PCI Configuration

名词说明:

PCI 为 Peripheral Component Interconnect 的缩写，它是由 Intel 所发表的另一种局部总线 (另一种为 VESA Local Bus)，以配合 Pentium 系列的微处理机为主，PCI 的最大特点在于它是和主机板不相关的接口，而且一台主机最多支持 16 台使用 PCI 接口的外围设备，这种总线架构还支持 PowerPC 等机种，因此 PCI 适配卡将可在不同工作平台上使用。

PCI 的数据总线 (data bus) 是 32 位或是 64 位宽，工作频率为 33 MHz，在 64 位宽的总线之下，每个 clock 可传送 8 个字节，因此最高的数据传输速率为每秒 33×8＝264 MB。

而 PCI 的改良版称为 PCI-X，采用六十四位的总线宽度与 133MHz 的传输速率，可以使得数据传输量为 1.064GBps，而且可以使用于现有的 PCI 适配卡，PCI-X 初期以工作站或是服务器为主。

架构说明:

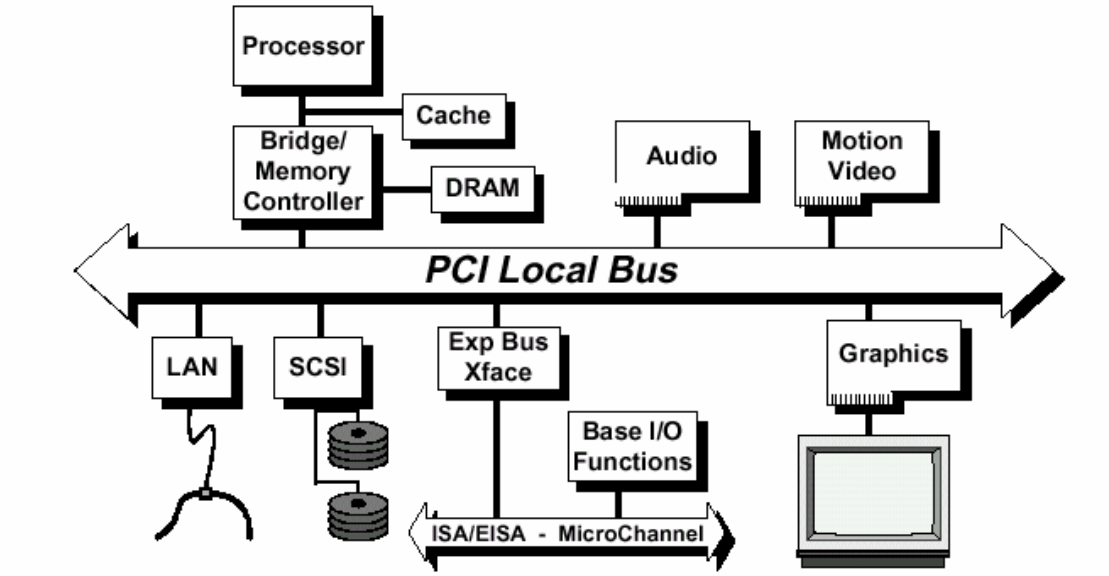


Figure 1-2: PCI System Block Diagram

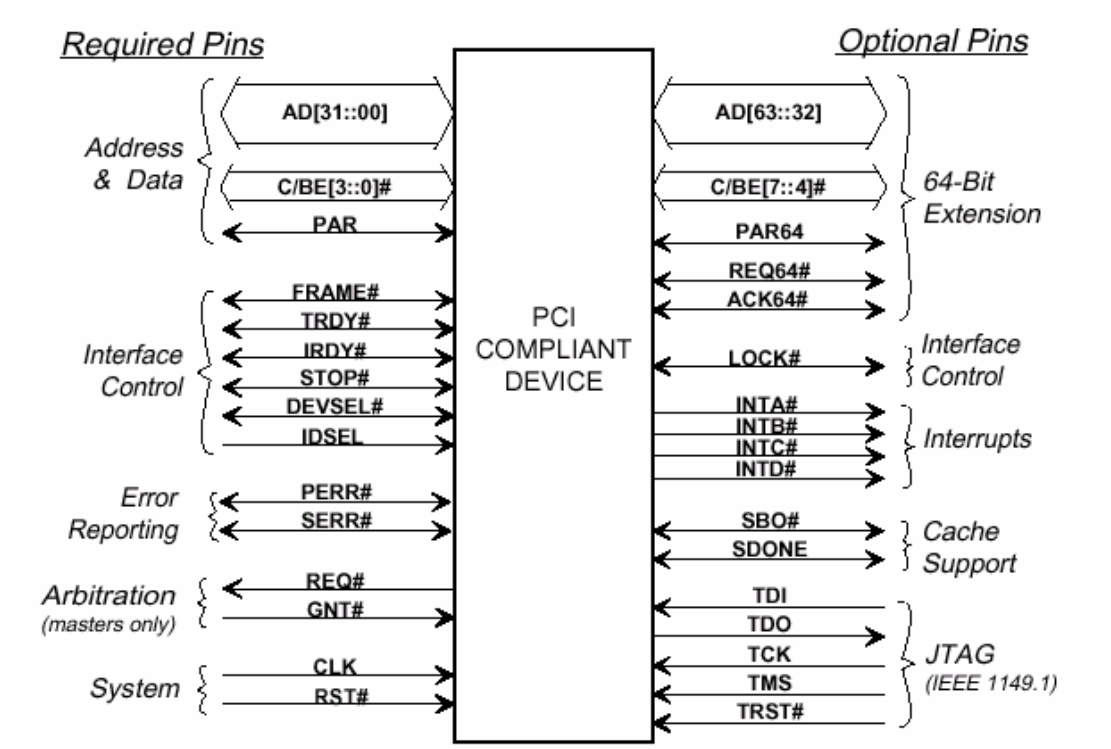


Figure 2-1: PCI Pin List

System(系统讯号):

CLK:

CLK 讯号为一个输入讯号,其提供所有交易,包括总线仲裁等的时序,PCI 装置的所有输入都是在 CLK 讯号的上升边缘被取样的.PCI 总线上的所有动作都与 PCI CLK 讯号同步,其讯号频率的范围在 0Mhz~33Mhz.

RST(Reset Signal)

当重置讯号被驱动成低态时,它会强迫所有 PCI 组态缓存器,master 及 target 状态机器与输出驱动器回到初始化状态.

