

第一篇

I/O 控制的剖析

- 第一章 微处理器的 I/O
- 第二章 I/O 适配器
- 第三章 可编程外设接口 8255A
- 第四章 8255A 的基本应用示例
- 第五章 系统中断
- 第六章 IBM PC 及 8259A 的基本应用示例
- 第七章 可编程计时器 8253/8254
- 第八章 8253/8254 的基本应用示例

第一章 微处理器的 I/O

本章学习目的

1. 读者可通过本章获得 IBM PC 的接口知识。
2. 读者通过 I/O 的读写时序图可以了解微处理器的 I/O 工作方式。

本章内容

- 1.1 基本的 I/O 接口
- 1.2 IBM PC 的 I/O 结构
- 1.3 IBM PC 的扩充槽(slot)结构
- 1.4 思考题

何谓接口? 见图 1.1。微处理器的各种接口。

微处理器是计算机系统的核心,它必须通过键盘、驱动器、屏幕、打印机等与外界组成一个完整的系统。外围设备与微处理器之间的数据传输必须依赖接口的协助,微处理器就是通过这种接口准确高速地控制着每一个外设。

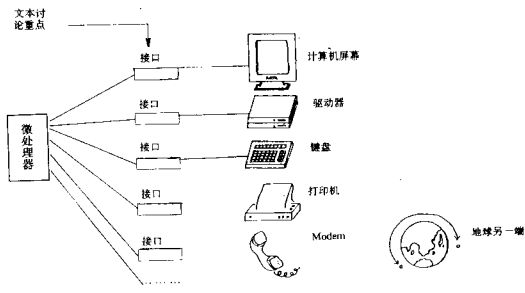


图 1.1 微处理器的接口

微处理器的 I/O 可分为以下类型:

- 输入/输出端口映射式 I/O (Input/Output Port Mapped I/O)

这种 I/O 的最大特色是它自己拥有独立的 I/O 指令，其存储器寻址和 I/O 寻址是独立的，只能以 IN, OUT, INS 与 OUTS 等指令访问，这种 I/O 为本书讨论的重点。

注：对 Turbo C 而言，IN, OUT, INS 与 OUTS 等指令相当于下列函数 *inportb()*, *outportb()*, *inport()* 和 *outport()*。有关这些函数的使用方式，可参考附录 B。本文叙述仍将以 IN, OUT 代表输入及输出，不过读者以 Turbo C 设计程序时，可将它想象成 *inport()* 和 *outport()*。

- 存储器映射式 I/O (Memory Mapped I/O)

这种存储器映射对象位于主存储器的地址空间内。例如屏幕上每一点都可称为是 I/O，因为这些 I/O 包含于主存内，所以称为存储器映射式 I/O。CPU 借助这些 I/O 将信息显示于屏幕上。

1.1 基本的 I/O 接口

典型的 IBM PC 系统有下列常用的外设元件：

- 8259A 可编程中断控制器。
- 8237 DMA 控制器，用以直接、快速的处理存储器的访问。
- 8272 软盘驱动器控制器 (Floppy disk controller)
- 82062, 82064 硬盘驱动器控制器 (Hard disk controller)
- 8274 复式串行控制器 (Multi-protocol serial controller)

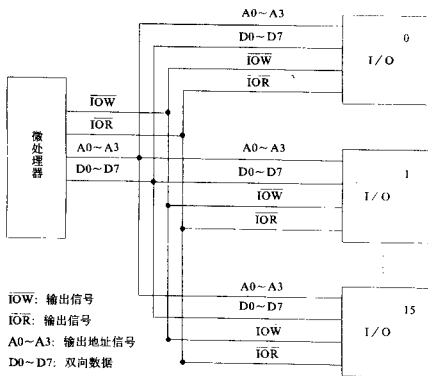


图 1.2 微处理器与 I/O 口结构方块图

- 82258 高等 DMA 控制器。
- 8255A 可编程外设接口 (Programmable peripheral interface)
- 82C53(82C54)可编程计时/计数器 (Programmable interval time)

