8.4.2 PCI总线信号定义

在一个PCI应用系统中，如果某设备取得了总线控制权，就称其为"主设备"；而被主设备选中以进行通信的设备称为"从设备"或"目标节点"。对于相应的接口信号线，通常分为必备的和可选的两大类，为了进行数据处理、寻址、接口控制、仲裁等系统功能， PCI接口要求作为目标的设备至少需要47条引脚，若作为主设备则需要49条引脚。下面对主设备与目标设备综合考虑，并按功能分组将这些信号表示在图8．19中。其中，必要的引脚在左边，任选的引脚在右边。

一．信号类型说明

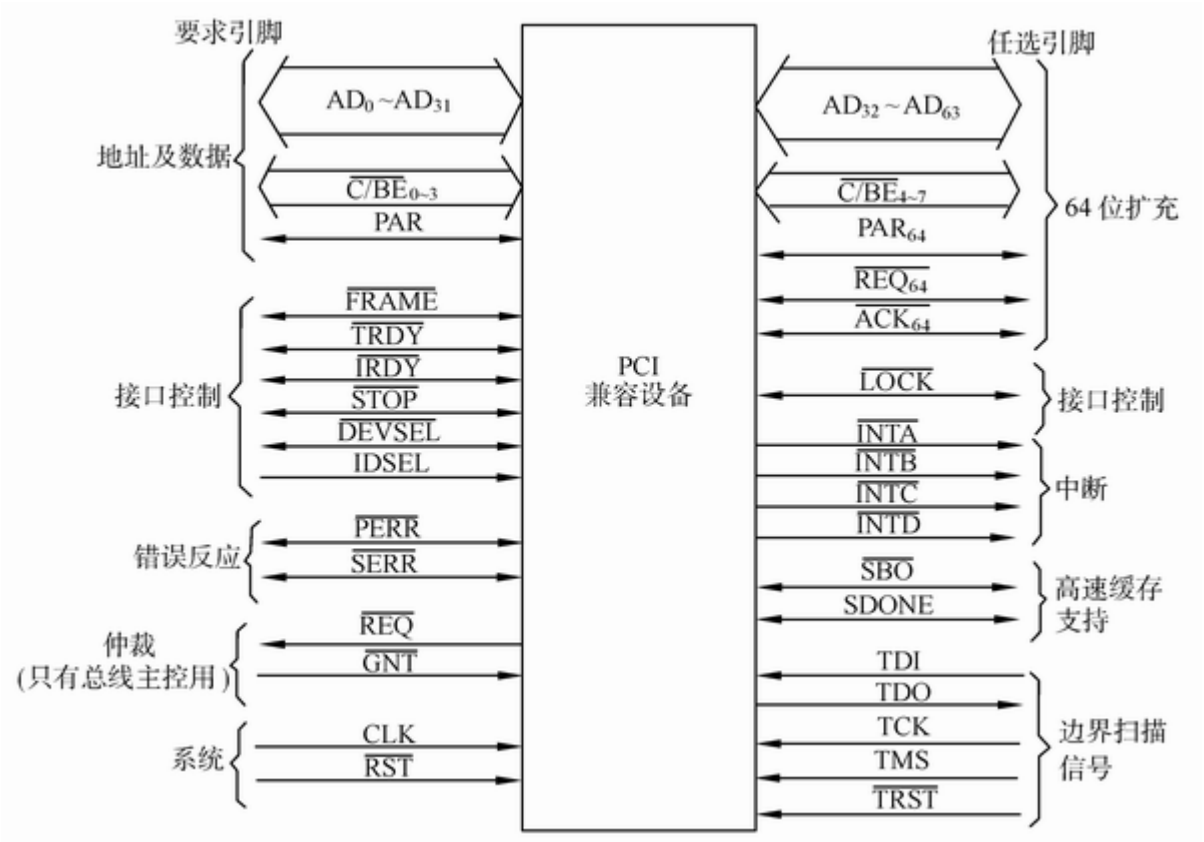

图8．19 PCI引脚示图

为了叙述方便，将PCI信号按数传方向及驱动特性划分为五种类型，各种类型的规定 如下：

in：输入信号。

out：输出驱动信号。

t/s：表示双向三态输入 / 输出驱动信号。

s / t / s：持续三态（Sustained Tri-State），表示持续的并且低电平有效的三态信号。在某一时刻只能属于一个主设备并被其驱动。这种信号从有效变为浮空（高阻状态）之前必须保证使其具有至少一个时钟周期的高电平状态。另一主设备要想驱动它，至少要等到该信号的原有驱动者将其释放（变为三态）一个时钟周期之后才能开始。同时，如果此信号处于持续的非驱动状态时，在有新的主设备驱动它之前应采取上拉措施，并且该措施必须由中央资源提供。

o/d：漏极开路（Open Drain）可作线或形势允许多个设备共同使用，

二． PCI总线信号定义

PCI总线的信号线共有100根，下面按功能分组进行说明。

1．系统引线

CLK in：时钟输入，为所有PCI上的接口传送提供时序。其最高频率可达66MHz，最低频率一般为0（DC），这一频率也称为PCI的工作频率。对于PCI的其他信号，除$\overline{RST}$、$\overline{IRQB}$、$\overline{IRQC}$、$\overline{IRQD}$之外，其余信号都在CLK的上升沿有效（或采样）。

$\overline{RST}$ in：复位，用来使PCI专用的特性寄存器和定时器相关的信号恢复规定的初始状况。每当复位时， PCI的全部输出信号一般都应驱动到第三态。

2．地址和数据引线

AD0～AD31 t/s：地址、数据多路复用的输入 / 输出信号。在$\overline{FMAME}$有效时，是地址周期；在$\overline{IRDY}$和$\overline{TRDY}$同时有效时，是数据周期。一个PCI总线的传输中包含了一个地址信号周期和一个（或多个）数据周期。PCI总线支持突发方式的读写功能。

地址周期为一个时钟周期，在该周期中AD0～AD31线上含有一个32位的物理地址。对于I / O操作，它是一个字节地址；若是存储器操作和配置操作，则是双字地址。

在数据周期， AD0～AD7为最低字节， AD24～AD31为最高字节。当$\overline{IRDY}$有效时，表示写数据稳定有效，$\overline{TRDY}$有效表示读数据稳定有效。

$\overline{C/BE0}$～3 t/s：总线命令和字节使能多路复用信号线。在地址周期内，这四条线上传输的是总线命令；在数据周期内，传输的是字节使能信号，用来表示在整个数据期中， AD0～AD31上哪些字节为有效数据。

3．接口控制信号

$\overline{FMAME}$s / t / s：帧周期信号。由当前主设备驱动，表示一次访问的开始和持续时间。 $\overline{FMAME}$无效时，是传输的最后一个数据周期。

$\overline{IRDY}$ s / t / s：主设备准备好信号。该信号有效表明发起本次传输的设备（主设备）能够完成一个数据期。它要与$\overline{TRDY}$配合使用，当这两者同时有效时，才能进行完整的数据传输，否则即为等待周期。在写周期，该信号有效时，表示有效的数据信号已在AD0～AD31中建立；在读周期，该信号有效时，表示主设备已做好接收数据的准备。