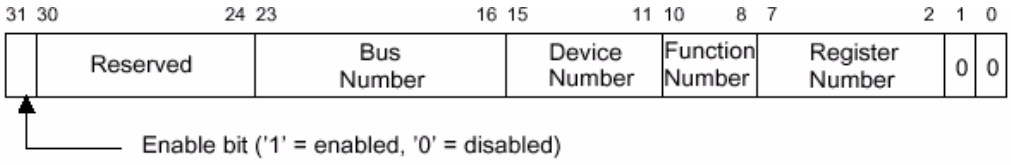


Figure 3-20: Layout of CONFIG_ADDRESS Register

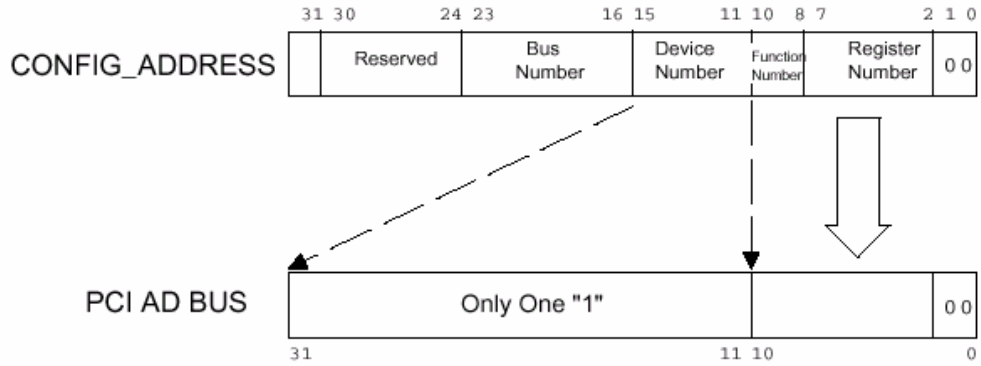


Figure 3-21: Bridge Translation for Type 0 Configuration Cycles

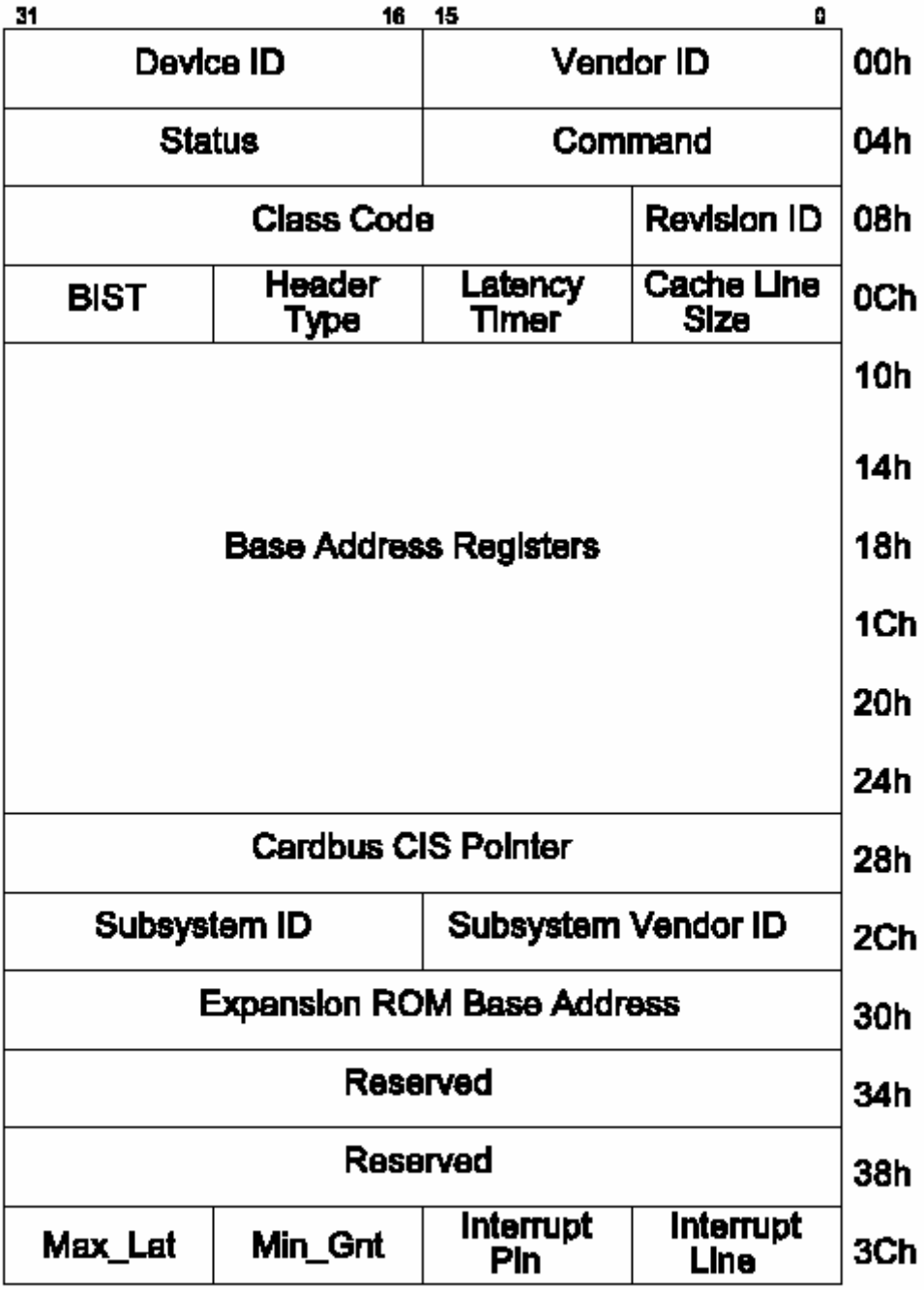


Figure 6-1: Type 00h Configuration Space Header

Device Identification

Vendor ID(Offset 00h~01h)

记录PCI厂商识别码,如Intel ID是 8086, VIA ID是 1106,Ali ID是 10B8.

Device ID(Offset 02h~03h)

记录PCI装置识别码.

以BIOS观点来看,通常为了避掉某些装置与系统的兼容性常藉由判断Vendor ID与Device ID 做不同的处理.

Device Control

Command Register(Offset 04h~05h)

指令缓存器

它是一个 16 位缓存器,提供基本装置响应或进行 PCI 存取能力的控制.

Figure 6-2 shows the layout of the register and Table 6-2 explains the meanings of the different bits in the Command register.

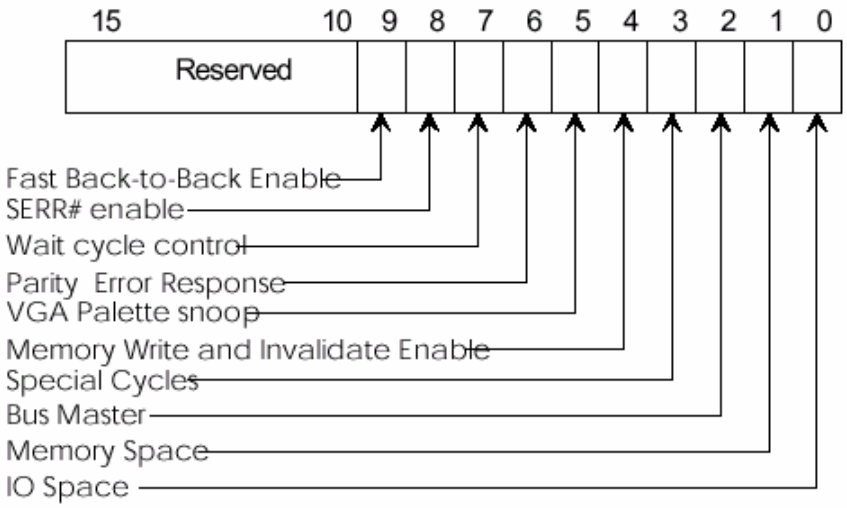


Figure 6-2: Command Register Layout

位	功能
0	IO空间(IO Space).当此位被设定为1时,显示该装置需求IO地址.
1	内存空间(Memory Space).当此位被设定为1时,显示该装置需求内存地址.
2	Bus Master.当被设定为1时,此装置能当作bus master来用,组态软件用此位来决定装置是否具有bus master能力.
3	特殊周期(Special Cycle). 当被设定为1时,装置可以监视PCI特殊周期.
4	Memory Write and Invalidate Enable. 当被设定为1时,装置可以产生Memory Write-and-Invalidate指令, 当设定为0时,装置使用Memory Write指令来取代,软件不应该致能此位,直到装置的快取线大小组态缓存器以系统快取线大小来初始化,组态软件可以用它及位2,Bus Master一起来侦测master是否能够使用Memory Write-and-Invalidate指令,假设它可以的话,则快取线大小将会被写入到master的快取线大小组态缓存器.
5	VGA调色盘监管(VGA Pallete Snoop). 当此位被设定为1时,此位指示其与VGA兼容的装置去监管对VGA Color Pallete(调色盘)缓存器的IO写入,在非VGA图形装置,重置会将此位设定为1,开启调色盘监管功能.
6	同位错误响应(Parity Error Response). 当此位被设定为1时,装置可以报告同位错误(以驱动PERR#到低组态的方式),当清除为0时,装置不会在发生同位错误事件时,驱动PERR#到低组态,不过,它仍必须设定在其

状态缓存器里的侦测到同位错误状态位.

7	逐步驱动控制(Stepping Control).控制装置是否可以进行地址/数据逐步驱动,从不使用逐步驱动的装置必须将此位以硬件接线设定为0,一定要使用逐步驱动的装置必须将此位以硬件接线设定为1,能够使用这两种方式的装置必须将此位实做成一个可读/可写的位,并且在重置后初始化为1.
8	SERR# Enable.当设定为1时,装置可以驱动SERR#线,设定为0会关闭装置的SERR#输出驱动器,此位与位6(同位错误报告)必须被设定,以便报告地址同位错误
9	Fast Back-to-Back Enable.假如Bus Master可以在第一次和第二次交易中,与不同的Target进行Fast Back-to-Back交易的话,此位元可用来启动或关闭这功能,假如所有在Bus Master所在之PCI总线上的Target是可以进行Fast Back-to-Back交易的话,组态软件可以设定此位来致能此Master进行Fast Back-to-Back交易的能力,无需担心在第一次和第二次交易里是否寻址相同的Target..
15:10	保留

Status Register

状态缓存器(Offset 06h~07h)

