24 32 Bits and More – The PCI Local Bus

For many PC owners, Chapters 22 and **23** must have been a great disappointment – even EISA bus and microchannel, as high-end solutions for the personal computer, run in a 50 MHz i486 with only a miserable 8 MHz and 10 MHz, respectively. This is far too slow for quick screen rewrites in a graphic oriented operating system or system interface such as Windows. Even a small dialogue window with 512 * **384** points, which only covers a quarter of the screen surface in a high resolution mode, consists of 192k pixels. Thus, in 256 colour mode, 192 points on the screen must first be saved and then rewritten, corresponding to 384 kbytes of screen data. However, the screen memory can only be rewritten if the graphic control chip on the graphic adapter is not performing a read access. This is usually only the case during horizontal and vertical returns; the 384 kbytes must, therefore, be transferred in the relatively short return time window. In a VGA card, a complete line scan including a horizontal return takes about 25 µs, the horizontal return itself approximately 4 µs. This means that the line return takes approximately a seventh of the time required by a horizontal scan. The transfer of the 384 kbytes in a 16-bit ISA system, therefore, realistically requires approximately a third of a second. This is clearly noticeable by the user. The refresh time of the screen memory is not even included in the calculation.

One solution to this problem is offered by graphic adapters containing a graphics processor, such as the TIGA or 8514/A. Another similar solution – especially when you consider the substantially higher clock frequencies of processors in comparison to bus clock speeds – is to drive the graphic system bus at the same frequency as the CPU. The local bus attempts to do just that. The first local buses located on motherboards were, therefore, nothing more than a fast interface to the display memory, so that the CPU data could be transferred far quicker than in a normal system with a standard expansion bus. Naturally, the concept can also be applied to other equipment requiring higher transfer rates. One such example would be a fast hard disk with an integrated cache memory. This leads to another problem: host adapters or controllers for hard disks are usually constructed as plug-in cards which are inserted into a bus slot. To begin with there was no standard, so for this reason the local bus slot can differ greatly from motherboard to motherboard. To rectify this incompatibility problem, Intel developed the PC1 (Peripheral Component Interconnect) *bus* and the VESA committee has developed the VL ***bus.*** They have been introduced independently of one another as local bus standards.

I would also like to mention that EISA and microchannel have recognized the threat that local bus systems represent. For this reason, the EISA specification has been extended to include the EMB (Enhanced Master Burst) cycle with a data bus width of 64 bits (EMB-66) and, additionally, the effective bus frequency has been doubled to a maximum of $16^2/_3$ MHz (EMB-133). Even microchannel has been improved, with a 64-bit streaming data procedure and a doubled clock cycle of 20 MHz, giving a maximum transfer rate of 160 Mbytes/s – the same as the 64-bit VL bus at 50 MHz. Whether these extensions actually represent a threat to local bus systems, or only increase the number of possible bus systems available, remains to be seen. Nevertheless, PCI can reach a transfer rate of 266 Mbytes/s which, of course, can only be achieved in perfect conditions. In the 80x86 processor family, only the Pentium achieves a similar or higher rate. However, PCI, as a processor independent bus system, should also serve the Alpha from DEC, for example (the Alpha version 21066 already includes a PCI interface on the processor chip). Currently, 266 Mbytes/s appears to be an enormous transfer rate. One should, however, not forget that not so long ago (almost 15 years) 640 kbytes of memory was considered as entirely sufficient and satisfactory for the next ten years. The consequences of this erroneous evaluation can now be found in their millions, including, perhaps, on your desk.

Today, PCI represents a high-end solution for powerful personal computers and workstations. As an introduction, I would like to list a few characteristics of this Intel-initiated bus system:

- coupling of the processor and expansion bus by means of a bridge,
- 32-bit standard bus width with a maximum transfer rate of 133 Mbytes/s,
- expansion to 64 bits with a maximum transfer rate of 266 Mbytes/s,
- supporting of multi-processor systems,
- burst transfers with arbitrary length,
- supporting of 5 V and **3.3** V power supplies,
- write posting and read prefetching,
- multimaster capabilities,
- operating frequencies from 0 MHz to a maximum of 33 MHz,
- multiplexing of address and data bus reducing the number of pins,
- supporting of ISA/EISA/MCA,
- configuration through software and registers,
- processor independent specification.