本书的讨论重点在于 8259A, 8255A, 82C53(82C54)的编程, 以实例分析、探讨这些接口的 I/O 及其应用。

1.2 IBM PC 的 I/O 结构

IBM PC 系统中有许多 I/O 系统专用的 IC, 如系统计时的 82C54, 中断处理的 8259A 及直接存储访问的 8237 等。这些 I/O 都有特定的地址, 微处理器通过这些地址与不同的 I/O 外设通讯。读者在设计 I/O 实验时, 必须查阅所用系统的使用手册, 避免不同的 I/O 使用相同的地址, 造成系统混乱。

在图 1.2 中共有 16 个 I/O 口, 由 A0-A3 确定不同的 16 个 I/O 地址。D0-D7 为系统送出的双向数据总线 (bus)。IOW 为 I/O Write 的数据控制总线, $\overline{\text{IOR}}$ 为 I/O Read 的数据控制总线。

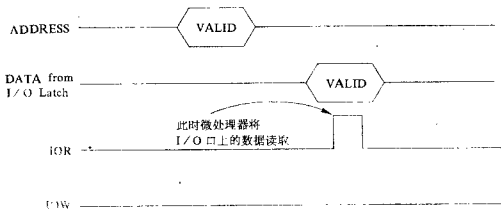


图 1.3 I/O 读取时序图

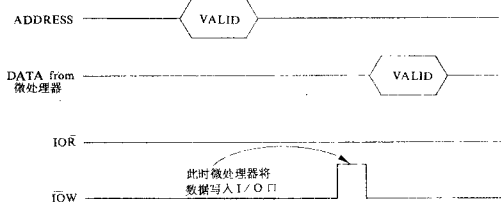
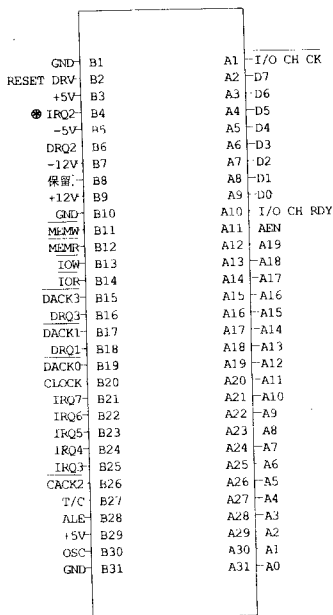


图 1.4 I/O 写入时序图

图 1.3 为系统读取 I/O 外设的时序图。当微处理器发出 $IN \times \times$ ($\times \times$ 为地址) 时, 所有的 I/O 均将自己的地址与 $\times \times$ 比较。如果确定自己被系统选到时, 即将数据从 I/O 放出, 然后被系统送出的 $\overline{IOR}$ 将 I/O 的数据送至系统数据总线上。

图 1.4 为系统将 DATA 写入 I/O 外设的时序图。当微处理器发出 $OUT \times \times$ ($\times \times$ 为地址) 时, 所有的 I/O 即将自己的地址与 $\times \times$ 比较。如果确定自己被选到时, 即可利用 $\overline{IOW}$ 将系统数据总线上的 DATA 写入 I/O 的数据锁存器 (DATA Latch) 中。



* 在 80386 中此信号为 IRQ9

图 1.5 扩充槽引脚 (62 脚)

1.3 IBM PC 的扩充槽 (Slot) 结构

1.3 IBM PC 的扩充槽 (Slot) 结构

IBM PC 提供了若干扩充槽供外设接口使用。譬如下列外设使用了这些扩充槽。

- 打印机
- EGA、VGA 显示卡
- 磁盘驱动器

扩充槽上有许多信号引脚, 经过这些特定的信号排列, 外设接口可以准确地和微处理器联系。在每个扩充槽中配有 62 个引脚, 各代表不同的信号, 每个槽上的外设可以配置任何标准设备, 如打印机、EGA、VGA 显示卡及驱动器。本书的讨论重点即在这个扩充槽上, 我们将设计一个硬件的接口, 插入这个扩充槽, 和微处理器沟通信息。