$\overline{TRDY}$ s / t / s：从设备准备好信号。该信号有效表示从设备已做好完成当前数据传输的准备工作，此时可进行相应的数据传输。同样，该信号要与$\overline{IRDY}$配合使用，这两者同时有效数据才能进行完整传输。在写周期内该信号有效表示从设备已做好了接收数据的准备。在读周期内，该信号有效表示有效数据已被送入AD0～AD31中，同理， $\overline{IRDY}$和$\overline{TRDY}$的任何一个无效时都为等待周期。

$\overline{STOP}$ s / t / s：停止数据传送信号，该信号由从设备发出。当它有效时，表示从设备请 求主设备终止当前的数据传送。

$\overline{LOCK}$s / t / s：锁定信号。是由PCI总线上发起数据传输的设备控制的，如果有几个不同的设备在使用总线，但对$\overline{LOCK}$信号的控制权只属于一个主设备（由$\overline{GNT}$信号标定）。当$\overline{LOCK}$信号有效时，表示驱动它的设备所进行的操作可能需要多个传输才能完成，如果对某一设备具有可执行的存储器，那么它必须能实现锁定，以便实现主设备对该存储器的完全独占性访问。对于支持 锁定的目标设备，必须能提供一个互斥访问块，且该块不能小于16 个字节。连接系统存储器的主桥路也必须使用$\overline{LOCK}$。

IDSEL in：初始化设备选择信号。在参数配置读写传输期间，用作片选信号。

$\overline{DEVSEL}$s / t / s：设备选择信号，由从设备驱动，该信号有效时，表示驱动它的设备已成为当前访问的从设备。它有效表明总线上的某一设备已被选中。

4．仲裁信号

$\overline{REQ}$t / s：总线请求信号。该信号一旦有效即表示驱动它的设备要求使用总线。它是一个点到点的信号线，任何主设备都应有自己的$\overline{REQ}$信号。

$\overline{GNT}$t /s：总线允许信号。用来向申请占用总线的设备表示其请求已获批准。这也是一 个点到点的

信号线，任何主设备都应有自己的$\overline{GNT}$信号。

5．错误报告信号

为了能使数据可靠、完整地传输，PCI局部总线标准要求所有挂于其上的设备都应具有错误报告线。

$\overline{PERR}$ s / t / s：数据奇偶校验错误报告信号，但是该信号不报告特殊周期中的数据奇偶 错。一个设备只有在响应设备选择信号$\overline{DEVSEL}$和完成数据期之后，才能报告一个$\overline{PERR}$。 对于每个数据接收设备，如果发现数据有错误，就应在数据收到后的两个时钟周期将$\overline{PERR}$ 激活。该信号的持续时间与数据期的多少有关，如果是一个数据期，则最小持续时间为一 个时钟周期；若是一连串的数据期并且每个数据期都有错，那么$\overline{PERR}$的持续时间将多于一个时钟周期。由于该信号是持续的三态信号，所以该信号在释放前必须先驱动为高电平。另外，对数据奇偶错的报告不能丢失也不能推迟。

$\overline{SERR}$o / d：系统错误报告信号。该信号用于报告地址奇偶错，特殊命令序列中的数据奇偶错，以及其他可能引起灾难性后果的系统错误。 $\overline{SERR}$是漏极开路信号，由返遣错误的单元驱动，在一个PCI时钟内有效。 $\overline{SERR}$信号的发出和时钟同步，因而满足总线上所有其他信号的建立时间和保持时间的要求。

6．中断信号

中断在PCI总线上是可选用的，低电平有效，用漏极开路方式驱动。同时，此类信号的建立和撤销是与时钟不同步的。PCI为每一个单功能设备定义一根中断线。对于多功能设备或连接器，最多可有4条中断线。对于单功能设备，只能使用 $\overline{INTA}$，其余3条中断线无意义。

PCI局部总线有四条中断线，定义如下：

$\overline{INTA}$o / d：中断A，用于请求一次中断。

$\overline{INTB}$ o / d：中断B，用于请求一次中断并只在多功能设备上有意义。

$\overline{INTC}$o / d：中断C，功能同中断B。

$\overline{INTD}$o / d：中断D，功能同中断B。

多功能设备上的任何一种功能都能连到任何一条中断线上。中断寄存器决定该功能用哪一条中断线去请求中断。如果一个设备只用一条中断线，则这条中断线就称为INTA，如果该设备用了两条中断线，那么它们就称为INTA和INTB，依此类推。对于多功能设备，可以是所有功能用一条中断线，也可以是每种功能有自己的一条中断线，还可以是上两种情况的综合，一个单功能设备不能用一条以上的中断线去申请中断。

系统供应商在对PCI连接器的各个中断信号和中断控制器进行连接时，其方法是随意的，可以是线或方式、程控电子开关方式，或者是二者的组合，这就是说，设备驱动程序对于中断共享事先无法作出任何假定，即它必须能够给任何逻辑设备提供中断。

7．高速缓存（Cache）支持信号

为了使具有可缓存功能的PCI存储器能够和贯穿写（Write-Through）或回写（Write- Back）的Cache相配合工作，可缓存的PCI存储器应该能实现两条高速缓存支持信号作为 输入。如果可缓存的存储器位于PCI总线上，那么连接回写式Cache和PCI的桥路必须要利用两条引脚，且作为输出，而连接贯穿写式Cache的桥只需要实现一个信号。上述两个信号的定义如下：

$\overline{SBO}$ in / out：双向试探返回信号（Snoop Backoff）。当其有效时，说明对某修改行的一次命中，所访问的数据为无效数据。当 $\overline{SBO}$ 无效而SDONE有效时，说明PCI发起方正在访问存储器的有效行并可进行高速缓存的操作。

SDONE in / out：监听完成信号（Snoop Done），表明对处理器Cache对主存的监听状态。当其无效时，说明监听仍在进行，否则表示监听已经完成。

8． 64位总线扩展信号

如果要进行64位扩展，以下信号都要使用。

AD32～AD63 t / s：扩展的32位地址和数据多路复用线，在地址周期（如果使用了双地址周期DAC命令且$\overline{\text{REQ64}}$有效时）这32条线上含有64位地址的高32位，否则它们是保留的；在数据周期，当$\overline{\text{REQ64}}$和$\overline{\text{ACK64}}$同时有效时，这32条线上含有高32位数据。

$\overline{\text{C/BE4}}$～7 t / s：总线命令和字节使能多路复用信号线。在数据周期，若$\overline{\text{REQ64}}$和$\overline{\text{ACK64}}$同时有效时，该 4 条线上传输的是表示数据线上哪些字节是有意义的字节使能信号。如$\overline{\text{C/BE4}}$对应第4个字节，$\overline{\text{C/BE5}}$对应第5个字节。在地址周期内，如果使用了DAC命令且$\overline{\text{REQ64}}$信号有效，则表明$\overline{\text{C/BE4}}$～7上传输的是总线命令，否则这些位是保留的且不确定。