24.1 PCI Bus Structure

An essential characteristic of the PCI concept is the strict decoupling of the processor's main memory subsystems and the standard expansion bus. You can see the structure of the PCI bus in Figure **24.1.**

The *PCI* bridge represents the connection between the subsystems of the processor's main memory and the PCI bus. For the user, bridges are generally an invisible interface between two bus systems (and also networks). All of the PCI units (also known as *PCI* agents) are connected to the PCI bus, for example, an SCSI host adapter, a LAN adapter, an I/O unit and a graphic adapter (see Figure 24.1). Contrary to the VL bus, these units should, as far as possible, be integrated onto the motherboard, but they are mostly constructed as adapters. In total, a maximum of three slots are provided for PCI units; for example, two slots can be used for an audio and motion video unit. Motion video concerns moving pictures, which require a staggering number of complicated calculations. For this reason, the corresponding PCI unit is often very large, making integration impossible. The audio/video extensions should make the PCI bus most suitable for future multimedia applications. The interface to the expansion bus is a third type of PCI unit. This means that the standard expansion bus – whether ISA, EISA, microchannel or another bus system – can be considered here as a PCI unit. In this way, in principle, every bus system can be integrated, enabling the PCI bus to be connected at a later date. In total, it is possible to connect the PCI bus with up to ten PCI units. In Figure **24.2** you can see a typical layout of a PCI bridge.