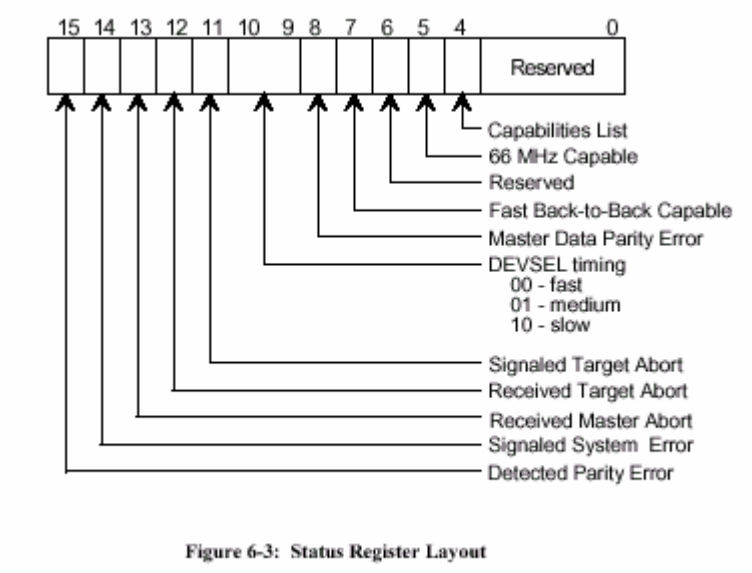


Figure 6-3: Status Register Layout

位	R/W	功能
3:0	R	保留
4	R	能力串行(Capabilities List).它是一个硬件接线,只读位,假如设定为1的话,则表示新能力串行指针缓存器必须在装置组态空间的Offset 34h里实作,在这种情况,组态程序执行装置应该去读取指针缓存器,以决定装置是否支持额外的能力(AGP,VPD等等),并且使用它们各自的组态缓存器,去组态它

		们.
5	R	66Mhz能力(66Mhz Capable). 1=装置能够在66Mhz速度下执行. 0=只能在33Mhz速度下执行.
6	R	保留
7	R	Fast Back-to-Back能力(Fast Back-to-Back Capable).此只读位元表示Target装置是否支持不同Target的Fast Back-to-Back交易,1=装置支持 0=装置不支持.
8	R/W	Master数据同位错误(Master Data Parity Error).此位只在Bus Master里实作,并且只有在下列条件时才会被设定: ● 报告的Bus Master是交易的initiator. ● (在读取过程中)它自己设定PERR#,或(在写入过程中)侦测到它被Target驱动到低态. ● 在Master的指令缓存器里的同位错误响应位被设定为1.
10:9	R	装置选择时序(Device Select (DEVSEL#)Timing). 它定义了Target装置最慢的DEVSEL#时序. 00b=Fast(快速的) 01b=Medium(中速的) 10b=Slow(慢速的) 11b=Reserved(保留)
11	R/W	发出Target Abort讯号(Signaled Target Abort).每当Target装置以Target Abort终止交易时,Target就会设定此位,不能发出Target Abort讯号的装置不需要实作此位元.
12	R/W	接收到Target Abort讯号(Received Target Abort).每当Bus Master的交易是以目前寻址的Target发出的Target Abort终止的时候,它就会设定此位.
13	R/W	接收到Master Abort讯号(Received Target Abort).每当Bus Master的交易是因Master Abort而终止(特殊周期除外)的时候,它就会设定此位.
14	R/W	发出系统错误讯号(Signaled System Error).每当装置在SERR#在线产生一个错误讯息(System Error)时,它应该设定此位.
15	R/W	侦测到同位错误(Detected Parity Error).每当装置侦测到一个同位错误时, 它应该设定此位(实时同位错误报告功能指令缓存器里的同位错误响应位关闭).

Revision ID(Offset 08h)

版本识别码缓存器

记录PCI装置版本序号,由装置制造商指派,假如制造商提供版本特定的驱动程序,这可确保OS加载正确的驱动程序.

Class Code

类别码(Offset 09h~0Bh)

它是一个 24 位只读的缓存器,它被分成三个字段:基本类别(Base Class)子类别(Sub Class)及程序接口(Programming Interface),

- 较高的字节定义功能的基本类别.
- 中间的字节定义在基本类别里的子类别.
- 较低的位元组定义程序接口.

当 OS 尝试找出一个可以配合类别驱动程序(Class Driver)一起工作的装置时,此暂存器是很有用的,如 OS 找到一个具有类别码 03h,以及子类别码为 01h 的装置(XGA),则它必须提供与其兼容的显示配接卡驱动程序和该装置一起工作.

Base Class	Meaning
00h	Device was built before Class Code definitions were finalized
01h	Mass storage controller
02h	Network controller
03h	Display controller
04h	Multimedia device
05h	Memory controller
06h	Bridge device
07h	Simple communication controllers
08h	Base system peripherals
09h	Input devices
0Ah	Docking stations
0Bh	Processors
0Ch	Serial bus controllers
0Dh	Wireless controller
0Eh	Intelligent I/O controllers
0Fh	Satellite communication controllers
10h	Encryption/Decryption controllers
11h	Data acquisition and signal processing controllers
12h - FEh	Reserved
FFh	Device does not fit in any defined classes

Base Class 00h

在类别码定义之前建立的装置.

00h 00h 所有非 VGA 装置.

00h 01h 与 VGA 兼容的装置.

Base Class	Sub-Class	Interface	Meaning
00h	00h	00h	All currently implemented devices except VGA-compatible devices.
	01h	00h	VGA-compatible device.

Base Class 01h

大量储存媒体控制器.

Base Class	Sub-Class	Interface	Meaning
01h	00h	00h	SCSI bus controller.
	01h	xxh	IDE controller (see below).
	02h	00h	Floppy disk controller.
	03h	00h	IPI bus controller.
	04h	00h	RAID controller.
	80h	00h	Other mass storage controller.

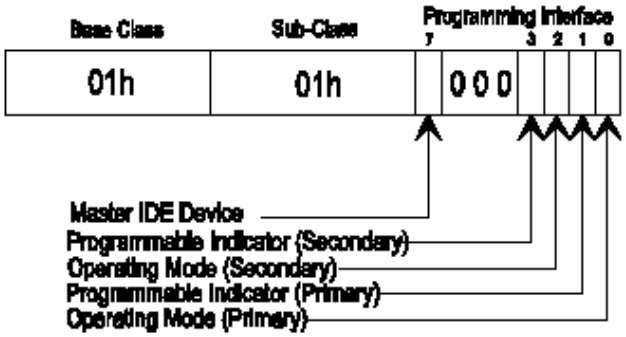


Figure D-1: Programming Interface Byte Layout for IDE Controller Class Code

子类别	程序接口	描述
00h	00h	SCSI 控制器
01h	xxh	IDE 控制器
02h	00h	软式磁盘驱动器控制器
03h	00h	IPI 控制器
04h	00h	RAID 控制器
05h	00h	其它大量储存媒体控制器

Base Class 02h

网络控制器.

Base Class	Sub-Class	Interface	Meaning
02h	00h	00h	Ethernet controller
	01h	00h	Token Ring controller
	02h	00h	FDDI controller
	03h	00h	ATM controller
	04h	00h	ISDN controller
	80h	00h	Other network controller

子类别	程序接口	描述
00h	00h	Ethernet 控制器
01h	00h	Token Ring 控制器
02h	00h	FDDI 控制器
03h	00h	ATM 控制器
04h	00h	ISDN 控制器
80h	00h	其它网络控制器

Base Class 03h

显示控制器.

Base Class	Sub-Class	Interface	Meaning
03h	00h	00000000b	VGA-compatible controller. Memory addresses 0A0000h through 0BFFFFh. I/O addresses 3B0h to 3BBh and 3C0h to 3DFh and all aliases of these addresses.
		00000001b	8514-compatible controller -- 2E8h and its aliases, 2EAh-2EFh
	01h	00h	XGA controller
	02h	00h	3D controller
	80h	00h	Other display controller

子类别	程序接口	描述
00h	00h	与 VGA 兼容的控制器,响应内存地址 000A0000h 到 000BFFFh(Video Frame Buffer, 视讯画面缓冲区),以及 IO 地址 03B0h 到 03BBh, 及 03C0h 到 03DFh,还有这些地址的所有别名 (alias).
	01h	与 8514 兼容的控制器,响应 IO 地址 02E8h 以及它的别名,02EAh 和 02EFH.
01h	00h	XGA 控制器.
02h	00h	3D 控制器.
80h	00h	其它显示控制器.

Base Class 04h

多媒体控制器.

Base Class	Sub-Class	Interface	Meaning
04h	00h	00h	Video device
	01h	00h	Audio device
	02h	00h	Computer telephony device
	80h	00h	Other multimedia device

子类别	程序接口	描述
00h	00h	视讯装置.
01h	00h	音效装置.
02h	00h	计算机电话装置.
80h	00h	其它多媒体装置.

Base Class 05h

内存控制器.

Base Class	Sub-Class	Interface	Meaning
05h	00h	00h	RAM.
	01h	00h	Flash.
	80h	00h	Other memory controller.