$\overline{\text{REQ64}}$ s / t / s： 64位传输请求。该信号由当前主设备驱动，表示本设备要求采用64位通路传输数据。它与$\overline{\text{FMAME}}$有相同的时序。

$\overline{\text{ACK64}}$ s / t / s： 64位传输认可。表明从设备将用64位传输。此信号由从设备驱动，并且和$\overline{\text{DEVSEL}}$具有相同的时序。

PAR64 t / s：奇偶双字节校验。是AD32～AD63和$\overline{\text{C/BE4}}$～7的校验位。当$\overline{\text{REQ64}}$有效且$\overline{\text{C / BE0}}$～3上是DAC命令时， PAR64将在初始地址周期之后一个时钟周期有效，并在DAC命令的第二个地址周期后的一个时钟周期失效。当$\overline{\text{REQ64}}$和$\overline{\text{ACK64}}$同时有效时，PAR64在备数据期内稳定有效，并且在$\overline{\text{IRDY}}$或$\overline{\text{TRDY}}$发出后的第一个时钟处失效。PAR64信号一旦有效，将保持到数据周期完成之后的一个时钟周期。该信号与AD32～AD63的时序相同，但延迟一个时钟周期。该信号线在任何给定的总线周期内应保证连同AD32～AD63和$\overline{\text{C/BE4}}$～7在内的所有信号线上的“1”的个数为偶数（偶校验）或者为奇数（奇校验）。在发送时产生而在接收时进行校验。

8.4.3 PCI总线的操作

一．总线命令

总线命令是由主设备发向从设备，其作用是规定主、从设备之间的传输类型，它出现 于地址周期的$\overline{C}$/$\overline{BE}$0～3上。这里的主设备是指通过仲裁而获得总线控制权的设备；从设备是指由：$\overline{C/BE0}$～3上命令及AD0～A上的地址所选中的目标设备。

表8．1给出了总线编码及类型说明。其中，命令编码中的1表示高电平，0表示低电平。

1．中断应答命令

中断应答命令是一个读命令，执行主设备从申请中断的从设备中读回中断矢量的操作。

2．特殊周期命令

该命令为PCI总线提供了一个简单的信息广播机制，通报处理器的状态或在各个从设备之间传递信息。

$\overline{C/BE}_{0\sim3}$	命令类型说明
0 0 0 0	中断应答(中断识别)
0 0 0 1	特殊周期
0 0 1 0	I/O 读(从 I/O 地址中读数据)
0 0 1 1	I/O 写(向 I/O 地址中写数据)
0 1 0 0	保留
0 1 0 1	保留
0 1 1 0	存储器读(从内存空间映像中读数)
0 1 1 1	存储器写(向内存空间映像写数据)
1 0 0 0	保留
1 0 0 1	保留
1 0 1 0	配置读
1 0 1 1	配置写
1 1 0 0	存储器多行读(存储器重复读)
1 1 0 1	双地址周期
1 1 1 0	存储器线读(高速缓冲存储器读)
1 1 1 1	高速缓冲存储器写

表8．1 总线命令表

3．I／O读命令

该命令用来从一个映射到I／O地址空间的设备中读取数据。

4．1／O写命令

该命令用来向一个映射到1／O地址空间的设备写入数据。

5．保留命令

保留命令编码留作将来使用。PCI的任何设备都不能将它们挪作它用，任何设备也不允许对保留命令出反应。

6．存储器读命令

该命令用来从一个映射到存储器地址空间的设备读取数据。

7．存储器写命令

该命令用来向一个映射到存储器空间的设备写入数据。

8．配置读命令

该命令用来从每个设备的配置空间读取数据。

9．配置写命令

该命令向每个设备的配置空间写入数据。

10．存储器多行读命令

该命令的作用是试图在主设备断开连接之前读取多行高速缓存数据。存储控制器应保证，只要$\overline{\text{FMAN}}$效，就连续不断地发存储器请求。该命令预定用于大块连续数据的传输。

11．双地址周期（DAC）

命令 该命令用于传送64位地址给支持64位寻址的设备。只支持32位寻址的目标把这种命 令当作保留待，而对该命令不响应。

12．存储器一行读命令

该命令与存储器读命令不同之处在于它还表示主设备要求读取多于两个32位的PCI数据周期，即进行据传送。此时，一次读一行缓存范围内所有数据，而不是一个单一的存储器周期。

13．存储器写无效命令

该命令与存储器写命令不同之处是它要保证最小的传输量是一个高速缓存的行，即主设备要在一次P中将寻址的高速缓存行的每个字节都写入，写入后发布写无效命令，用于维护Cache一致性的写无效协议。

二．命令使用规则

所有PCI设备都是配置(读和写)命令的目标，都必须做出应答。对其他的命令则有选择地 响应。命令执保证I／O(读和写)命令的执行顺序。有重定位功能或寄存器的目标设备 应能通过配置寄存器映射到存储空就为没有I／O空间设备的使用提供了一种选择。当这种映射实现时，无论设备映射到I／O空间还是存储器命令执行规则都对系统设计者 提供保证。

总线主控可以根据需要使用任选指令，目标(从设备)也可根据需要而选用指令，但如果它 选用了基本指令，它就必须支持所有存储器命令。否则，就必须利用别名将这些为优 化性能而设的命令（存储器一行储器多行读和存储器写无效命令），转变为基本的存储 器命令。例如，一个从设备可以不实现存储器一行令，但是它必须能接受该命令的请求 ，并按存储器读命令来处理。同理，一个从设备可以不实现存储器写但它必须能接受该命令的请求，并按存储器写命令来处理。

对于系统存储器的数据读写，建议在主设备支持的情况下尽量采用存储器写无效命令和存 储器行读命果主设备确实不能支持上述优化性能的命令，可采用存储器读写命令。

对于使用存储器读命令的主设备，所有命令可进行任何长度的访问。

三． PCI总线协议

PCI的总线传输机制是突发成组传输。一个突发分组由一个地址周期和一个（多个）数据周期组成。存储器空间和I／O空间的突发传输。这里的突发传输是指主桥（位于主处理器和PCI总线之间）可以将多写访问在不产生副作用的前提下合并为一次传输，一个设备通过将基址寄存器的预取位置1，来表示允许预和合并写数据。一个桥可利用初始化时配置软件所提供的地址范围，来区分哪些地址空间可以合并，哪些并。当遇到要写的后续数据不可预取或者一个对任何范围的读操作时，在缓冲器的数据合并操作必须停止的合并结果清洗，但其后的写操作，如果是在预取范围内，便可与更后面的写操作合并，但无论如何不能并过的数据合并。

只要处理器发出的一系列写数据（双字）所隐含的地址顺序相同，主桥路总是可以将它们组成突发数于从处理器中发出的I／O操作不能被组合，所以这种操作一般只有一个数据周期。