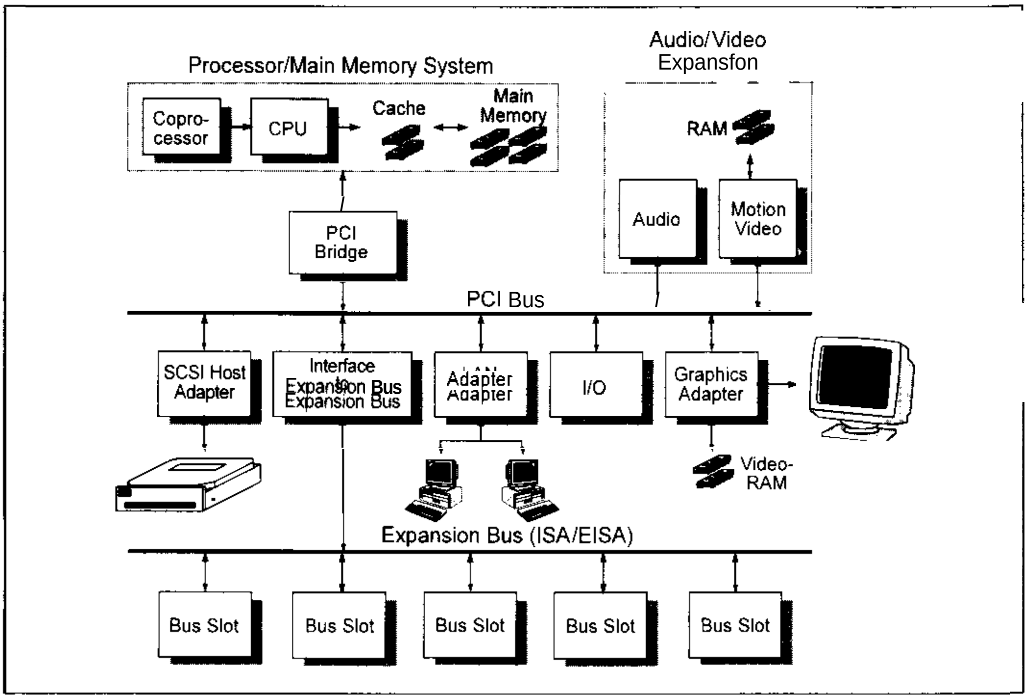


Figure 24.1: The PCI bus

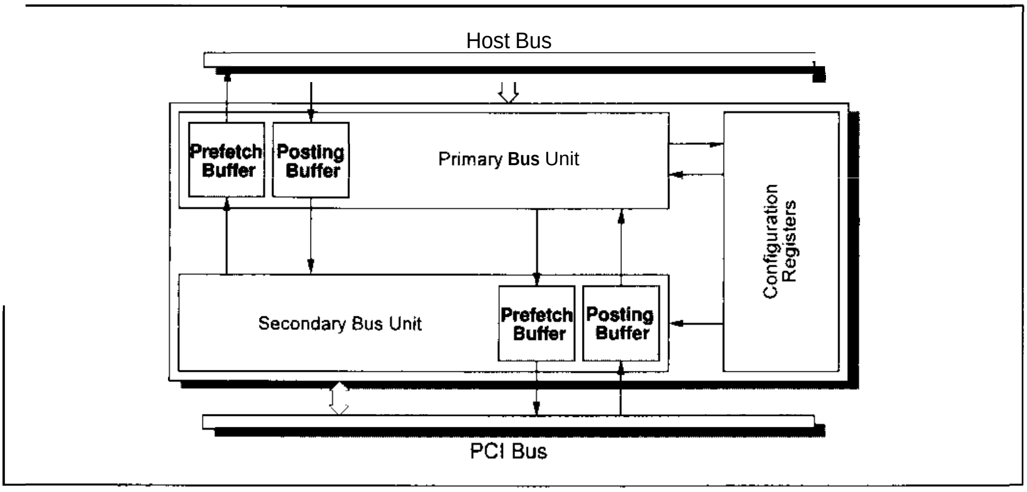


Figure 24.2: Typical layout of *a PCI bridge.*

The local bus to the host CPU, or to a PCI bridge located nearer to the host CPU, is known as the *primary* bus. The PCI bus to the PCI units, or to a further PCI bridge, is known as the secondary bus. The register of the configuration address area stores the configuration data. The prefetch buffers read in data as a reserve, if a read access has been previously performed and it is possible that further data accesses may be required. In a similar way, the posting buffers temporarily store write data in order to pass it on to the addressed bus later.

The PCI bus and its bus cycles use a multiplexing scheme in which the lines are alternately used as address and data lines. This saves on the number of lines required, but two to three clock cycles are now required for a single transfer, as the address is transferred in the first cycle, write data in the second, and read data in the third. For this reason, the maximum data transfer rate is only 66 Mbytes/s (write access) and 44 Mbytes/s (read access), with a 32-bit bus width. In addition, the PCI bus also includes a very powerful burst mode in which the address is only transferred once. Then, the sender and receiver increase the amount of data transferred for the address with each clock cycle because the address is always implicitly known. With burst mode, any number of transfer cycles can be performed. The maximum data transfer rate in burst mode increases to 133 Mbytes/s with a 32-bit data bus and 266 Mbytes/s with a 64-bit data bus. Whether or not the addressed PCI units can keep up with this is, of course, another matter.

An essential characteristic of the PCI bridge - and here lies the main difference to all other bus systems - is that it independently forms burst accesses. This means that the PCI bridge independently joins together incoming single transfer read and write operations to form burst accesses if the addresses of the individual accesses sequentially follow one another. This is also the case, for example, with a 32-bit bus, if the CPU communicates with the double-word addresses DW0–DW1–DW3–DW4–DW5 one after the other. Note that DW2 is not called up. PCI solves this problem elegantly, in that the bridge produces the burst DWO–DWl–DW2–DW3–DW4–DW5 and simply deactivates all $\overline{\text{BEx}}$ signals for DW2, in order to indicate that no data will be transferred.