子类别	程序接口	描述
00h	00h	RAM 内存控制器.
01h	00h	Flash(快闪)内存控制器.
80h	00h	其它内存控制器.

Base Class 06h

桥接器装置

Base Class	Sub-Class	Interface	Meaning
06h	00h	00h	Host bridge
	01h	00h	ISA bridge
	02h	00h	EISA bridge
	03h	00h	MCA bridge
	04h	00h	PCI-to-PCI bridge
		01h	Subtractive Decode PCI-to-PCI bridge. This interface code identifies the PCI-to-PCI bridge as a device that supports subtractive decoding in addition to all the currently defined functions of a PCI-to-PCI bridge.
	05h	00h	PCMCIA bridge
	06h	00h	NuBus bridge
	07h	00h	CardBus bridge
	08h	xxh	RACEway bridge (see below)
	80h	00h	Other bridge device

子类别	程序接口	描述
00h	00h	Host / PCI 桥接器
01h	00h	PCI / ISA 桥接器
02h	00h	PCI / EISA 桥接器
03h	00h	PCI / Micro Channel 桥接器
04h	00h	PCI / PCI 桥接器
	01h	相减解碼 PCI-to-PCI 桥接器,支持除正常 PCI-to-PCI 功能之外的相减译码.
05h	00h	PCI / PCMCIA 桥接器
06h	00h	PCI / NuBus 桥接器
07h	00h	PCI / CardBus 桥接器
08h	xxh	RACEway 桥接器,RACEway 是一个 ANSI 标准(ANSI / VITA 5-1994)的交换架构(Switching Fabric)接口位元 7:1 是保留,位 0 值为 0 表示桥接器是在通透模式(Transparent Mode),值为 1 表示它是再端点模式(End-Point Mode).

80h 00h 其它桥接器型态.

Base Class 07h

简单的通讯控制器

Base Class	Sub-Class	Interface	Meaning
07h	00h	00h	Generic XT-compatible serial controller
		01h	16450-compatible serial controller
		02h	16550-compatible serial controller
		03h	16650-compatible serial controller
		04h	16750-compatible serial controller
		05h	16850-compatible serial controller
		06h	16950-compatible serial controller
	01h	00h	Parallel port
		01h	Bi-directional parallel port
		02h	ECP 1.X compliant parallel port
		03h	IEEE1284 controller
		FEh	IEEE1284 target device (not a controller)
	02h	00h	Multiport serial controller
	03h	00h	Generic modem
		01h	Hayes compatible modem, 16450-compatible interface (see below)
		02h	Hayes compatible modem, 16550-compatible interface (see below)
		03h	Hayes compatible modem, 16650-compatible interface (see below)
		04h	Hayes compatible modem, 16750-compatible interface (see below)
	80h	00h	Other communications device

子类别	程序接口	描述
00h	00h	通用的 XT 兼容序列控制器.
	01h	与 16450 兼容的序列控制器.
	02h	与 16550 兼容的序列控制器.
	03h	与 16650 兼容的序列控制器.
	04h	与 16750 兼容的序列控制器.
	05h	与 16850 兼容的序列控制器.
	06h	与 16950 兼容的序列控制器.
01h	00h	平行阜.
	01h	双向平行阜.
	02h	遵循 ECP 1.X 规格的平行阜.
	03h	IEEE 1284 控制器.
	Feh	IEEE 1284 Target 装置(非控制器).
02h	00h	多阜序列控制器.
03h	00h	通用调制解调器.
	01h	与 Hayes 兼容的调制解调器,与 16450 兼容的接口,BAR 0 映像或 IO 映像(如 BAR 型态所指示).

	02h	与 Hayes 兼容的调制解调器,与 16550 兼容的接口,BAR 0 映像或 IO 映像(如 BAR 型态所指示).
	03h	与 Hayes 兼容的调制解调器,与 16650 兼容的接口,BAR 0 映像或 IO 映像(如 BAR 型态所指示).
	04h	与 Hayes 兼容的调制解调器,与 16750 兼容的接口,BAR 0 映像或 IO 映像(如 BAR 型态所指示).
80h	00h	其它通讯装置.

Base Class 08h

基本的系统周边装置.

Base Class	Sub-Class	Interface	Meaning
08h	00h	00h	Generic 8259 PIC
		01h	ISA PIC
		02h	EISA PIC
		10h	I/O APIC interrupt controller (see below)
		20h	I/O(x) APIC interrupt controller
	01h	00h	Generic 8237 DMA controller
		01h	ISA DMA controller
		02h	EISA DMA controller
	02h	00h	Generic 8254 system timer
		01h	ISA system timer.
		02h	EISA system timers (two timers)
	03h	00h	Generic RTC controller
		01h	ISA RTC controller
	04h	00h	Generic PCI Hot-Plug controller
	80h	00h	Other system peripheral

子类别	程序接口	描述
00h	00h	通用 8259 可程序中断控制器(Programmable Interrupt Controller, PIC).
	01h	ISA PIC.
	02h	EISA PIC.
	10h	IO PIC.
	20h	IO(x)APIC 中断控制器.
01h	00h	通用 8237DMA 控制器.
	01h	ISA DMA 控制器.
	02h	EISA DMA 控制器.
02h	00h	通用 8254 定时器.
	01h	ISA 系统定时器.
	02h	EISA 系统定时器.
03h	00h	通用 RTC 控制器.

	01h	ISA RTC 控制器.
04h	00h	通用热插拔(PCI Hot-Plug).
80h	ooh	其它系统外围装置.

Base Class 09h

输入装置.

Base Class	Sub-Class	Interface	Meaning
09h	00h	00h	Keyboard controller.
	01h	00h	Digitizer (pen).
	02h	00h	Mouse controller.
	80h	00h	Other input controller.

子类别	程序接口	描述
00h	00h	键盘控制器.
01h	00h	Diditizer(pen).
02h	00h	鼠标控制器.
03h	00h	扫瞄器控制器.
04h	00h	通用游戏连接阜控制器.
	10h	游戏连接阜控制器.具有程序接口=10h 的游戏连接阜控制器表示任何在这请求/指定 IO 地址空间之功能里的基地址缓存器,在该 IO 空间的缓存器符合标准的"传统" 游戏连接阜,在 IO 区域里位移为 00h 的字节其行为与传统游戏连接阜接口相同,当读取此字节时,会传回摇杆/手把(Joystick/Gamepad)的信息,在写入此字节时会启动 RC 定时器,位移为 01h 的字节是位移为 00h 字节的别名,所有在 IO 区域里的其它字节是未指定的,且可以给制造商以自己特有的方式来使用.
80h	00h	其它输入控制器.

Base Class 0Ah

船屋系统.

Base Class	Sub-Class	Interface	Meaning
0Ah	00h	00h	Generic docking station.
	80h	00h	Other type of docking station.

子类别	程序接口	描述
00h	00h	通用的船乌系统.
80h	00h	其它型态的船乌系统.

Base Class 0Bh

处理器

Base Class	Sub-Class	Interface	Meaning
0Bh	00h	00h	386
	01h	00h	486
	02h	00h	Pentium
	10h	00h	Alpha
	20h	00h	PowerPC
	30h	00h	MIPS
	40h	00h	Co-processor

Base Class 0Ch

序列总线控制器.

Base Class	Sub-Class	Interface	Meaning
0Ch	00	00h	IEEE 1394 (FireWire)
		10h	IEEE 1394 following the 1394 OpenHCI specification
	01h	00h	ACCESS.bus
	02h	00h	SSA
	03h	00h	Universal Serial Bus (USB) following the Universal Host Controller Specification
		10h	Universal Serial Bus (USB) following the Open Host Controller Specification
		80h	Universal Serial Bus with no specific programming interface
		FEh	USB device (not host controller)
	04h	00h	Fibre Channel
	05h	00h	SMBus (System Management Bus)

子类别	程序接口	描述
00h	00h	Firewire(IEEE 1394).
	10h	使用 1394 OpenHCI 规格的 IEEE 1394.
01h	00h	ACCESS.bus
02h	00h	SSA(Serial Storage Architecture,序列储存架构).
03h	00h	使用 Universal Host Controller 规格的 USB (Universal Serial Bus,通用序列阜)控制器.
	10h	使用 Open Host Controller 规格的 USB (Universal Serial Bus,通用序列阜)控制器.
	80h	无特定程序接口的 USB 控制器.
	FEh	USB 装置(非主控器(Host Controller))
04h	00h	Fiber(光纤)Channel.
05h	00h	SMBus(System Management bus,系统管理汇流排).