在PCI总线中，除了$\overline{\text{RST}}$、$\overline{\text{INTA}}$、$\overline{\text{INTB}}$、$\overline{\text{INTC}}$、$\overline{\text{INTD}}$之外，其他所有信号都在时钟上升沿被采个信号都有相对于时钟前沿的建立和保持时间，在此期间不允许有信号跳动，该时间一过，信号的变化就了。这样的时间范围，对AD0～AD31、AD32～AD63、PAR、PAR64和IDSEL信号，只是在适当的时钟沿对$\overline{\text{LOCK}}$、$\overline{\text{IRDY}}$、$\overline{\text{TRDY}}$、$\overline{\text{FMAME}}$、$\overline{\text{DEVSEL}}$、$\overline{\text{STOP}}$、$\overline{\text{REQ}}$、$\overline{\text{GNT}}$、$\overline{\text{REQ64}}$、$\overline{\text{ACK64}}$、$\overline{\text{SBO}}$、$\overline{\text{SDONE}}$、$\overline{\text{SERR}}$，在每个时钟沿都存在；对$\overline{\text{PERR}}$、$\overline{\text{C／BE0}}$～3、$\overline{\text{C/BE4}}$～7（用作总线命令），只在$\overline{\text{FM}}$一次有 效的时钟沿存在。$\overline{\text{C／BE0}}$～3和$\overline{\text{C/BE4}}$～7（用作字节允许）在地址段或数据段完成后的那个时钟

有时间限制。$\overline{IRQA}$、$\overline{IRQB}$、$\overline{IRQC}$、$\overline{IRQD}$和$\overline{RST}$则不受此限制。

1. PCI总线的传输控制

PCI总线上所有的数据传输基本上都是由以下三条信号线控制的：

$\overline{FMAME}$：由主设备驱动，指明一个数据传输的起始和结束。

$\overline{IRDY}$：由主设备驱动，允许插入等待周期。

$\overline{TRDY}$：由从设备驱动，允许插入等待周期。

一般来说， PCI总线的传输遵循如下管理规则：

（1）$\overline{FMAME}$和$\overline{IRDY}$定义了总线的忙、闲状态。当其中一个有效时，总线是忙的；两个都无效时，总空闲状态。

（2）一旦$\overline{FMAME}$信号被置为无效，在同一传输期间不能重新设置。

（3）除非设置了$\overline{IRDY}$信号，一般情况下不能设置$\overline{FMAME}$信号无效。

（4）一旦主设备设置了$\overline{IRDY}$信号，直到当前数据期结束为止。主设备不能改变$\overline{IRDY}$信号和$\overline{FMAM}$状态。

2. PCI的编址

PCI定义了三个物理地址空间：存储器地址空间、I/O地址空间和配置地址空间，前两个是一般总线都空间；第三个是用以支持PCI硬件配置的特殊空间。

PCI总线的编址是分布式的，每个设备都有自己的地址译码，从而省去了中央译码逻辑。PCI支持两种设备地址译码：正向译码和负向译码。所谓正向译码就是每个设备都监视地址总线上的访问地址是否落在范围内，因而速度较快。而负向译码是指该设备要接受未被其他设备在正向译码中接受的所有访问，因此码方式只能由总线上的一个设备来实现。由于它要等到总线上其他所有设备都拒绝之后才能译码，所以速然而，负向译码对于标准扩展总线这类设备是很有用的，因为这类设备必须响应一个很零散的地址空间。和反向译码设备都不对保留的总线命令发出$\overline{DEVSEL}$响应信号。

（1）I / O地址空间

在I / O地址空间，全部32位AD线都被用来提供一个完整的地址编码（字节地址），使得要求地址精确一级的设备不需多等一个周期就可完成地址译码（产生$\overline{DEVSEL}$信号），也使负向地址译码节省了一个时钟

在I / O访问中， AD0～AD1这两位很重要，并要与$\overline{C / BE0}$～3配合，才能进行一次有效的访问。

（2）内存地址空间

在存储器访问中，所有的目标设备都要检查AD0～AD1，要么提供所要求的突发传输顺序，或者执行备断开操作。对于所有支持突发传输的设备都应能实现线性突发性传输顺序，而高速缓存的行切换不一定存储器地址空间，用AD2～AD31译码得到一个双字地址的访问。在线性增长方式下，每个数据周期过后，一个DWORD（4个字节）增长，直到对话结束。在存储器访问期间，AD0AD1的含义如下：

当AD0AD1为00时，突发传输顺序为线性增长方式； AD0AD1为01时，为高速缓存行切换方。式； 为1X时，为保留。

（3）配置地址空间

在配置的地址空间中，要用AD2～AD7将访问落实到一个DWORD地址。当一个设备收到配置命令时，IDSEL信号成立且AD0AD1为00，则该设备即被选为访问的目标。否则就不参与当前的对话。如果译码出符合某桥路的编号，且AD0AD1为01，则说明配置访问是对该桥后面的设备，即不与桥直接连接的设备。

（4）字节校正

用字节使能信号$\overline{C / BE0}$～3来指出哪些字节带了有意义的数据，在每个数据周期内，可以自由改变字能，使之对传输数据的实际含义和有效部分进行界定，这一功能称作字节校正或字节对齐。

（5）总线的驱动与过渡

为了避免多个设备同时驱动一个信号到PCI总线上而产生竞争，在一个设备驱动到另 一个设备之间设

个过渡期，又称为交换周期。在时序图上，交换期用"→←"来表示。

在每个地址周期和数据周期，所有的AD线都必须被驱动到稳定的状态（数据），即使是在当前数据传涉及到的字节所对应的AD线也不例外。在实际应用中，如果对功耗要求较高时，为尽量减少由于总线上信所造成的功耗，对当前总线周期中不用的字节用与前一周期相同的数据去驱动它们。

四． PCI总线的数据传输过程

PCI的数据传输过程包括读传送、写传送、传送终止等。由于篇幅的原因，本节只介绍PCI总线上的读（读操作）。

时序图8．20中示出了参与32传送的各种重要信号之间的关系。实线表示正被当前总线主控或目标驱动号；虚线表示没有设备驱动的信号，但若此虚线处在基准位置时，仍然可表示它具有一个稳定的值；当三虚线画在高、低状态之间时，说明它的值是不稳定的（如，AD线或$\overline{C/BE}$线）；当一实线变成连续的点画表明它由原来的被驱动状态变成了现在的三态；当一实线由低向高跳变后成为连续的点画线时，则说明该过预充电变为高电平，然后变成三态（释放）。

图8．20表示了PCI总线上的一次读操作中有关信号的变化情况。

从图中可看出，一旦$\overline{FMAME}$信号有效，地址周期就开始，并在时钟2的上升沿处稳定有效。在地址周AD0～AD31上包含有效地址，$\overline{C/BE0}$～3上含有一个有效的总线命令。数据期是从时钟3的上升沿处开始此期间， AD0～AD31线上传送的是数据，而$\overline{C/BE}$线上的信息指出数据线上的哪些字节是有效的（即哪是当前要传输的）。要特别指出的是，无论是读操作还是写操作，从数据周期的开始一直到传输的完成，的输出缓冲器必须始终保持有效状态。