For this purpose, extensive and intelligent read and write buffers are included in the bridge. Even the i486-50 with 100 Mbytes/s has, in single transfer mode, a bus bandwidth greatly exceeding the transfer rate of the PCI bus in single transfer mode. Thus, without PCI burst mode, the PCI bus would again be the hold up. The PCI bridge solves this problem, where possible, in that it joins together these single accesses to form a PCI burst. An example would be an access to the video RAM in order to open a window (write access) or to save the contents of the window (read access). Both tasks are performed conventionally (without a graphics processor) by the CPU, which performs multiple single transfers with sequential addresses. Single transfers are performed because the video RAM typically represents a non-cachable area of the memory address range and, thus, cannot be addressed by CPU burst cycles. This is reserved for cache line fills or write-backs (Pentium). The PC1 bridge can independently recognize that the single transfers have sequential addresses and can join these single transfers together to form a burst. In this way, the PCI bus bandwidth at 32 bits is increased to 133 Mbytes/s, thus PCI is no longer the bottleneck. Pentium single transfers are also fully served by such bursts because, during single transfers, only 32 bits of the 64-bit address bus are used. At 66 MHz, this gives a maximum Pentium bus bandwidth of 133 Mbytes/s. The maximum rate in 64-bit pipelined burst mode at 528 Mbytes/s clearly surpasses the PCI maximum rate (266 Mbytes/s at 64-bit), but was developed mainly for the reloading or writing back of cache lines in an L2-cache; the PCI bus was not taken into consideration.

In addition to the memory and 1/0 address areas already known from the 80x86 CPUs, PCI also includes a third area, namely the configuration address area. It is used for accessing the configuration register and the configuration memory of each PCI unit. A configuration memory of 256 bytes is provided for each unit.

The decoupling of processor memory subsystems and the PCI bus through the PCI bridge is strong enough to allow the bridge and CPU to operate in parallel, providing that the CPU is not currently addressing a PCI unit. In this way, for example, it is possible to exchange data between two PCI units via the PCI bridge, while the CPU is only addressing the applicable memory to perform a program.

24.2 Bus Cycles

The PCI bridge is considerably more intelligent than an ISA/EISA or MCA bus controller. It switches CPU accesses through to the addressed PCI unit, or «filters» such accesses to address the unit optimally (through the PCI bridge). PCI currently recognizes 12 types of access which are differentiated by the four command signals $C/\overline{BE3}$–$C/\overline{BE0}$ (the command and byte-enable signals are multiplexed). The transfer types are (the level of the command signals $C/\overline{BE3}$–$C/\overline{BE0}$ are shown in parentheses):

- INTA sequence (0000),
- special cycle (0001),
- I/O read access (0010),
- 1/0 write access (0011),
- memory read access (0110),
- memory write access (0111),
- configuration read access (1010),
- configuration write access (1011),
- memory multiple read access (1100),
- dual addressing cycle (1101),
- line memory read access (1110),
- memory write access with invalidation (1111).

Every transfer begins with an address phase, during which the address/data pins ADx transfer the address and the $C/\overline{BEx}$ pins transfer the instruction code. One or more data phases follow this phase, during which the same address/data pins ADx transfer data and the $C/\overline{BEx}$ pins transfer the byte-enable signals. In a bunt cycle, multiple data phases can follow a single address phase. In PCI terminology, the requesting PCI unit is known as the ***initiator*** (ISA/EISA/MCA: busmaster), the addressed PCI unit as the ***target*** (ISA/EISA/MCA: slave). Every transfer starts with the activation of the FRAME signal. The target indicates its readiness with an active $\overline{TRDY}$ signal (target ready). An active $\overline{TRDY}$ during a write access indicates that the target can take the data from the lines AD31–AD0 (32 bits) or A63–A0 (64 bits) respectively, in a read access, that the addressed data has been sent on the lines AD31–AD0 or A63–AO, respectively.

Thus, $\overline{\mathrm{TRDY}}$'s function corresponds to the $\overline{\mathrm{READY}}$ and $\overline{\mathrm{BREADY}}$ signals known from the 80x86 CPUs. In addition, the initiator must also indicate its readiness to the PCI bridge, namely through an active $\overline{\mathrm{IRDY}}$ signal (initiator ready). An active $\overline{\mathrm{IRDY}}$ during a write access indicates that the initiator has sent the write data on the lines AD31–AD0 or A63–A0, respectively. In a read access, $\overline{\mathrm{IRDY}}$ indicates that it takes the addressed data from the lines AD31–AD0 or A63–A0, respectively.