Base Class 0Dh

无线控制器

Base Class	Sub-Class	Interface	Meaning
0Dh	00	00h	iRDA compatible controller
	01h	00h	Consumer IR controller
	10h	00h	RF controller
	80h	00h	Other type of wireless controller

子类别	程序接口	描述
00h	00h	与 iRDA 兼容的控制器.
01h	00h	消费型 IR 控制器.
02h	00h	RF 控制器.
80h	00h	其它型态无线控制器.

Base Class 0Eh

智能型 IO 控制器

Base Class	Sub-Class	Interface	Meaning
0Eh	00	xxh	Intelligent I/O (I2O) Architecture Specification 1.0
		00h	Message FIFO at offset 040h

子类别	程序接口	描述
00h	xxh	遵循 I20 架构规格的智能型 IO 控制器.
	00h	在相对地址(Offsset)40h 的讯息 FIFO.

Base Class 0Fh

卫星通讯控制器

Base Class	Sub-Class	Interface	Meaning
0Fh	01h	00h	TV
	02h	00h	Audio
	03h	00h	Voice
	04h	00h	Data

子类别	程序接口	描述
01h	00h	TV(电视).
02h	00h	Audio(音效).
03h	00h	Voice(语音).
04h	00h	Data(资料).

Base Class 10h

加密/解密(Encryption/Decryption)

Base Class	Sub-Class	Interface	Meaning
10h	00h	00h	Network and computing en/decryption
	10h	00h	Entertainment en/decryption
	80h	00h	Other en/decryption

子类别	程序接口	描述
00h	00h	网络与运算加密/解密.

10h 00h 娱乐加密/解密.

80h 00h 其它加密/解密.

Base Class 11h

数据撷取与讯号处理控制器

Base Class	Sub-Class	Interface	Meaning
11h	00h	00h	DPIO modules
	80h	00h	Other data acquisition/signal processing controllers

子类别 程序接口 描述

00h 00h DPIO 模块

80h 00h 其它数据撷取与讯号处理控制器.

Cache Line Size Register

快取线大小缓存器(Offset 0Ch)

此读/写组态缓存器指定了以 dword 为单位的系统快取线大小(例如:在 P6 的系统里此缓存器的值为 08h,表示其快取线大小为 8 个 dword,或 32 个字节),实作了 Memory Write-and-Invalidate 指令的 Bus Master 必须实作此缓存器,因为它必须知道快取线大小,以便确保它从快取线边界起始交易,并遵守它将整条线写入内存的约定,所以当这缓存器被设定为 0(表示组态软件尚未把快取线大小告诉它)时,Bus Master 不可以使用 Memory Write-and-Invalidate 指令,在这种情况,Master 只能够使用 Memory Write(内存写入)指令来更新内存.

装置可以限制它所支持的快取线大小数目,假如组态软件写入一个未支持的数值,则装置会将此数值当作 0 来动作.

Latency Timer

等待时间(Offset 0Dh)

等待时间以 PCI 时脉周期为单位,定义了在 bus master 起始新交易时,它可保留总线有权的最小时间量,最理想的是,每一个 bus master 都应该将此缓存器实作为可读/写的缓存器,藉此给予组态软件在 bus master 群组间分配可用的总线时间上有最大的弹性,组态软件以读取功能的 Min_Gnt(Minimum Grant 最小授权)缓存器,决定 bus master 想要的时间片段.

Header Type

表头型态缓存器(Offset 0Eh)

每一个 PCI 功能拥有保留给实作其组态缓存器用的 64 个组态 dword 区块,前 16 个 dword 的格式(Offset 00h~3Fh)与使用是由 PCI 规格预先定义的,这区域被称为装置的组态表头区域(或表头空间(Header Space)),规格目前定义了三种表头格式,称为 Header Type Zero,One 与 Two:

- **Header Type One 是为 PCI-to-PCI 桥接器定义的(01h).**

- Header Type Two 是为 PCI-to-CardBus 桥接器定义的(02h).
- Header Type Zero 是给所有除了 PCI-to-PCI 与 CardBus 桥接器以外的装置使用的(00h).

Header Type 位 7 是用来表示此装置含有除了桥接器功能以外的其它功能,固位 7=1 表示为多功能,0=表示为单功能.

Built-in Self Test

BIST内建自我测试(Offset 0Fh)

非必要的,假如装置不支持BIST的话,在读取此缓存器时,它必须要传回为0,装置的BIST是以设定位6为1的方式来请求,在完成BIST后,装置会将位6重置,但必须在二秒钟内完成重置,位0~3为0则表示成功,非零值代表装置特定的错误码.

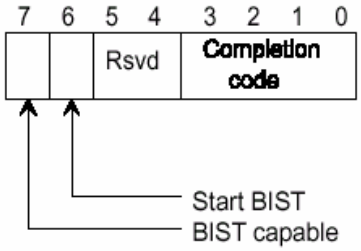


Figure 6-4: BIST Register Layout

Table 6-4: BIST Register Bits

Bit Location	Description
7	Return 1 if device supports BIST. Return 0 if the device is not BIST capable.
6	Write a 1 to invoke BIST. Device resets the bit when BIST is complete. Software should fail the device if BIST is not complete after 2 seconds.
5-4	Reserved. Device returns 0.
3-0	A value of 0 means the device has passed its test. Non-zero values mean the device failed. Device-specific failure codes can be encoded in the non-zero value.

Bass Address

基地址缓存器(Offset 10h~27h)

在开机时,系统必须自动地组态,以便让所有 IO 与内存能占用互不相干的地址范围,为了达到此目标,系统必须有能力侦测到装置需要多少个内存与 IO 地址范围,以及每一个大小.当位 0 传回 1 则表示这是一个 IO 译码器,不是内存译码器,位[31:2]是基地址字段,并且用来 IO 区块所需的大小以及设定它的起始地址.

区块大小的决定与地址范围的指定:

- 是否实作基地址缓存器?
- 它是一个内存,还是 IO 地址译码器?
- 假如它是内存译码器,它是一个 32 位还是 64 位的基地址缓存器?

- 假如它是内存译码器,与此缓存器相关的内存是可预读的还是不可预读的?
- 它需要多少内存或地址空间,并且是以什么单位来排列?

这所有信息可以很简单的用写入一个全部为 1 的值到基地址缓存器里,然后读取它来确认,传回值为 0 表示没有实作基地址缓存器,假设所读取的值不为 0,程序执行装置可从基地址栏的最低有效位向上扫瞄传回值,以找出第一个被设定为 1 的位来决定所需的内存或 IO 空间的大小.

内存范例:

假设 FFFFFFFFh 被写入在组态 13h~10h 的基地址缓存器里,然后再读取时,所得的值为 FFF00000h,由此可知有任何位可以被更改成 1 表示实作了基地址缓存器,

- 位 0 = 0 表示这是一个内存地址译码器.
- 位 [2:1] = 00b 表示它是一个 32 位内存地址译码器.
- 位 3 = 0 表示它不是可预读的内存.
- 位 20 是在基地址字段里所找到的第一个 1,此位的二进制权值为 1,048,576,表示这是一个 1MB 大小的内存地址译码器.

接下来,程序执行装置写入 32 位基地址到缓存器里,不过只有位[31:20]是可写入的,译码器接收位[31:20]并假设所指定的基地址的位[19:0]为 0,这表示基地址可以被 1MB,既所请求的内存范围大小整除,所指定的起始地址一定会被所请求的内存范围大小整除是 PCI 译码器的特性.

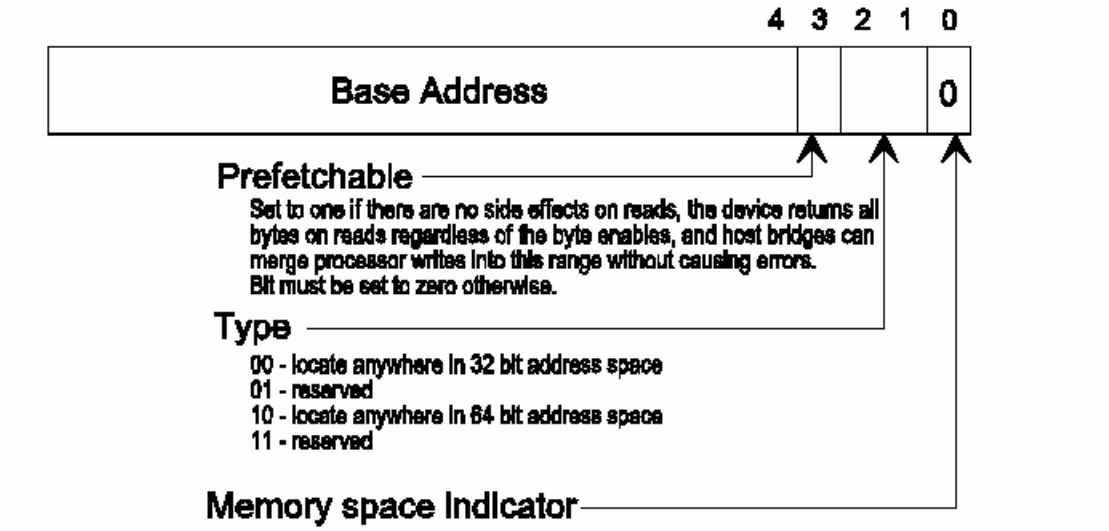


Figure 6-5: Base Address Register for Memory

IO 范例:

假设 FFFFFFFFh 被写入在组态 17h~14h 的基地址缓存器里,然后再读取时,所得的值为 FFFFFF01h,位 0 为 1,表示这是一个 IO 地址译码器,从位 2(基地址栏的最低有效位向上扫瞄)开始向上扫瞄,找到位 8 是第一个被成功地改成 1 的位,此二进权值为 256,表示此 IO 地址译码器大小为 256B,或将所得的值作补码运算后再加 1,易可得相同的值.

