较就可以判断端口 378H 是否正常。U7 是一个 5 位 D 型触发器，它受控于 WPC，计算机向打印机发出的控制命令就被锁存在该芯片中，由于向打印机发 5 种控制信号(为什么用 5 种控制信号，第 2 章中会解释)即可，所以 U7 的数据输入端只用了数据总线中的低 5 位。U7 的清零端与扩展槽中的信号线 RESET 相连，这主要是为了在计算机初始化时也能将打印机控制口初始化。U7 输出的低 4 位分别通过 U8 去往 25 芯 D 型连接器和 U2。U8 的作用有两个，第一个作用是增加驱动能力以便通过 D 型连接器向打印机传送正确控制信号，第二个作用是适应打印机对不同电平信号的要求。U2 是一个 8 位反相数据缓冲器，它被分成两部分，一部分受控于 RPC，也就是将 U7 锁存的信息读回到计算机中，通过将该数据与前面送出的数据比较就可判断控制数据的正确性；U2 的另一部分受控于 RPB，当 RPB 为低电平时，这一部分电路将打印机的状态信息输送到扩展槽中的数据总线上被计算机接收，需注意的是打印机状态信息共 5 位，它占用数据总线的高 5 位。图 1-4 中还有一个标为 U10 的芯片，它也是一个数据缓冲器，共包含 4 个独立控制的三态缓冲器，图中的两个用于补充 U2 的不足(U2 只有 8 个三态缓冲器，而控制打印机信号线与打印机状态线共 10 条，故需外加两个三态缓冲器)。U10 的另一个输入端为 12 脚、输出端为 11 脚的缓冲器另有用途，通过它打印机状态信号线 ACK 传给扩展插槽中标为 IRQ7 的引脚，这是为打印机向计算机申请中断服务而设的，以后的章节里我们会着重讨论该中断服务，读者在此不必深究。

到此为止我们以打印机接口为例介绍了计算机标准接口卡的设计过程，对于专业技术人员来讲可能不难，但是对于成千上万的初学者来讲，各种信号线的意义、诸多硬件术语就显得过于复杂了，怎样化复杂为简单尽快掌握计算机硬件功能就成为他们的共同要求。下面的一节内容就是讨论这方面问题。

1.2 利用计算机已有的标准接口卡

对于开发过 8051 系列单片机的人来说，一般有两种感受：第一种是设计直观简单，8051 系列单片机从总体上说它共有 4 个端口，每个端口上又有多少不等的输入输出信号线，设计者既可以对每个端口进行总体操作，也可以对每个端口中的每一位进行单独操作。例如把端口 P0 的第 1 位做为信号输入线，而把端口 P0 的第 2 位做为信号输出线，不必像 PC 机那样要先用地址线译码决定端口地址再用 IOR 或 IOW 信号线决定端口是

数据输入读还是数据输出写，最后还要对数据线进行处理。正因为如此简单明了，设计开发单片机的工程技术人员相对较多；另一种感受是困难的一面，由于单片机主要用于工业控制，因而基于单片机开发的软件一般不通用，大量软件资源不能被利用，造成重复开发，再就是单片机开发环境不尽如人意，虽然有许多基于 PC 机研制的各种开发系统，但它们只不过是在用 PC 机仿真，仍不如 PC 机的开发环境那样友好方便，更何况世界上有无数的 PC 机工作人员在不断地开发新的 PC 机编程软件。能不能在开发 PC 机时将单片机的优点结合过来呢？答案是肯定的，PC 机的功能远远超出单片机的功能，在它本身实现单片机的优点完全可能，本章的第一节介绍的打印机接口就是很好的例证。

对于打印机接口，从它与外部联系的角度上看它有三个端口，这三个端口既可单独使用又可配合使用，而且能读能写，这与单片机的四个端口很相似。但是，从内部看它与单片机有着本质区别：它可以利用计算机的大量存储单元，有各种各样从低级到高级的软件开发环境，还有键盘显示器等硬件开发环境，最主要的是开发所见所得不像单片机那样要进行仿真。另外，对端口的操作所用指令也不多，用户只要用计算机的 IN 或者 OUT 指令就可以直接控制每个端口的每一位信号线。例如，在本章开头提出的步进电机控制问题，我们不必像第一节讲的那样重新做一块标准的 PC 机接口板，只需利用打印机接口中的数据输出端口 378H 或者打印机控制口 379H 再配合一条 OUT 命令就能实现对步进电机的控制，因为对步进电机的控制只需三个具有 TTL(Transitor-Transitor Logical)电平的信号线即可，上述两个端口输出的是 TTL 电平信号，刚好符合步进电机的要求，根本用不着考虑诸多地址线、数据线以及控制线的相互配合。