信号名称及扩充槽引脚请参阅图 1.5。现将各信号的代表意义说明如下:

O

表示由微处理器输出。

I

表示由外设输入至微处理器。

A0~A19(O)

地址位 0~19。这 20 个地址位用以指定存储器及接口位置。

D0~D7(I/O)

数据位 0~7。

OSC(O)

振荡器的频率, 由振荡器输出到此, 再送到外设上。

CLK(O) (CLOCK)

PC 系统振荡频率输出。

RESET DRV(O)

此复位信号用来复位(RESET)外设, 这一逻辑信号电平通常为低电平。

ALE(O) (Address Latch Enable)

地址锁定允许, 在一连串的系统总线上, 载有各种不同的信息, 当 ALE 产生时, 会将此时的总线状态锁定, 被锁定的信息即被视为地址区。

MEMW (O) (Memory Write)

存储器写入信号, 低电平动作的三态(Three State)信号 (高、低、高阻抗), 将要写数据写入存储器。

MEMR (O) (Memory Read)

存储器读取信号, 低电平动作的三态(Three State)信号 (高、低、高阻抗), 用来读取存储器的数据。

DRQ1~DRQ3 (I) (DMA Request)

当外界需要直接存储器访问 (DMA) 服务时, 可通过这些信号告诉 DMA。

注: DRQ1 具有较高的优先权。

- 在 DMA 应用中, 其实 DRQ0 具有较高的优先权, 但此信号已被指定用以刷新

(Refresh)动态存储器,所以不能被用户使用。

DAK0~DAK3 (I) (DMA Acknowledge)响应信号

当系统知道接口通过 DRQ1~DRQ3 提出服务要求之后,会产生此信号通知接口“我知道了”。

I/O CH CK (I) (I/O Channel Check)

I/O 通道 (I/O Channel) 的奇偶校验同步位,低电位时有效 (Active Low),此时表示 I/O 通道同步错误。

I/O CH RDY (I) (I/O Channel Ready)

此信号表示 I/O 通道准备完成的意思。当外界接口配置未完成准备工作时,会将此信号设为低电平,以告诉微处理器暂缓运行,以适合低速外部设备与高速微处理器之间的信息输送。

IRQ2~IRQ7 (I) (Interrupt Request)

第二级至第七级的中断请求。第零级和第一级已分别被 82C53 系统计时器及键盘扫描中断使用,因此不能被其它外设使用。

$\overline{\text{IOR}}(\text{O})$ (I/O Read)

I/O 读取信号,为低电位时工作 (Active Low)。

$\overline{\text{IOW}}(\text{O})$ (I/O Write)

I/O 写入信号,为低电位时工作。

AEN (O) (Address Enable)

地址允许,将被锁定的地址输回系统总线。

T/C(O) (Terminal Count)

结束计数,表示 DMA 已经完成所要求的信息传输。

注意所有符号上端有横线 (例如英文字 $\overline{\text{IOR}}$ 上方的横线) 表示低电位有效,其余为高电位有效。

表 1.1 介绍 I/O 扩充卡信号驱动的能力。请读者注意,以免发生计算机系统不能驱动外设,或因而产生死机的问题。

表 1.1 I/O 扩充卡推动的能力

总线信号名称	IOL	TOH
D7~D0	23.7	-15.0
A19~A16	7.2	-2.5
A15~A14	21.3	-2.5
A13	23.2	-2.6
A12~A0	23.5	-2.6
IOR, IOW, MEMR, MEMW	24.0	-5.0
CLK	23.2	-15.0
AEN	24.0	-15.0

(续表)

总线信号名称	IOL	TOH
DACK0	24.0	-15.0
DACK1	3.2	-0.2
DACK2, DACK3	2.8	-0.2
ALE	14.6	-0.9
RESET DRV	8.0	-0.4
T/C	8.2	-0.4
OSC	5.2	-1.0