CardBus CIS Pointer

CardBus 与 PCI 共享的装置实作此非必要的缓存器,此字段指向 CardBus 卡的借面卡信息结构(CardBus Information Structure,CIS),此缓存器是只读的,并且包含在下列其中一个地方的 CIS 相对地址(offset):

- 在功能的装置特定组态空间里(在功能的组态空间的 Offset 40h 之后)的相对地址(Offset).
- 与其中一个装置内存基地址缓存器所指示的起始地址间之相对地址(offset).
- 在装置扩充 ROM 程序代码影像(Code Image)里的相对地址(offset).

CIS 的详细说明请参考"CardBus System Architecture".

Subsystem Vendor ID and Subsystem ID

Subsystem Vendor ID

子系统制造商识别码(Offset 2Ch~2Dh)

Subsystem ID

子系统识别码(Offset 2Eh~2Fh)

PCI 功能可以位于适配卡上或内建装置里,使用相同 PCI 核心逻辑(由 third-party(其它厂商)生厂)的两个适配卡或子系统可能会有相同制造商识别码与装置识别码(假如核心逻辑制造商以硬件接线将它们自己的识别码设定在这些缓存器里),假如在这种情况下,在 OS 辨识要加载到内存的真正装置驱动程序上,可能会有问题.

这两个强制性的缓存器被用来确认装置所在的附加卡或子系统,使用这两个缓存器,OS 可以分辨在由不同制造商制造,但使用相同 third-party 核心逻辑的适配卡或子系统之间的差异,这使得随插即用(Plug-and-Play)OS 有能力去寻找要加载到内存的正确驱动程序.

Expansion ROM Base Address

扩充 ROM 基地址缓存器(Offset 30h~33h)

对于含有装置 ROM 的装置来说是必要的,许多 PCI 装置都含有一个内含装置的装置驱动程序之装置 ROM, 在开机时,系统必须自动地组态,以便让所有 IO 与内存能占用户不相干的地址范围,为了达到此目标,系统必须有能力侦测到 ROM 需要多少个内存空间,可以很简单的用写入一个全部为 1 的值到基地址缓存器里,然后读取它来确认,它必须检查在 ROM 里的前两个位置是否有扩充 ROM 的签名,以决定 ROM 是否真的被安装,假如有安装的话,组态程序必须 SHADOW 此 ROM,并且执行其初始化程序代码.

- **位 0 = 1 致能装置的 ROM 地址译码器.**
- **位[10:1]保留.**
- **位[31:11]用来指定 ROM 的起始地址.**

假设 FFFFFFFFh 被写入到 ROM 的基地址缓存器里(位 0,既扩充 ROM 致能位被清除,所以 ROM 地址译码器不会被致能,直到指定起始内存地址之后),随后再读取此缓存器,所得的值为 FFFE0000h,这表示:

- 位 0 = 0 ,表示 ROM 地址译码器目前被关闭.
- 位[10:1]保留.
- 在基地址字段(位[31:11])里,位 17 为程序执行装置可以设定为 1 的最低有效位,其二进制权值为 128K,表示 ROM 译码器需要 128K 的内存地址,然后,程序执行装置写入一个 32 位的起始地址到此缓存器中,指定 128KB 的地址边界当作 ROM 的起始地址.

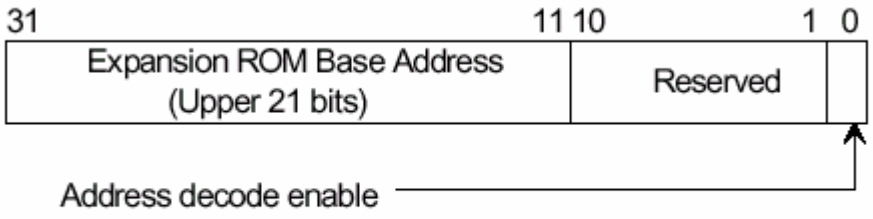


Figure 6-7: Expansion ROM Base Address Register Layout

PCI Expansion ROM Header Format

Offset	Length	Value	Description
0h	1	55h	ROM Signature, byte 1
1h	1	AAh	ROM Signature, byte 2
2h-17h	16h	xx	Reserved (processor architecture unique data)
18h-19h	2	xx	Pointer to PCI Data Structure

PCI Data Structure Format

Offset	Length	Description
0	4	Signature, the string "PCIR"
4	2	Vendor Identification
6	2	Device Identification
8	2	Pointer to Vital Product Data
A	2	PCI Data Structure Length
C	1	PCI Data Structure Revision
D	3	Class Code
10	2	Image Length
12	2	Revision Level of Code/Data
14	1	Code Type
15	1	Indicator
16	2	Reserved

ROM Header Extensions

Offset	Length	Value	Description
0h	1	55h	ROM Signature byte 1
1h	1	AAh	ROM Signature byte 2
2h	1	xx	Initialization Size - size of the code in units of 512 bytes.
3h	3	xx	Entry point for INIT function. POST does a FAR CALL to this location.
6h-17h	12h	xx	Reserved (application unique data)
18h-19h	2	xx	Pointer to PCI Data Structure

Interrupt Line

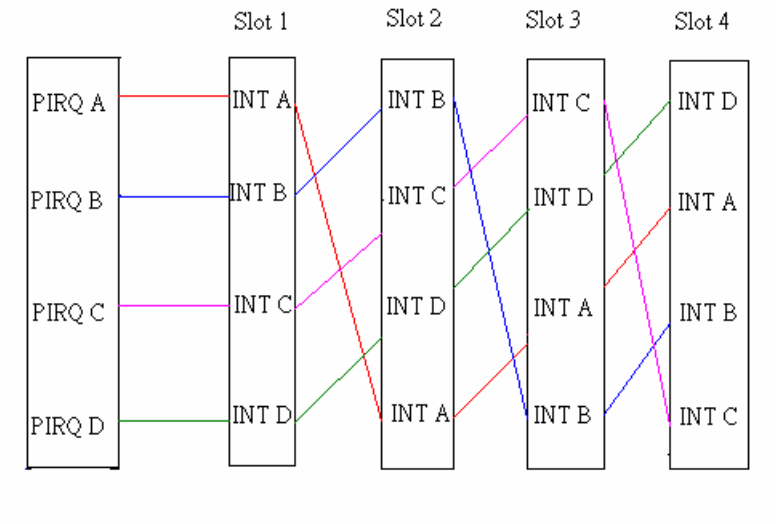
中断讯号线(Offset 3Ch)

PCI装置的中断号码配置与绕线设定处理和此字段密不可分.

PCI装置对于中断讯号的定义中,设计了以下四组中断讯号线.

INT A# , INT B#, INT C#, INT D#

在当今的许多PC系统里,内崁在PCI/ISA桥接器内的可程序化路由器都具有四个输入端,用来连接PCI中断线路,目前大多数的PCI装置是单功能装置, 单功能PCI装置只能用INT A#(不得使用INT B#,INT C#,INT D#)来产生中断请求,在多功能PCI装置上,装置设计者可以实作最多四支中断接脚,INT A#,INT B#,INT C#,INT D#,因其是内含二到八个功能的实体套件,ISA中断请求线是正缘触发(Positive edge-triggered),且不可共享,但PCI中断请求线是低态动作的(Active low),准位触发的(Level-sensitive),可共享的(Shareable)请求线,PCI中断只能连接到未被ISA装置占用的ISA中断请求在线.