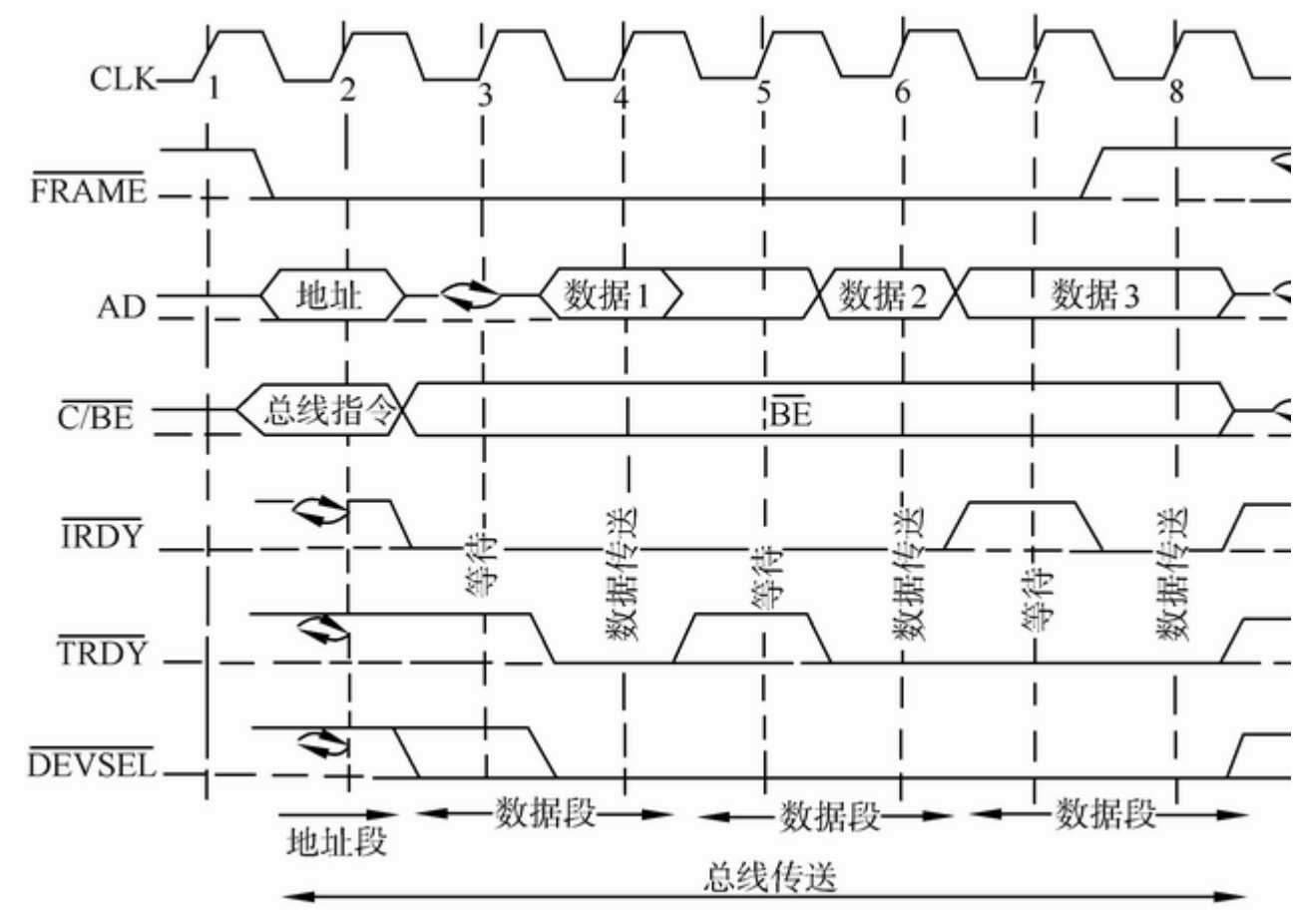

图8．20 PCI的基本读操作

图中的$\overline{DEVSEL}$信号和$\overline{TRDY}$信号是由地址周期内所发地址选中的设备（从设备）提供的，但要保证T

$\overline{\text{DEVSEL}}$之后出现。而$\overline{\text{IRDY}}$信号是 发起读操作（主设备）根据总线的占有情况自动发出的。数据的真正
$\overline{\text{IRDY}}$和$\overline{\text{TRDY}}$同时有效的时钟前沿进行的，这两个信号的其中之一无效时，就表示需插入等待周期，此时
数据传输。这就说明，一个数据周期可以包含一次数据传输和若干个等待周期。在图8．20中，时钟4，6，
进行了一次数据传输，而在时钟3，5，7处插入了等待周期。

在读操作的地址周期和数据周期之间，AD线上要有一个交换周期，这需要由从设备利用$\overline{\text{TRDY}}$强制实
就是$\overline{\text{TRDY}}$的发出必须比地址的稳定有效晚一拍）。但在交换周期过后并且有$\overline{\text{DEVSEL}}$信号时，从设备必须
线。

在时钟7处尽管是最后一个数据周期：但由于主设备因某种原因不能完成最后一次传输（此时$\overline{\text{IRDY}}$无
故$\overline{\text{FMAME}}$不能撤销，只有在时钟8处，$\overline{\text{IRDY}}$变为有效后， $\overline{\text{FMAME}}$信号才能撤销。

8.4.4 PCI总线开发技术

由以上内容可知，PCI总线具有严格的标准和规范，这就保证了它具有良好的兼容性，符合PCI规范的扩展卡可插入任何PCI系统可靠地工作；PCI总线可以提供高数据传送速率（132MB/s）或(264Mb / s)；PCI总线与CPU无关，与时钟频率亦无关，可适用于各种平台，支持多处 理器和并发工作；PCI总线还具有良好的扩展性，通过PCI-PCI桥路，可进行多级扩展。PCI 总线的自动配置功能使其应用更为简单、方便，由于该总线标准为元件和插件分配了相应的 配置寄存器，对于某个系统只要有嵌入的自动配置软件，就可以在系统加电时自动配置PCI 总线上的设备，为用户提供了极大的方便。PCI总线是目前PC机上最先进的一种总线。以下 分硬件和软件两个方面来说明PCI总线开发技术。

一、 PCI总线协议专用芯片

PCI总线协议非常复杂。如果开发PCI扩展板使用可编程逻辑阵列芯片来完成PCI协议的执 行，其工作量和难度都是很大的，为了简化设计可以使用第三方生产的PCI总线协议芯片． 如PLX的PCI9080、PCI9052、PCI9060ES，AMCC的S5933，S5920，Intel公司的i960RP等。

S5933是功能强大、使用方便的PCI协议控制芯片。该芯片符合PCI局部总线规范2.1版，可 以作为PCI总线目标设备，实现基本的传送要求，也可以作为PCI总线主控设备，访问其他PC I总线设备。S5920则主要用于设计总线上的目标设备不具有PCI总线控制功能。