不难看出，这样做的优点如下：

1) 设计简单。用户不必关心计算机扩展槽中繁杂的信号意义及其它们之间的相互关系，只需考虑打印机三个端口之间的相互配合即可。

2) 危险性小。用户不必打开机箱，更不必将电路板在扩展插槽中插来拔去，这样就减少了对计算机内部其它部件损坏的机会，即使用户将打印口损坏了也不会祸及计算机主板和其它设备，花几十元钱将损坏的打印机接口卡换掉计算机就能正常工作。

3) 节省空间。由于不占用计算机扩展槽，所以可将有限的扩展插槽留给其它接口板用。

4) 开发方便。PC 机的开发软件成百上千，稍微利用一下就能编写出功能丰富的软件应用系统。

5) 安装方便。用户设计的电路板只要用标准的 25 芯电缆线与打印机接口相连即可，不必考虑其它尺寸。

第 2 章　打印机接口卡的基本编程

第 1 章中我们着重介绍了打印机接口卡的硬件设计原理，本章将就其软件编程问题进行讨论。

2.1　计算机基本输入输出口的编程

在第 1 章中我们曾提到对打印机接口卡的控制用 IN 或者 OUT 两条计算机指令。其具体形式是，先将 I/O 端口的地址送往计算机的通用寄存器 DX 中，然后再利用 IN 或者 OUT 指令将该端口上的数据送往计算机通用寄存器 AL 或者将寄存器 AL 中的数据送往 I/O 端口，两种命令的格式如下：

从 I/O 端口读取数据(PORT 为端口地址)：

```
MOV   DX,  PORT
IN    AL,  DX
```

向 I/O 端口输出数据(PORT 为端口地址)：

```
MOV   DX,  PORT
OUT   AL,  DX
```

让我们在计算机的调试程序 DEBUG 命令下直接用 IN 或 OUT 指令来检测一下打印机接口卡各端口读写的正确性。

检测端口 378H：先向该端口写一数据，然后从该端口读回，将输出的数据与读回的数据进行比较，看其正确性。在 DOS 命令下运行 DEBUG 后，用 A(ASSEMBLE) 汇编命令输入如下指令。

```
-a
mov    dx,   378
mov    al,   88
out     dx,  al
in     al,   dx
```

(注意：DEBUG 命令下所有的数值均为 16 进制)

指令输入完后，再用 T(TRACE)单步跟踪命令执行，结果如下：

```
-t
AX=0000  BX=0000  CX=0000  DX=0378  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0103   NV UP EI PL NZ NA PO NC
3654:0103 B088            MOV    AL,88
-t
AX=0088  BX=0000  CX=0000  DX=0378  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0105   NV UP EI PL NZ NA PO NC
```

```
3654:0105 EE              OUT     DX,AL
-t
AX=0088  BX=0000  CX=0000  DX=0378  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0106   NV UP EI PL NZ NA PO NC
3654:0106 EC              IN      AL,DX
-t
AX=0088  BX=0000  CX=0000  DX=0378  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0107   NV UP EI PL NZ NA PO NC
3654:0107 47              INC     DI
```

当执行完第一步后，通用寄存器 DX 中的值变为 378H，端口地址 378 被送往 DX 中；执行第二步后寄存器 AX 中的值变为 0088H，相应寄存器 AL 的值为 88H；执行第三步后，由于是向 378H 端口写数据，所以除了指令指针 IP(Instruction Pointer) 有变化外，其它寄存器没有变化；接下去执行第四步，此时寄存器 AX 中的值仍为 0088H，说明读回来的值与刚才输出的值是相等的，也就是说端口 378H 可能是正确的，之所以说可能是正确的，是因为我们并没有检测每一位信号的正确性，只有对每一位输出 0 或 1 检测都正确后才能说该端口是正确的。