在系统内存F0000到FFFFF搜寻表格的签名(Signature)($PIR),便可侦测到PCI中断路由表格.格式如下:

从表格起点算起的相对字节	大小	说明
0	4	Signature
4	2	Version

6	2	Table Size
8	1	PCI Interrupt Router’s Bus
9	1	PCI Interrupt Router’s DevFunc
10	2	PCI Exclusive IRQs
12	4	Compatible PCI Interrupt Router
16	4	Miniport Data
20	11	Reserved
31	1	Checksum
32	16	First Slot Entry
48	16	Second Slot Entry
(N+1)*16	16	nth Slot Entry

```
align   16
    public  ms_irq_route_table_sign
ms_irq_route_table_sign label byte
           db  "$PIR"
ms_irq_route_table_ver  dw  0100h
ms_irq_route_table_size dw  32+(5*16)
ms_irq_route_table_bus  db  00h
ms_irq_route_table_dev  db  VT586_P2I_DEV_FUNC
ms_irq_route_table_irq  dw  0h
ms_irq_route_table_id   dd  05861106h       ; VIA 82C586
;ms_irq_route_table_id  dd  05961106h       ; VIA 82C596
ms_irq_route_table_mini dd  0h
ms_irq_route_table_resd db  11 dup(0)
ms_irq_route_table_cksm db  0
```

紧接着是16字节的插槽项目格式,依系统主机板所使用的芯片组,内建装置与插槽数而有所不同. 格式如下:

从表格起点算起的相对字节	大小	说明
0	1	PCI Bus Number
1	1	PCI Device Number
2	1	Link Value for INT A#
3	2	IRQ Bitmap for INT A#
5	1	Link Value for INT B#
6	2	IRQ Bitmap for INT B#
8	1	Link Value for INT C#
9	2	IRQ Bitmap for INT C#
11	1	Link Value for INT D#

12	2	IRQ Bitmap for INT D#
14	1	Slot Number
15	1	Reserved

Interrupt Pin

中断讯号脚(Offset 3Dh)

中断讯号线记录器其记录该 PCI 装置使用何者 INT#讯号,BIOS则依其所提供的信息告知CHIPSET并给予正确中断绕线设定.

值为1表示其使用INT A,2则为INT B,3则为INT C,4则为INT D.

此记录器是只读并无法写入.

MIN_GNT and MAX_LAT

MIN_GNT(Minimum Grant 最小授权)(Offset 3Eh)

此为只读的组态缓存器,其值为0,代表此Bus Master对于指派给予的LT的设定值并没有特定的要求,若非为0则表示master需要多长的时间片段以得到适当的效能,以250ns为单位.

Max_LAT:Max_Lat(Offset 3Fh)

缓存器的值表示master多久会想要存取总线一次, 此为只读的组态缓存器,对于bus master来说是非必要的.

Device Status

PCI 2.2 新增新能力(New Capabilities)

组态缓存器 offset 06h~07h 为装置状态缓存器,其位 4=1 表示支持新能力,新能力指针缓存器在组态缓存器 offset 34h,其细节如下:

位 0~7 为能力识别码

位 8~15 为指向下一个新能力的指标.

识别码:	说明
00h	保留
01h	PCI Power Management Interface
02h	AGP
03h	VDP (Vital Product Data)
04h	Slot Identification
05h	MSI (Message Signaled Interrupt)
06h	Compact PCI Host Swap
7~255d	保留

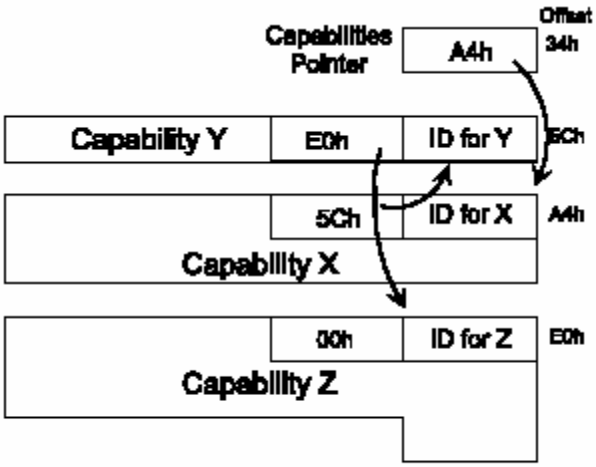
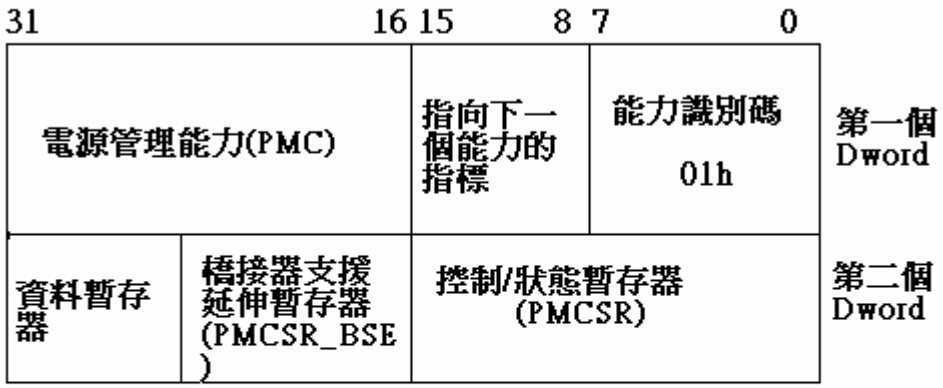


Figure 6-10: Example Capabilities List

7::0	CAP_ID	The value of 05h in this field identifies the function as message signaled interrupt capable. This field is read only.

7::0	NXT_PTR	Pointer to the next item in the capabilities list. Must be NULL for the final item in the list. This field is read only.



PMC Register:

位 说明

15:11 PME支持(PME_Support)字段,表示功能可以在该状态下,驱动PME#到低态(电源管理事件PM Event)的PM状态,位为0表示在这个PM状态下不支持PME#产生.

位	对应的PM状态
11	D0
12	D1
13	D2
14	D3 hot
15	D3 cold(功能需要辅助电源来供应PME逻辑电力)

10 D2支持位

9 D2支持位

8:6 辅助电流(Aux_Current)字段.

5 装置特定的初始化(Device-Specific Initialization,简称DSI)位.

4 保留

3 PME时脉(PME Clock)位,此位为1表示功能需要有PCI时脉存在,才能产生PME#,不需要时脉或不产生PME#的功能会以硬件接线

将本位元设定为0.

2:0 版本字段(Version).

位	遵循的规格版本
210	
001	1.0
10	1.1

PM控制/状态(PMCSR)缓存器

- 假如功能实作PME能力的话,则此缓存器含有一个用来反应先前致能之PME是否发生的PME状态(PME Status)位.
- 假如功能实作PME能力的话, 则此缓存器含有一个PME致能(PME Enable)位,它允许软件去致能或关闭功能驱动PME#讯号到低态的能力.
- 假如实作了非必要的数据缓存器的话,则此缓存器含有两个字段:
  - 允许软件选择可以透过数据缓存器来读取的信息.
  - 并提供数据缓存器数值必须相乘的比例因子.
- 软件使用此缓存器的电源状态(Power State)字段,来决定功能目前的PM状态,以及把功能转移到新PM状态.

位	读/写(R/W)	说明
15	读/写,想要将1清除为0的话,	PME状态(PME Status)位,假如功能支持PME的话,此位会反应功能是否遇到PME（即使在

	把1写入其中	此缓存器里的PME_En位将功能在发生PME事件时,驱动PME#到低态的能力关闭),假如设定为1,则功能遇到PME,而软件可以用写入1到其中的方式将它清除.
14:13	只读	数据比例(Data Scale)字段.
12:9	读/写	数据选择(Data Select)字段.
8	读/写	PME致能(PME_En)位. 1=致能功能在发生PME事件时,驱动PME#到低态的能力. 0=关闭.
7:2	只读	保留.
1:0	读/写	电源状态(Power State)字段,软件用此字段来居决定功能目前的PM状态(以读取此字段元的方式) 或是将它放入到一个新的PM状态(以写入此字段元的方式),假如软件选择功能不支持的PM状态,则写入必须正常地完成,但是写入的数据会被丢弃,且状态不变. 10 PM状态 00b D0 01b D1 10b D2 11b D3hot