The PCI bridge can, in this way, almost function as a fast buffer between the initiator and the target, thus synchronizing the two PCI units. Only in this way is it possible for the bridge to convert CPU single accesses into a PCI burst. With the help of $\overline{\mathrm{IRDY}}$ and $\overline{\mathrm{TRDY}}$, the PCI bridge can also perform write posting and read prefetching. During write *posting*, the CPU first writes the data at a much higher speed (faster than the PCI bus can currently pass it on) to a buffer. The $\overline{\mathrm{READY}}$ signal is essential for this transfer in the 80x86. The posting buffer then transfers the data to the addressed PCI unit. For this, $\overline{\mathrm{IRDY}}$ and TRDY are necessary. If, for whatever reason, the PCI bus can pass on the data faster than the CPU can supply it, the PCI bridge deactivates the $\overline{\mathrm{IRDY}}$ signal to inform the target that the transfer has not yet finished, and that the bridge is waiting for the CPU. During read *prefetching*, the PCI bridge reads data from the target faster than necessary and stores it in a prefetch buffer; it then passes the data on to the CPU later. If the prefetch buffer is full because the data has not been collected quickly enough, then the bridge must deactivate the IRDY signal to inform the target that it should not deliver any more data. Naturally, $\overline{\mathrm{IRDY}}$ and TRDY can also be used to exchange data between two PCI units without affecting the CPU (for example, if data from a hard disk with a PCI adapter has to be written directly to the video RAM on a different PCI graphics card to recreate a window).

The initiator ends or interrupts the transfer by deactivating $\overline{\mathrm{FRAME}}$. The target can also stop the transfer by activation of the $\overline{\mathrm{STOP}}$ signal; this leads to a so-called target abort. I would briefly like to discuss the bus transfer types in the following section.

INTA sequence

During an INTA, an interrupt controller is implicitly addressed, that is, without explicitly sending out an I/O or memory address (the applicable controller recognizes the INTA sequence independently and automatically reacts to it). In the data phase, it transfers the interrupt vector through the ADx lines.

Special cycle

A special cycle can be used to transfer information to all PCI agents, for example, on the operating condition of the processor. The least significant ADx word AD15–AD0 delivers an information code corresponding to Table 24.1. The x86 specific codes correspond to the codes for the i486 and Pentium special cycles such as Flush, INTA, etc., and are sent out with the least ADx word AD31–AD16.

I/O read access

This access reads data from a PCI unit in the I/O address area. The addresses ADx define a byte address, that is, AD1 and AD0 must also be decoded because, in the PCI address area, both

AD15–AD0	Information
0000h	processor shutdown
0001h	processor halt
0002h	x86 specific code
0003h to ffffh	reserved

Table 24.1: Information codes for the PCI special cycle

8-bit and 16-bit ports could be available. The access usually consists of an address and a data phase.

I/O write access

This access transfers data to a PCI agent in the I/O address area. Here also, the addresses ADx define a byte address. The access most often consists of an address and a data phase.

Memory read access

The memory read access addresses a PCI unit in the memory address area and reads data from this agent. A read prefetch buffer is useful in supporting the read access and can increase its speed. The addresses ADx define a double word address, that is, AD1 and ADO need not be decoded. Instead, the byte-enable signals $\overline{\text{BEx}}$ indicate which groups of the 32-bit or 64-bit data bus contain valid values. The access usually encompasses an address and a data phase.

Memory write access

This access addresses a PCI agent in the memory address area and transfers data to it. Write posting usually increases the speed of the write operation considerably, because the initiator only has to wait for a return signal from the posting buffer and not from the target. As in the memory read access, the addresses ADx define a double word address, that is, AD1 and ADO need not be decoded, but the byte-enable signals $\overline{\text{BEx}}$ must. The access usually consists of an address and a data phase.