检测端口 379H，与 378H 端口不同的是 379H 端口只能读取数据，所以如果先向该端口写一数据，然后从该端口读回就不一定能看出其正确性，但是为了与端口 378H 形成对比，不妨先运行 DEBUG 后，用 A(ASSEMBLE)汇编命令输入如下指令。

```
-A
mov   dx,379
mov   al,88
out   dx,al
in    al,dx
-t
AX=0000  BX=0000  CX=0000  DX=0379  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0103   NV UP EI PL NZ NA PO NC
3654:0103 B088            MOV     AL,88
-t
AX=0088  BX=0000  CX=0000  DX=0379  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0105   NV UP EI PL NZ NA PO NC
3654:0105 EE              OUT     DX,AL
-t
AX=0088  BX=0000  CX=0000  DX=0379  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0106   NV UP EI PL NZ NA PO NC
3654:0106 EC              IN      AL,DX
-t
AX=005F  BX=0000  CX=0000  DX=0379  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0107   NV UP EI PL NZ NA PO NC
3654:0107 47              INC     DI
```

前两步分别将端口地址和欲输出的数据送往寄存器 DX 和 AL 中，第三步将 AL 中的数据送往端口 379H，关键看第四步，执行后 AL 中的值变为 5FH 而不是 88H 了。在这种情况下无论输出什么数据，读回的数据是不变的，这正说明该端口只能输入不能

输出。

测试端口 37AH，同端口 378H 一样该端口既能输出数据又能输入数据，但是又有所不同，因为端口 37AH 只用到了数据总线中的低 5 位，高 3 位数据没有用到处于悬空状态，所以在测试时无论送出什么数据，读回的数据必然大于或等于二进制数 11100000B 也就是十六进制 E0H，下面的程序可验证这一点。仍然在 DEBUG 命令下敲入下列指令：

```
-a
mov  dx,  37a
mov  al,  88
out  dx,  al
in   al,  dx
```

单步执行结果如下：

```
-t
AX=0000  BX=0000  CX=0000  DX=037A  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0103  NV UP EI PL NZ NA PO NC
3654:0103 B088          MOV   AL,88
-t
AX=0088  BX=0000  CX=0000  DX=037A  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0105  NV UP EI PL NZ NA PO NC
3654:0105 EE            OUT   DX,AL
-t
AX=0088  BX=0000  CX=0000  DX=037A  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0106  NV UP EI PL NZ NA PO NC
3654:0106 EC            IN    AL,DX
-t
AX=00E8  BX=0000  CX=0000  DX=037A  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3654  ES=3654  SS=3654  CS=3654  IP=0107  NV UP EI PL NZ NA PO NC
3654:0107 47            INC   DI
```

从上面的执行结果来看，送出的数据与读回的数据确实不同，而且读回的数据的确大于 E0H。然而，端口 37AH 究竟是否正确呢？我们只需看其低 5 位数据即可，读回的数据为 E8H 化为二进制为 11101000B，而输出数据 88H 化为二进制为 10001000B，可以看出它们的低 5 位是完全相同的，也就是说端口 37AH 也可能是正确的。

通过对打印机接口卡三个端口的检测，我们了解了软件对于计算机 I/O 端口的基本控制过程。但是，它们并没有像 8051 单片机那样对某一端口的某一位数据进行单独控制的指令，然而我们可以通过软件的手段来实现这一功能。假设我们欲改变某一端口某一数据线的状态而对其它数据线的状态保持不变，我们可以先将该端口的数据读回来，在保证其它位不变的情况下只改变相应的位的逻辑状态，最后再将改动后的数据向该端口输出，就可实现对该端口的某一数据线的单独控制。对于某一位数据的改变无非是将其置 0 或置 1，如果是置 0，只需将一个该数据位为 0 而其它数据位为 1 的数据与读回的数据相“与”，如果是置 1，就要将一个相应数据位为 1 而其它数据位为 0 的数据和读回的数据相“或”。仍以端口 378H 为例，在测试端口 378H 时我们曾输出了 88H，假设我们现在要使 378H 端口上的第 0 位数据线为高电平状态，可以采用下列程序：

```
MOV   DX,  378
IN    AL,  DX
OR    AL,  01     ;00000001B
OUT   DX,  AL
```

执行上述指令后端口 378H 中数据线的最低位就变为高电平了。同样，下列程序会将端口 378H 的第 1 位数据线置为低电平。

```
MOV   DX,  378
IN    AL,  DX
AND   AL,  0FD    ;11111101B
OUT   DX,  AL
```