注 1: 上表以毫安(mA)为单位

注 2: 不同主机有不同的驱动能力

表 1.2 介绍 I/O 扩充槽输入信号的负载。

表 1.2 I/O 扩充槽输入信号负载的能力

I/O 总线信号名称	LOAD	
	IIL	IIH
D7~D0	-0.42	0.05
I/O CH CK	-0.42	0.03
I/O CK RDY	-0.44	0.03
IRQ7~IRQ2	-0.01	0.01
DRQ3~DRQ1	-0.01	0.01

注: 上表以毫安(mA)为单位

1.4 思考题

1. 何谓接口 I/O? 并将所有您知道的 I/O 接口列出来。
2. 拆开您的计算机, 分辨出所有的 IC 及扩充槽, 将所有的 IC 列出来, 并说明它们的功能。
3. 比较 IBM PC XT, AT, 386, 486, 说明它们有何不同。
4. 比较计算机外设各个驱动器的不同点。它们之间可否互换, 如将 A 驱动器和硬盘 C 互换, 它们的适配器是否相同?

第二章 I/O 适配器

本章学习目的

1. 读者通过本章可以获得如何设计适配器的相关知识。
2. 读者经过本章亦可温习电子线路的基本概念。
3. IC 测试、适配器的测试基本知识也可以借助本章获得。

本章内容

- 2.1 微计算机 I/O 扩充槽
- 2.2 如何使用扩充槽
- 2.3 I/O 适配器
- 2.4 适配器的测试
- 2.5 思考题

I/O 适配器是 PC 系统和外界交换信息的外设接口。任何外设控制都必须借助这个适配器。如果读者打开计算机，将会发现在计算机背后插着多块外设卡，这些卡有些连接计算机屏幕，有些连接驱动器、键盘或打印机。这些卡又称为适配器。

2.1 微计算机 I/O 扩充槽

在第一章中曾经详述 PC 系统的 I/O 扩充槽，这些扩充槽有的是 62 脚，有的是 36 脚，用以提供不同的 PC 系统接口使用。62 脚的扩充槽不论在 XT 中还是 AT 中大致相同，唯一的不同点是在 80386 的 62 脚扩充槽的第 4 脚信号为 IRQ9 与 XT 的 IRQ2 不同。如果读者详细研究 80386 系统之后会发现，80386 的 PC BIOS 已将此部分处理过了，所以对用户而言，可以将 IRQ9 当成 IRQ2 来处理。在本章、第六章及研究专题中有些示例是在 80386 系统下编程的，读者可以参考并加以应用。

2.2 如何使用扩充槽

市面上的适配器，形式如图 2.1。

在图 2.1 中，我们可以发现 I/O 标准适配器上的 62 个信号和 PC 系统扩充槽上的 62 个信号相连接。这 62 个信号的定义完全相同，不同的扩充槽结构和信号也完全相同，所以任何一个同脚数的扩充槽均可以接受任何驱动器或打印机，甚至于将来开发出来的新卡。

在本书中，所讨论的重点为 8255A、82C53 / 82C54 及 8259A。针对以上的要求我们将这 3 种集成电路设计在这个 I/O 标准适配器上，通过所讨论过的 62 个标准信号和 PC 系统互传信息。

这些 PC 上的标准扩充槽已经在全世界通用，任何新的适配器必须与之配套，否则不能和任何的 IBM PC 机兼容。

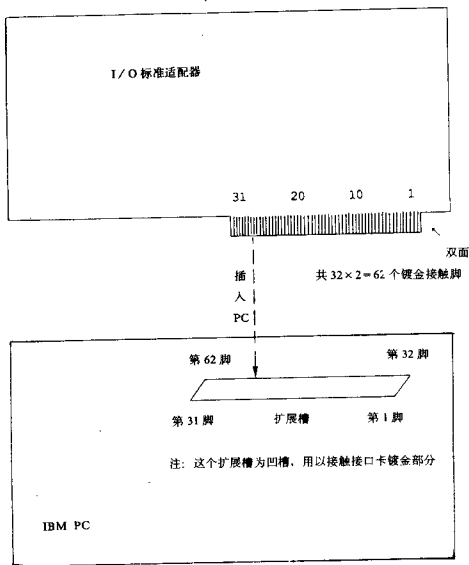
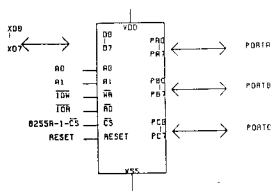