Configuration read access

PCI uses this access to address the configuration address area of the respective PCI unit. Here, IDSEL is activated to select the unit. Address bits AD7–AD2 indicate the addresses of the double words to be read in the configuration address area of the unit; AD1 and ADO are equal to 0. ADIO–AD8 are also used for selecting the addressed unit in a multifunction unit. All other ADxs have no significance. The access most often consists of an address and a data phase.

Configuration write access

This access is the counterpart to the configuration read access, the only differencebeing that the initiator writes data into the configuration address area of the target. The addressing is accomplished as in the configuration read access. The access usually encompasses an address and a data phase.

Memory multiple access

This access represents an extension to the line memory read access (and therefore, naturally, of the normal memory read access) and indicates to the target that the initiator (busmaster) wishes to read more than a complete cache line, or a correspondingly sized data block, from the memory without caching. The access is mainly beneficial for long sequential memory accesses, for example, if a large part of the video RAM corresponding to a window is to be saved. Thus, the access consists of one address phase and multiple data phases until FRAME is deactivated by the initiator.

Dual addressing cycle

This cycle is used to transfer a 64-bit address to a PCI unit, which only contains a 32-bit address/data path. Normally, the 64-bit address would be transferred through the 64-bit address/data path in a single clock cycle. However, if only a 32-bit address/data path is available, the 64-bit address must be split into a least significant 32-bit address (first addressing cycle) and a most significant 32-bit address (second addressing cycle). The dual addressing cycle serves for this purpose. In the first addressing cycle, the PCI bridge initiates a dual addressing cycle and transfers the least significant 32-bit address, in the second addressing cycle it initiates another cycle and transfers the most significant 32-bit address. The target then joins the 64-bit address back together. Thus, only in the second addressing cycle is it determined whether the access is a simple read access or an 1/0 access etc. Note that even PCI agents which support the 64-bit PCI data bus AD63–AD0 need not necessarily also handle 64-bit addresses (as an example, the Pentium contains a 64-bit data bus, but only has a 32-bit address area). On the other hand, a PCI unit may support 64-bit addressing, hut may only contain a 32-bit data bus (such as a memory mapped register).

Line memory read access

This access represents an extension to the normal memory read access, but does not contain as many data transfers as the memory multiple access. The instruction informs the target that the initiator wishes to read more than two 32-bit data blocks from the memory, typically to the end of a cache line. This access is also mainly beneficial for long sequential memory accesses. Thus, the access consists of one address phase and multiple data phases until FRAME is deactivated by the initiator.

Memory write access with invalidation

This access represents the counterpart to a line memory read access or a memory read access. Through a memory write access with invalidation, the initiator indicates that a minimum of one complete cache line should he transferred. In this way, the content of the main memory is then automatically more up to date than the content of the cache, so that the applicable line can also be invalidated; thus, a write-back is bypassed. The cache line size is defined in a configuration register of the master that sent out the cache line write-back. If a value of a subsequent cache line is to be written with this access, the complete cache line must be transferred, or a simple memory write access must be performed.

The PCI specification recommends the use of different read access modes to the memory in accordance with Table **24.2,** depending on whether or not a register for the cache line size is implemented in the busmaster.

With register for cache line size	
quantity of data to be transferred	mode
half cache line or less	memory read access
half to three cache lines	line memory read access
more than three cache lines	memory multiple access
Without register for cache line size	
quantity of data to be transferred	mode
two transfers or less	memory read access
three to twelve data transfers	line memory read access
more than twelve data transfers	memory multiple access

Table 24.2: Use of the PCI block access modes

For extensive sequential write accesses, the memory write access with invalidation should be used as far as possible. However, a register for the cache line size is required for this in the initiator.

Note that for memory accesses, address bits AD1 and A W are not decoded for the addressing. PCI always addresses double words; the active bytes within a double word are then given by $\overline{\text{BEx}}$. AD1 and **ADO,** however, are not without purpose; they are very important for setting the addressing sequence. For all burst accesses, PCI makes a distinction between linear and toggle incrementations of the addresses: linear incrementation (AD1 = A W = 0) means that all addresses follow one another sequentially; toggle incrementation (AD1 = 0, ADO = 1) means that the addresses follow in the style of a cache line fill, for instance they are interleaved as in the Pentium, and thus do not follow one another sequentially.