到目前为止，对于计算机打印口上的三个端口我们完全可以像 8051 单片机的端口那样使用，只要用 IN 或者 OUT 命令就可以同外部设备通讯。

2.2 对打印机的控制编程

在这一节里我们将着重介绍打印机接口卡的原始用法：与打印机的连接。只有对打印口接口最原始的应用有所了解，才能在此基础上开发出新的产品。

2.2.1 打印机接口卡各端口的定义

在介绍打印口的硬件电路时，我们曾提到它们的定义，这里再详细介绍一下。

1）端口 378H 的定义：

本端口被定义为数据传输口，因为它有 8 位数据线，刚好组成一个字节，计算机将打印内容通过它送往打印机。

2）端口 379H 的定义：

该端口被定义为打印机状态检测口，它共有 5 位数据，各位数据代表的意义如图 2-1 所示。

数据位：	D7	D6	D5	D4	D3	D2	D1	D0
意　义：	busy	ack	pe	slct	error	X	X	X

*注："X"表示无意义，可不予考虑。

图 2-1　打印机状态字定义

各信号的具体含义如下：

BUSY ——打印机忙信号，当打印机忙时它不接收来自计算机的数据，此时计算机应等待，该信号是高电平有效，即 BUSY 为高电平时，打印机忙。

ACK —— 打印机应答信号，它是为打印机向计算机申请中断服务而设的，该信号应是低电平脉冲信号。

PE ——打印机无打印纸信号，当无纸时打印机一般会响警铃，它是高电平有效。

SLCT —— 打印机联机信号，它是高电平有效，一般情况下打印机不会处于脱机状态，所以在编程时可不考虑这个信号。

ERROR—— 打印机出错信号，这种错误警告并不是将打印机的所有错误传给计算机，只有几个严重的错误，像脱机、无纸等才会引起警告。该信号为低电平有效。

3）端口 37AH 的定义：

端口 37AH 被定义为打印机控制信号，它只有 5 位信号，各位的定义如图 2-2 所示。

数据位：	D7	D6	D5	D4	D3	D2	D1	D0
意　义：	X	X	X	IRQE	SLCT	INIT	AUTOFD	STROBE

*注："X"表示无意义，可不予考虑。

图 2-2　打印机控制字各位定义

各信号的具体含义如下：

IRQE ——打印机中断允许信号。当该信号为高电平时，如果打印机向计算机发出中断服务请求信号，计算机就可去执行中断服务程序。当该信号为低电平时，不允许打印机向计算机发出中断服务请求。

SLCT ——与打印机处于联机状态，一般情况下可不予考虑，事实上我们并不希望脱机。它是高电平有效。

INIT ——打印机初始化信号。该信号应是一个低电平脉冲信号，脉冲宽度应根据打印机的要求而定。IBM 公司在其早期的打印机中要求为 50μs。当用户开始使用打印机时为了确保打印机状态的正确给予初始化，一般情况下不需对打印机初始化，所以该信号应保持为高电平。

AUTOFD——自动换行信号，当该信号为高电平时，打印机只要收到回车（打印头回到最左边）数据就自动向前移动一行距离。大多数情况下计算机会把回车符与换行符一同送往打印机，所以该信号常被置为低电平。

STROBE ——数据选通信号，该信号为一低电平脉冲信号，它的作用是告知打印机将端口 378H 中的数据接收过去。

2.2.2　打印机接口卡与打印机的硬件连接

打印机接口卡与打印机的连接是通过一个 25 芯屏蔽电缆实现的，其中接口卡端为 25 芯 D 型插头，而打印机端是标准 CENTRONICS 插座，具体的连接关系如图 2-3 所示。

在图 2-3 的连接关系中，我们看到 25 芯信号线中的 18－25 共 8 根信号线接地，这是为减少外界干扰，减少信号之间的耦合而设计的，现在市场上有许多打印电缆只用其中的一根线接地，这是不符合设计标准的。