图 2.1 微计算机系统和适配器的连接结构

2.3 I/O 适配器

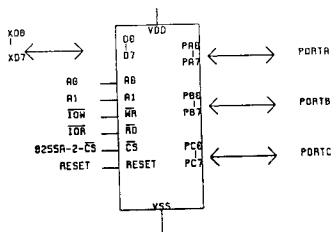
本书所谈论的重点在于 8255A, 82C53/82C54 及 8259A 三种集成电路。我们设计了一个适配器，利用这三种集成电路和外界传递信息。这三种集成电路为 I/O 接口所不可缺少的重要组件，随后的各章节中将详述它们的功能。详细研究之后，可了解 PC 系统所有接口的基本结构。以下简介这个适配器的结构。



ADDRESS	REGISTER
3E0H	PORTA
3E1H	PORTB
3E2H	PORTC
3E5H	CTR REG

8255A-1 I/O ADDRESS

图 2.3A 8255A I/O 地址及引脚图



ADDRESS	REGISTER
3E4H	PORTA
3E5H	PORTB
3E6H	PORTC
3E7H	CTR REG

8255A-2 I/O ADDRESS

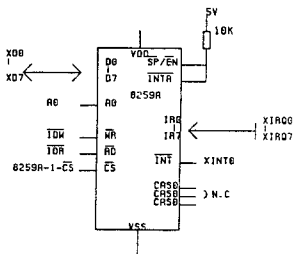
图 2.3B 8255A I/O 地址及引脚图

2.3.3 8259A I/O 地址及引脚图

这个 I/O 适配器提供了一个 8259A, 且只在主(Master)模式下使用, 所以 $\overline{SP}/\overline{EN}$ 都为高电平。

2.3.4 8253 / 8254 I/O 地址及引脚图

这个 I/O 适配器上提供了一个 8253/8254, 其中 CLK0 由系统的时钟频率除以 4 而得到, CLK1 及 CLK2 接上系统的 CLK。



ADDRESS	REGISTER
3ECH	ICW1
3EDH	ICW2
3EDH	ICW3
3EDH	ICW4
3EDH	OCW1
3ECH	OCW2
3ECH	OCW3

8259A-1 I/O ADDRESS

图 2.4 8259A I/O 地址及引脚图

一般来说,集成电路的测试可分为三个部分:

- DC 参数(直流部分)测试
- AC 参数(交流部分)测试
- 功能测试

DC参数:一些与时间无关的参数

如输入电压的要求 V_{il}, V_{ih}

输出电压的规格 V_{ol}, V_{oh}

输入漏电流 I_{il}

输出漏电流 I_{ol}

电源供应电流 I_{cc}

AC参数:一些与时间有关的参数

如输入波上升时间 T_r

输入波下降时间 T_f

一些模拟信号的增益测量,或其它的复杂信号测量等也属于 AC 参数。

功能测试:测IC的功能,这一部分是属于最重要的部分,也是最容易了解的部分。

一般集成电路在芯片做完之时即经过一次测试,称为 (Pretest)。好的芯片 (Chip) 在经过封装之后又经过一次测试,称为 (Final Test)。Final Test 的最主要的目的是测试 IC 在封装时是否有损坏。一个好的测试程序必需能将真正有问题的集成电路挑出,但是很少有任何测试程序能做完 100% 的测试。以一个微处理器的集成电路为例,它含有数万个晶体管,如何在 10 秒之内将这数万个晶体管完全测完实属不易,而且常常有些晶体管会有动态的问题(譬如说,它会受另一个晶体管的影响),这样更增加了集成电路测试的难度。几乎没有一家集成电路公司曾对外宣布他们的微处理器经过 100% 的测试。