PCI9052是PLX技术公司为扩展适配板卡推出的高性能PCI总线目标（从）模式的接口芯片 。该芯片可与多种局部总线相连，并且支持相对慢的局部总线在PCI总线上的突发传送速率 达到132MB/s。PCI9052的可编程配置直接与复用或非复用的8/16/32位局部总线相连。8位和 16位模式便于ISA卡直接向PCI卡转换。

PCI9052接口芯片作为通用PCI接口，其应用场合和范围是广泛的。随着PC机中ISA扩展槽 数量的逐渐减少直至取消，PCI扩展槽已成为PC机主板配置的主流，今后的扩展板的开发必 定是基于PCI接口的。

1．PCI9052的主要功能与特性

- 符合PCI2．1规范，支持低成本从属适配器；
- 带有五个局域总线地址空间和四个片选；
- 具有双向FIFO，可用于零等待状态突发操作；
- PCI总线的传输速度可高达132兆字节/秒；
- 支持多路复用和非多路复用的8位、16位和32位通用 局城总线；
- 支持局域总线与FCI时钟的异步运行；
- 支持Big / Little Endian编码字节转换；
- 支持来自两个局域总线的中断所生成的PCI中断；
- 可用串行EEPROM装载配置信息；
- 具有ISA模式，支持PCI总线到ISA总线的单周期存储 器(8位、16位)读写和I / O访问。

2．PCI9052的应用操作

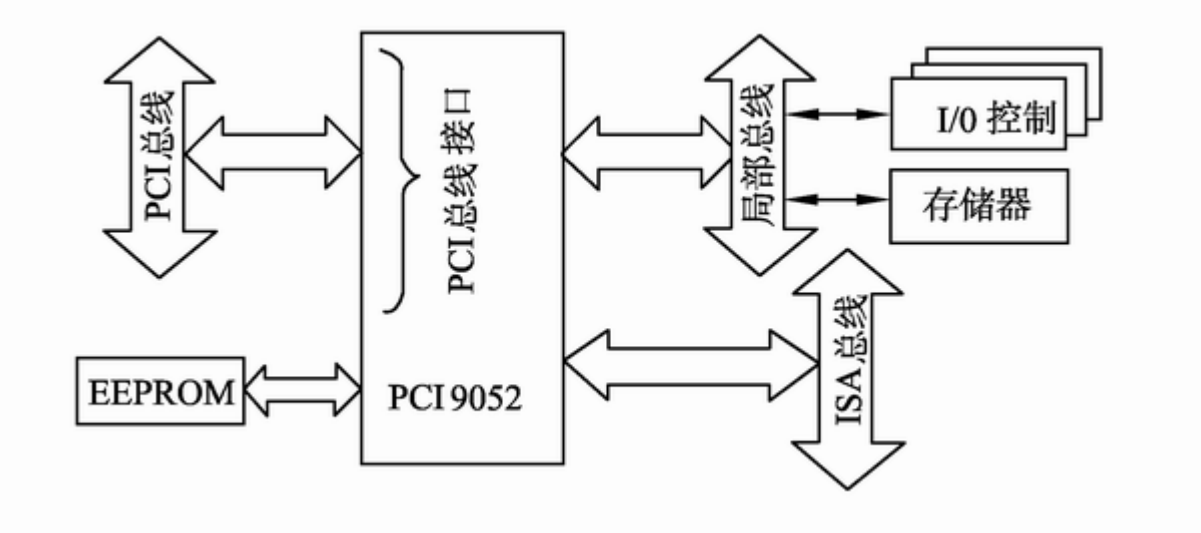

图8．21　PCI9052的信号接口示意图

（1）初始化

在上电时，PCI总线的RST信号有效，同时，PCI9052输出局部复位信号LRESET并检查EEPROM是否存在数值。若存在，则根据EEPROM中的内容设置内部寄存器，否则设为缺省值。PCI 配置寄存器只能通过EEPROM或PCI主机CPU来进行设置。

（2）复位

PCI9052总线接口在RST信号输入有效时将引起整个PCI9052的复位，并输出LRESET局部复 位信号。PCI总线上的主机可以通过设置控制寄存器中的软件复位比特来对PCI9052进行复位 ，并输出面LRESET信号。

（3）串行存储器接口

复位后，PCI9052开始读串行EEPROM，若读出的第一个字非FFFFH，则PCI9052继续读操作 ，否则认为EEPROM无效。对PCI9052来讲，EEPROM的前四个字节应为52H、90H、B5H和10H， 其中9052H为设备号，10B5H为厂商编号。

（4）寄存器访问

PCI9052的内部寄存器可通过PCI总线的主机CPU和串行EEPROM进行访问，这些内部寄存器 分为PCI配置寄存器和局部总线配置寄存器。主要有以下几种：

● 设备与厂商寄存器：该寄存器位于PCI配置寄存器的 起始处，用于标识设备类别及制造厂家；

● 状态寄存器：状态寄存器内含与PCI总线相关的事件 信息；

● 命令寄存器：用来控制设备对PCI访问的响应；

● 局部配置寄存器存储器访问的PCI基地址寄存器：系 统BIOS利用此寄存器为PCI9052局部配置寄存器的存储器访问分配一段PCI地址空间，范围为 128字节，韧始化时，主机对此寄存器写入FFFFFFFF，然后读回FFFFFF70，以确定其占用空 间为128字节；

● 局部配置寄存器I／O访问的PCI基地址寄存器：系统 BIOS利用此寄存器为PCI9052局部配置寄存器的I／O访问分配一段PCI地址空间；

● 局部地址空间0访问的PCI基地址寄存器：系统BIOS 利用此寄存器为PCI9052局部地址空间0的访问分配一段PCI地址空间；

PCI主机处理器可以直接对局部总线上的设备进行读/写操作。PCI9052配置寄存器能够访 问映射到局部的地址空间。同时片内的读写FIFO使PCI9052能够支持PCI总线与局部总线间的 高性能猝发传送。PCI总线主控访问局部总线的示意图如图8．22所示。

3．局部总线ISA接口模式

PCI9052的新功能是它直接提供给用户一个ISA逻辑接口，从而保证了ISA到PCI的平滑转换 ，另外，ISA接口还能支持8／16位存储器或I／O设备。用户通过对EEPROM的编程可将PCI9052置为ISA接口模式，在ISA接口模式下，LRESET信号将由低有效变为高有效， 并可将局部总线空间2、3配置为无复用方式。