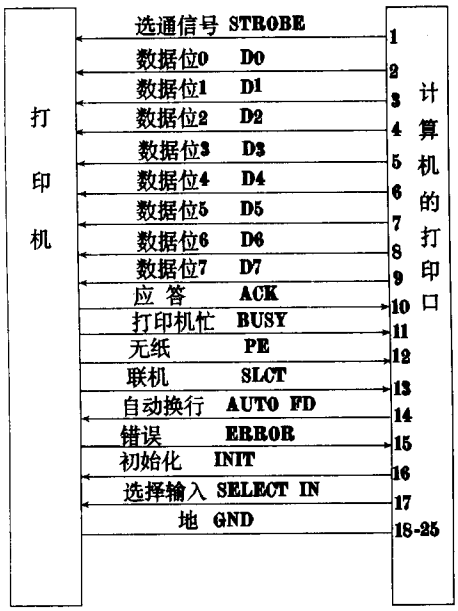


图 2-3　计算机与打印机连接关系

2.2.3　计算机与打印机的软件接口

所谓软件接口就是在硬件基础上两台设备之间按某种协议进行的信息交换。由于是计算机控制打印机，所以必须先知道打印机对各信号相互关系的要求后才能在计算机上进行编程，图 2-4 是打印机要求各信号所遵守的时序图。

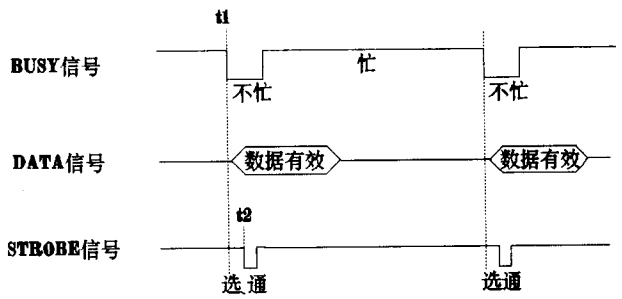


图 2-4　打印机接口时序图

从图 2-4 中可以看出，在 t1 时刻打印机处于不忙状态，计算机检测到这一信息后将当前要打印的数据送往数据线上，数据被送出大约 0.5μs 后在 t2 时刻计算机发出一低电平选通脉冲，打印机检测到该脉冲后就立即将打印机忙信号线置为高电平，也就是忙状态，同时对数据线上的数据进行处理，计算机发出选通脉冲后即等待着打印机“不忙”的信息以便传输下一个数据，当打印机将打印数据处理完后，再次将打印机忙信号线置

为低电平，以便告诉计算机传送下一数据。图 2-5 是向打印机传送打印数据的程序流程图。

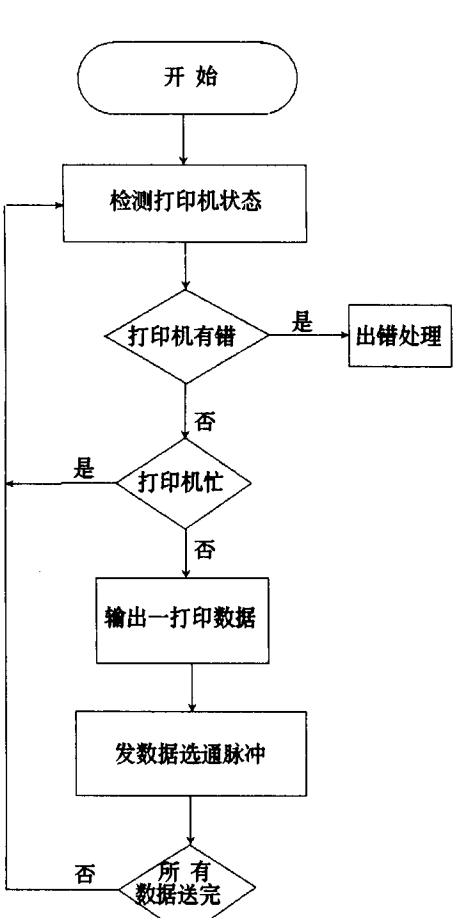


图 2-5 打印机输出流程图

根据上面制定的通讯协议，现编制有关的打印程序。姑且不考虑打印机是否出错，假设打印机一切已准备好，下面的一段程序就可使打印机在打印纸上打印出一字符“0”，并且向前走一张纸。我们仍用调试程序 DEBUG 来执行这一任务。在 DEBUG 命令下敲入下面的程序，并用单步执行命令 t 运行：

```
-a
3654:0100  BA7903      MOV     DX,0379
3654:0103  EC          IN      AL,DX       ;读取打印机状态
3654:0104  A880        TEST    AL,80       ;测试打印机是否忙
3654:0106  74FB        JZ      0103        ;如果忙，则继续读取打印机状态
```