2.4.2 适配器的测试

适配器上有一些集成电路,这些集成电路基本上并不在适配器测试程序上详细测试。适配器的测试主要是测集成电路是否完好地焊在适配器上。相比之下,这种测试就简单多了。这类的测试,我们自己就可以书写程序测试。笔者写了一个程序用以简单地测试本书所使用的适配器,原理如下:

- 8255A-1: 写入值“55H”于PORT 3E0中,再读回。如果成功读回,即表示这片 IC 是好的。参考思考题 1。
- 8255A-2: 写入值“55H”于PORT 3E3中,再读回。如果成功读回,即表示这片 IC 是好的。不然就是坏的,或是适配器上并没有这一片 IC 存在。
- 8253 / 8254: 写入值“55H”于第0个计数器中,在“free run”下读回其值两次,如果不相同,表示这片 IC 的第 0 个计数器有动作,所以是好的。如果相同,则再读一次,如果与前两次不同,则表示这片 IC 的第 0 个计数器是好的。三次相同的概率为 $1/256^3 = 5.9 \times 10^{-8}$, 程序误判的概率相当小,所以不予考虑。参考思考题 2。
- 8259A: 依次写入ICWS
ICW1: 13H

ICW2: 0

ICW3: 0

ICW4: 55H

再读回ICW4，如果为55H表示此部分是好的。

示例程序 iotest.c

I/O 适配器的测试，下图是示范测试的结果。

I/O Card Testing

8255A-1 O.K.

8255A-2 O.K.

8254 O.K.

8259 O.K.

```
/* ----- */
/*      Program Name : iotest.c      */
/*      Testing the I/O card.      */
/* ----- */

#include <dos.h>
#include <conio.h>
#define PORT1 0x3e3
#define PORT2 0x3e7
#define PORT3 0x3eb
unsigned char val[] = { 0x13, 0x00, 0x00, 0x01, 0x55 };
int addr[] = { 0x3ec, 0x3ed, 0x3ed, 0x3ed, 0x3ed };

void main()
{
    unsigned char bytread, bytewrite, tmp;
    int PORT;
    int i;
    unsigned char testing = 0x55;

    clrscr();          /* clear the screen */
    gotoxy(34,2);
    printf("I/O Card Testing"); /* print the program title */

    /* Testing the 8255A-1 */
    gotoxy(30,5);
```



```

    printf("8255A-1");
/* set PORT 3e0 for output */
    bytewrite = 0x80;
    outportb(PORT1,bytewrite);

    bytewrite = testing;
    PORT = 0x3e0;
    outportb(PORT,bytewrite); /* send testing data to PORT 0x3e0 */

/* set PORT 3e0 for input */
    bytewrite = 0x90;
    outportb(PORT1,bytewrite);

    bytread = inportb(PORT); /* get the data from PORT 0x3e0 */
    gotoxy(41,5);
    if ( bytread == testing )
        printf("O.K.");
    else
        printf("%dFail or not implemented.",bytread);

/* Testing the 8255A-2 */
    gotoxy(30,7);
    printf("8255A-2");
/* set PORT 3e4 for output */
    bytewrite = 0x80;
    outportb(PORT2,bytewrite);

    bytewrite = testing;
    PORT = 0x3e4;
    outportb(PORT,bytewrite); /* send testing data to PORT 0x3e4 */

/* set PORT 3e4 for input */
    bytewrite = 0x90;
    outportb(PORT2,bytewrite);

    bytread = inportb(PORT); /* get the data from PORT 0x3e4 */
    gotoxy(41,7);
    if ( bytread == testing )
        printf("O.K.");
    else
        printf("Fail or not implemented.");

/* Testing 8254 */
    gotoxy(30,9);

```