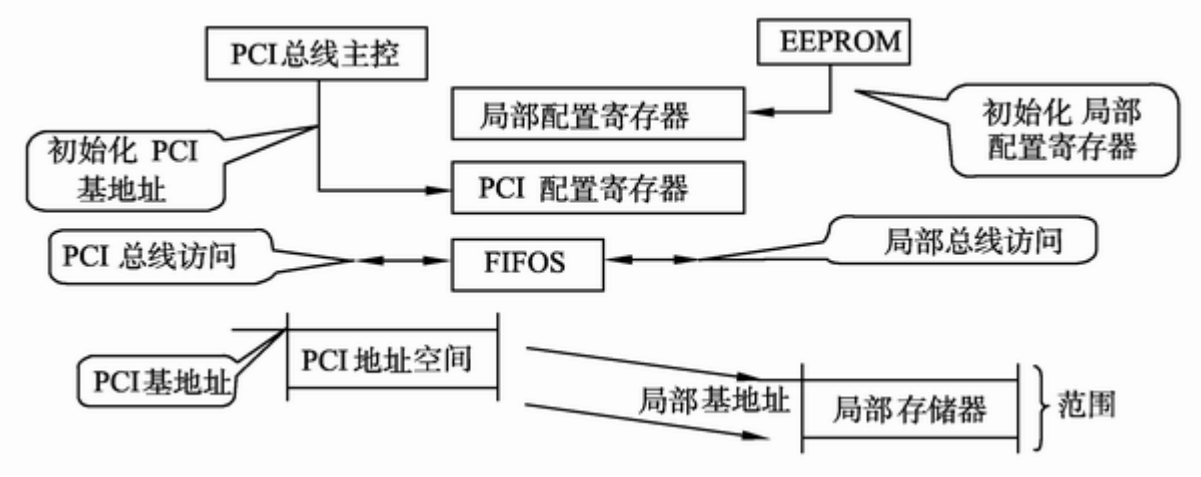

图8．22　PCI总线主控访问局部示意图

4．基于PCI9052的PCI接口卡设计

利用PCI总线目标接口芯片PCI9052设计PCI接口卡非常简便，图8．23是PCI总线数据解密卡的原理框图。图中的PCI9052设置为ISA模式，单片机与存储器使用ISA总线信号，而PCI总线 上的主机CPU可通过PCI9052直接读写存储器中的数据并控制单片机,进而控制解码操作。

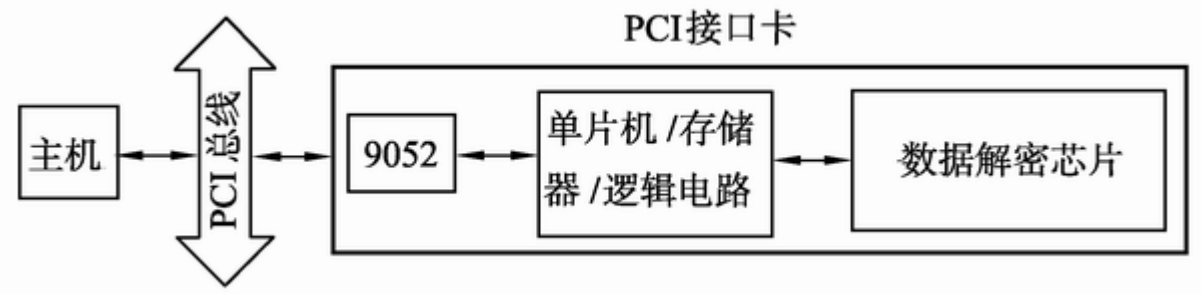

图8．23　PC I9052接口卡框图

二、 PCI设备Windows通用驱 动程序设计

1．驱动程序的模式和开发工具

设备驱动程序是指管理某个外围设备的一段代码。驱动程序不会独立地存在，而是操作系 统的一部分。通过设备驱动程序，多个进程可以同时使用这些资源，从而可以实现多进程并行运行。一般，将调用设备驱动程序的PC机程序称为用户程序。

Windows 98 可以兼容Windows 95的驱动程序，同时它又推出一个新的Win32 Drivers Mod e (WDM)驱动类型。Windows 98中有些设备（如USB设备）的驱动程序必须为WDM模式。这个 新的类型实际是在Windows NT的驱动模型的基础上增加了即插即用等内容。WDM驱动也可以 用在Windows 2000中。从长远的角度看。今后开发人员只要开发WDM驱动就可以了，但从目 前的市场情况来看，Windows 95是无法放弃的，所以WDM在还无法完全替代其它类型的设备 驱动。

Intel 80386以上的微处理器有4个特权级别：0级、1级、2级和3级，一般操作系统运行于 特权级0级上，而用户程序运行在3级上，并为了保证系统的安全性、稳定性和可移植性，对 硬件操作进行限制,因此只能通过设备驱动程序访问硬设备。对Windows 95，有VxD和打印机 驱动,VxD运行在系统的0级上，可以执行特权级指令，对任何I/O设备有全部访问权，所以大 多数硬件驱动程序都是VxD。VxD驱动通常以 ．vxd为扩展名，放在Windows\Sy stem目录下，可以在Windws 95启动时装入，也可以在程序运行时根据需要动态地载入。动 态加载有助于节约系统内存和资源。但打印机驱动程序不是VxD，它运行在3级上。由于Windows NT禁止用户模式的程序访问I/O端口，直接控制物理设备的驱动程序都是内核模式的。PCI通用驱动程序要求对各种硬件资源访问，所以应该选择工作在0级的驱动程序模式。

开发设备驱动程序有多种开发工具可以选择。主要包括：

（1）微软的软件包DeviceDriverKit（DDK）。该软件包中包括了有关设备驱动开发的文档、 编译需要的头文件和库文件、调试工具和程序范例，另外在DDK中还定义了一些设备驱动程序可以调用的系

统底层服务。但是由于DDK主要是使用汇编语言进行描述的，开发起来比较困难。

（2）Numega公司的VtoolsS。该软件包是基于C／C++的，支持BorlandC++和VisMalC++，使 用和维护比较方便。

（3）KRF—Tech公司的WinDriver。WinDriver允许用户使用MSDEV Vissual C／C++、Borla nd＼Delphi或者其他Win32编程工具软件在用户模式(UserMode)上编写设备驱动程序。

2．PCI驱动程序的特点

在设计驱动程序之前，首先要对欲控制的硬件设备进行细致地分析，更需要详细了解硬件设备的特性。硬件设备的特性会对驱动程序的设计产生重大的影响。需要了解的最主要的硬件特性包括：

（1）设备的总线结构

设备采用什么总线结构非常关键，因为不同的总线类型（如ISA和PCI）在许多硬件工作机 制上是不同的，所以驱动程序设计也不同。

（2）寄存器

要了解设置的控制寄存器、数据寄存器和状态寄存器，以及这些寄存器工作的特性。

（3）设备错误和状态

要了解如何判断设备的状态和错误信号，这些信号要通过驱动程序返回给用户。

（4）中断行为

要了解设备产生中断的条件和使用中断的数量。

（5）数据传输机制

最常见的数据传输机制是通过I/O端口（port），也就是通过CPU的IN/OUT指令进行数据读 写。PC的另一种重要的传输机制是DMA，但PCI规范不包括从属DMA的说明。

（6）设备内存

许多设备自身带有内存，PCI设备大多是采用映射的方式映射到PC系统的物理内存。有的 设备还要通过驱动程序设置设备的接口寄存器。

3．PCI设备驱动程序的功能

硬件驱动程序的功能虽然千差万别，但基本功能就是完成设备的初始化、对端口的读写操作、中断的设置、响应和调用以及对内存的直接读写。虽说Windows 9X和Windows NT的操作系统模型不同，但驱动程序所要完成的工作却是相同的，下面以Windows 9X为主进行介绍。