相关信息与实例

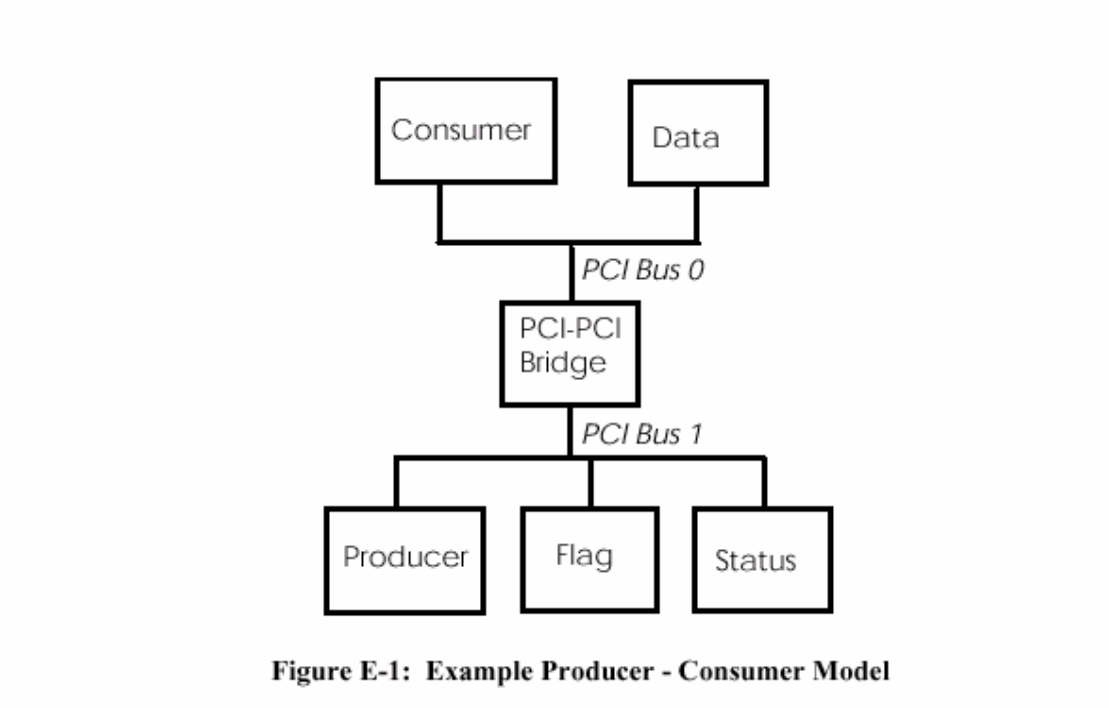


Figure E-1: Example Producer - Consumer Model

当客户要求支持 **S5 PME Wake up** 功能但 **PCI** 装置不支持时,**BIOS** 可利用下列程序来做到**:**

```
STR_SAVE_PCI_DEVICE LABEL   BYTE
        dw      MKF_PCI_SLOT1_DEV_FUNC      ;(25-11)shl 3
        dw      MKF_PCI_SLOT2_DEV_FUNC      ;(26-11)shl 3
        dw      MKF_PCI_SLOT3_DEV_FUNC      ;(27-11)shl 3
        dw      MKF_PCI_SLOT4_DEV_FUNC      ;(28-11)shl 3
        dw      MKF_PCI_SLOT5_DEV_FUNC      ;(29-11)shl 3
NO_OF_STR_PCI_TABLEequ      ($ - offset
cgroup:STR_SAVE_PCI_DEVICE)/2

    mov     si,offset cgroup:STR_SAVE_PCI_DEVICE
    mov      cx,NO_OF_STR_PCI_TABLE
loop_for_pme_d3:
    push    cx
    mov     dx, word ptr cs:[si]
    mov     ah,00h                          ; 00h
    call    read_pci
    cmp     al,0ffh
    jz      next_for_pme_d3
    mov     ah,PCI_REG_STATUS               ; 06h
```

```
        call        read_pci
        test        al, CAPABILITIES_BIT            ; 10h
        jz          next_for_pme_d3
        mov         ah,PCI_CAP_PTR                  ; 34h
read_cap_id:
        call         read_pci
        cmp         al, 00
        jz          next_for_pme_d3
        mov         ah,al
        call        read_pci
        cmp         al, PME_CAP_ID              ; 01h
        jz          found_for_pme_id
        inc         ah
        jmp         read_cap_id
found_for_pme_id:
        add         ah,PMECS_REG+1                  ; 05h
        call        read_pci
        and         al,0feh                 ; Clear & Disable PME
        call        write_pci
        dec         ah
        call        read_pci
        and         bl,0fch                 ; set power on state
        call        write_pci
        inc         ah
        call        read_pci
        or          al,01h                  ; Enable PME
        call        write_pci
next_for_pme_d3:
        inc         si
        inc         si
        pop         cx
        loop        loop_for_pme_d3
        mov         cx,3333
        call        pm_fixed_delay
        popa
for_pme_disable:
        ret
;-------------------------------------------------------------------;
```

```
;                   STACK REQUIRED                                  ;
;                                                                   ;
;     Input : (AL)    Mask of PCI register bits to test.            ;
;                 (optional)                                        ;
;             (AH)        Index of PCI register to access.          ;
;             (DL)7:3   PCI device #                                ;
;                 2:0   PCI function # on device                    ;
;             (DH)        7:0   PCI bus # of device                 ;
;                                                                   ;
;     Output: (AL)   Value read from chip set register.             ;
;             (ZF)ZR   Corresponding bits in chip set               ;
;                      register to mask are all zero.               ;
;                 NZ   Corresponding bits in chip set               ;
;                      register to mask has at least                ;
;                      one bit set.                                 ;
;                                                                   ;
;     Register destroyed : (AL)                                     ;
; NOTE :                                                            ;
;   *   Do not invoke with "ret_sp".    Invoke with "call" only. ;
;   *   Do not change IF status.                                    ;
;-------------------------------------------------------------------;

read_pci                        PROC    NEAR

        push    edx
        push    eax
        pushf

        xchg    ax,dx
        movzx   eax,ax                          ; (eax) = device & func #
        shl     eax,8                           ; (eax) = dev & func # in
                                        ; proper position.
        bts     eax,31

        mov     al,dh                           ; (al) = reg # to access.
        and     al,0fch                     ; (eax) = proper config enb.

        shl     edx,16
```

```
        mov     dx,00cf8h               ; (dx) = CONFIG_ADDRESS
        cli
        out     dx,eax
        jcxz    short $+2
        jcxz    short $+2

        shr     edx,24                  ; (dl) = reg # to access.
        and     dl,003h             ; (dl) = byte to access of reg
        or      dx,00cfch               ; (dx) = CONFIG_DATA

        in      al,dx                   ; (al) = data read from reg
        jcxz    short $+2
        jcxz    short $+2

        mov     dl,al

        popf
        pop     eax
        test    al,dl               ; For returning proper flag info.
        mov     al,dl
        pop     edx

        ret

read_pci                ENDP
```

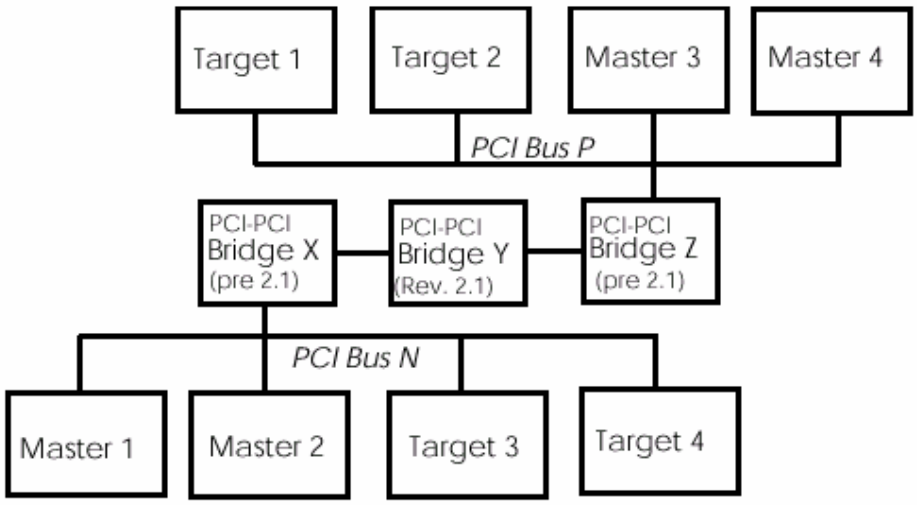


Figure E-2: Example System with PCI-to-PCI Bridges

- Host/PCI Bridge:
  通常被称为北桥(North Bridge),它连接主处理器总线与 root PCI 总线.
- PCI-to-ISA Bridge:
  通常被称为南桥(South Bridge),它连接 root PCI 总线与 ISA(或 EISA) 总线,南桥通常还整合中断控制器(Interrupt Controller),IDE 控制器(IDE Controller),USB Host 控制器(USB Controller)与 DMA 控制器(DMA Controller).
- PCI-to-PCI Bridge:
  它连接 root PCI 总线与 PCI 总线