In Figure **24.3** you can see the flow of the most important signals for a typical PCI read access. Note that the first clock cycle after the address phase is used to switch the transfer direction of the multiplexed PCI address/data bus. During the address phase, the ADx pins deliver a value (the address), while during the data phase they receive a value (read data). As in the 8086/88, with its multiplexed address/data bus, a dummy cycle is required for the change in direction. Thus, the first value can be transferred in the third PCI clock cycle at the earliest. The PCI read transfer burst is, therefore, most economically performed (not wait states) as a 3–1–1–1– . .burst.

A typical PCI write access is performed in a similar way. It is also initiated by the activation of FRAME. In Figure **24.4,** the flow of the most important signals is shown. Here, contrary to a read access, the multiplexed PCI address/data bus need not be switched. The data phase can follow immediately after the address phase, without the need for a dummy cycle. Thus, the first value can be transferred in the second PCI clock cycle. The PCI read hansfer burst is most economically performed (no wait cycles) as a 2–1–1–1– . . . burst.

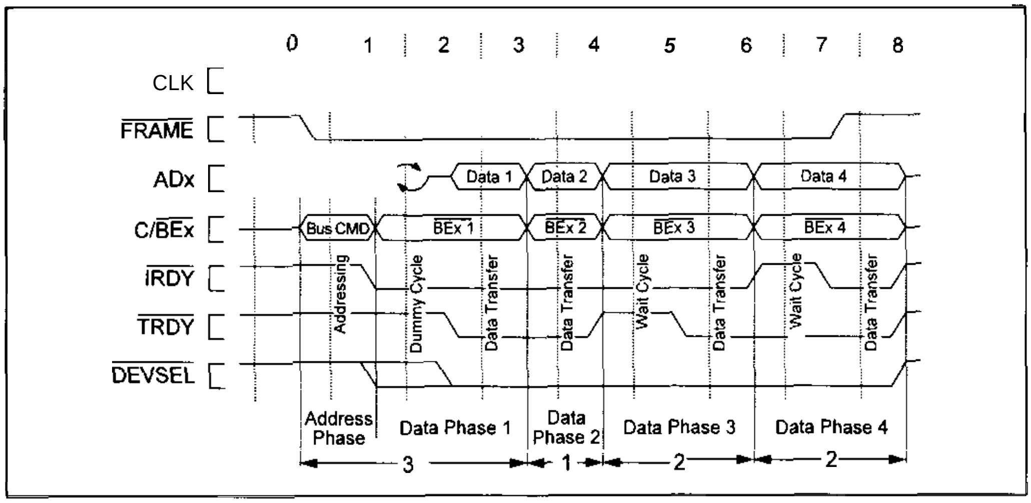


Figure 24.3: The PCI read transfer burst. Contrary to the optimum 3–1–1–1–... burst, a 3–1–2–2 burst is shown, whereby in the first instance an inactive $\overline{TRDY}$, *and in the second instance an inactive* $\overline{IRDY}$, *requests a wait state. The addressed PCI unit reacts with an active* $\overline{DEVSEL}$ *signal, if it has identified itself as the target of the PCI transfer. The transfer (3–1–2–2 burst) is ended by the deactivation of* $\overline{FRAME}$.