（1）设备初始化

PCI设备驱动程序要实现识别PCI器件、寻址PCI器件的资源和对PCI器件中断的服务。PCI 系统BIOS功能提供了BIOS的访问与控制的具体特征，所有软件（设备驱动程序、扩展ROM码 ）将通过标准中断号1AH调用BIOS功能访问特殊部件。PCI BIOS规范有完整的有关PCI BIOS 功能的描述。

在PCI设备驱动程序的初始化过程中，利用指定器件识别号（deviceid）商识别号 （vendorid）、检索号（index）搜索PCI器件，通过调用PC IBIOS确认其存在，并 确定其物理位置：总线号、器件号和功能号，这是该器件／功能在系统中的唯一寻址标志。 利用总线号、器件号和功能号可以寻址该器件／功能的PCI配置空间。

设备驱动就需要从配置空间获得硬件的参数。PCI设备的许多参数，包括所用的中断号， 端口地址的范围及I/O方式、存储器的地址及存储器映射方式等，都可以从PCI配置空间的各 基址所对应的寻址空间中得到。读写配置空间可以调用BIOS中断1AH，也可以先向配置空间 地址寄存器(0CF8H)写入总线和设备号。在前面搜索PCI器件时得到的和寄存器号，再对配置 空间数据寄存器(0CFCH)进行读写。对设备驱动来说，最重要的是获得基址寄存器(BADR)， 不能认为PCI器件资源总是设计设备时设置的初值，系统可能会根据硬件情况为PCI设备分配 新的资源。我们所设计的PCI设备使用的基址1-3都是按I/O空间映射的，而基址4是按内存方 式映射的。确定一个端口是按什么方式映射的，可以读对应端口的配置寄存器。读出后，判 断其0位，如果0位为数值0，表示其是按内存方式设置的，否则为I/O方式设置的。内存方式 和I/O方式的配置寄存器的含义参见有关文献。如果要获得基址的大小，可以向基址寄存器 写入FFFFH，然后读基址寄存器，如果是内存方式，从第4位开始的0的数目表示基址的大小 ，如果是I/O方式，则从第2位开始的0的数目表示基址的大小。

（2）端口操作

在PC机上，I/O端口寻址空间和内存寻址空间是不同的，所以处理方法也不同。I/O空间是一个64K字节的寻址空间，它不象内存有实模式和保护模式之分，在各种模式下寻址方式相 同。在Windows 9X下，用户程序可以直接使用I/O指令，而不一定非通过专门的驱动程序来 完成，所以如果软件对硬件的操作完全是通过I/O端口操作来完成的，甚至可以不用专门设 计驱动程序，直接由应用程序来完成对硬件的控制。由于PCI总线是32位的总线标准，在进 行I/O操作时通常要进行双字（DWORD）操作，而且以前大多数C/C++编译软件都没有提 供双字的函数，所以需要构造双字操作读写函数inpd/outpd。

（3）内存的读写

Winsows工作在32位保护模式下，保护模式与实模式的根本区别在于CPU寻址方式上的不同 ，这也是Windows驱动程序设计中需要着重解决的问题。Windows采用了分段、分页机制（参见第三章）。这样做最大的好处就是一个程序可以很容易地在物理内存容量不一样的、配置 范围差别很大的计算机上运行，编程人员使用虚拟存储器可以写出比任何实际配置的物理存 储器都大得多的程序。每个虚拟地址由

16位的段选择子和32位段偏移量组成。通过分段机制 ，系统由虚拟地址产生线性地址。再通过分页机制，由线性地址产生物理地址。线性地址被 分割成页目录(Page Directory)、页表(Page Table)和页偏移(Offset)三个部分。当建立一 个新的Win32进程时，操作系统会为它分配一块内存，并建立它自己的页目录、页表，页目 录的地址也同时放入进程的现场信息中。当计算一个地址时，系统首先从CPU控制器CR3中读 出页目录所在的地址，然后根据页目录得到页表所在的地址，再根据页表得到实际代码 / 数据页的页帧，最后再根据页偏移访问特定的单元。硬件设备读写的是物理内存，但应用程序 读写的是虚拟地址，所以存在着将物理内存地址映射到用户程序线性地址的问题。

从物理地址到线性地址的转换工作也是由驱动程序来完成的。在Windows 95下，使用DDK 的VMMCallMapPhysToLinear进行地址映射。驱动程序的内存映射部分主要是调用VxD 的系统服务MapPhysToLinear。在VtoolsD中这个函数的定义如下：

PVOID MapPhysToLineag（CONSTVOID*PhysAddr，DWORDnBytes，DWORDFlags）；其中第一个参数PhysAddr就是要映射的内存的物理地址的起始位置，而nBytes是内存区域的长度，Flags必须设置为0。这个函数返回的就是这段物理地址映射的线性内存地址。如果指定的内存 不能存取，函数将返回FFFFFFFFH。

（4）中断的设置、响应与调用

对中断的设置、响应与调用应该在驱动程序中完成。 对中断的调用象前面调用BIOS的1AH中断读取配置寄存空间可以由DDK的ExecInt完成。

PCI设备驱动程序应当从PCI配置寄存器的中断寄存器（INTLN）和中断引脚寄存器（INTPIN）中获得有关中断的信息。DDK还提供了响应中断事件的服务。如在Windows 95中，VPICD服务 用来管理所有硬件中断事件。PC机的硬件中断需要确定硬件中断的IRQ，对一个特定的IRQ中 断源，VPICD或者提供缺省的中断处理函数，或者允许其它VxD重载中断处理函数。在VtoolsD中，要处理硬件中断应该从VHardwareInt继承一个类。在这个类中，VtoolsD提供了编写中 断响应程序所需的功能。

4．设备驱动的调用

编写设备驱动并不是最终的目的，总是需要由用户程序来调用驱动并实现一定的功能。一 般调用设备驱动是使用CreateFile函数打开设备文件，得到一个文件句柄。

打开设备文件后，调用DeviceIoControl函数就可以同设备驱动程序交换数据了。

完成硬件操作之后，可以调用CloseHandle(hVxD)；关闭设备驱动。

5．设备驱动的进一步封装

以上是对驱动程序的初步设计。但考虑到在上面调用设备驱动时使用的DeviceIoControl 函数仍是比较复杂的，程序也不太容易具有通用性。而且，在有些开发工具中，如Visual Basic，不包括直接读写I/O端口的语句，所以可以考虑根据不同软件的需要对驱动程序进行 不同的封装。比如用DLL, ActiveX, VCL和C++类库类库进行封装。DLL可以在大多数软件环 境中进行调用。ActiveX可以在Visual Basic等可视编程环境中使用。VCL可以在Delphi和C+ + Builder中使用。考虑到许多用户使用Visual C++，所以也可使用C++类库方式。