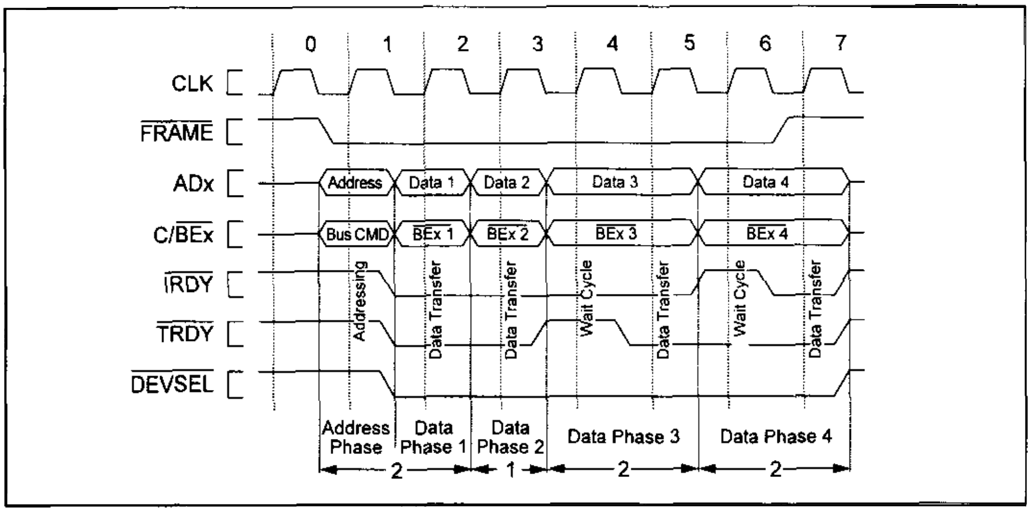


Figure 24.4: The PCI write transfer burst. Contrary to the optimum 2–1–1–1–.. burst. a 2–1–2–2 burst is shown, whereby in the first instance an inactive $\overline{TRDY}$, *and in the second instance an inactive* $\overline{IRDY}$, *requests a wait cycle. The addressed PCI unit reacts with an active* $\overline{DEVSEL}$ *signal, if it has identified itself as the target of the PCI transfer. The transfer (2–1–2–2 burst) is ended by the deactivation of* $\overline{FRAME}$.

24.3 Bus Arbitration

PCI bus arbitration is performed separately for each access, that is, a busmaster cannot hold up the PCI bus between two accesses. This may occur with EISA and MCA. For this reason, a PCI burst, which in the sense of bus arbitration represents a single access, can extend over any number of transfer cycles. However, this single arbitration does not impair the transfer bandwidth of the PCI bus because the arbitration is «hidden» behind the active bus; a hidden arbitra*tion* takes place. This means that the arbitration is already being performed if the active bus access is still running. In this way, no PCI bus cycles are required for the arbitration.

PCI includes two control signals for arbitration: $\overline{REQ}$ and $\overline{GNT}$. Each busmaster has its own request and grant signals, which are intercepted and used by a central arbitration logic. However, the PCI specification does not lay down a model for the arbitration; this is a task for the system designer. Usually, a model corresponding to the CACP from microchannel, or the bus arbitrators from EISA are used. The PC1 specification only requires that a PCI busmaster must activate the $\overline{REQ}$ signal to indicate a request for the PCI bus, and the arbitration logic must activate the $\overline{GNT}$ signal so that the requesting master can gain control of the bus. The busmaster must then begin a bus transfer within 16 CLK cycles, otherwise a time overrun error occurs.

24.4 DMA

If you look at the layout of the PCI slots (Section 24.9) and the specific PCI signals (Section 24.11), you can see that contrary to ISA/EISA and microchannel, no Direct Memory Access (DMA) is implemented. The «normal» DREQx and DACKx signals known from the PC are missing. At first, this may appear as a backward step because, above all in EISA and MCA PCs, the transfer of large quantities of data from peripheral devices to the main memory, and vice versa, can be performed very quickly via a DMA (even though PC manufacturers very rarely use this channel). The busmaster concept (not only in PCI, but also in EISA and microchannel) actually makes direct memory accesses superfluous. The DMA controller is usually located on the motherboard, but typically controls an 1/0 unit located on an adapter (such as the hard disk controller). In this way, the data transfer is triggered, for example, by the DREQ signal (single transfer mode). However, the necessary bus control signals (such as IOW, MEMR, etc.) are produced by the DMA controller (as busmaster) itself, that is, the CPU is not involved with the data transfer. Stated another way, you could regard the combined efforts of the (busmaster) DMA controller and of the trigger signal DREQ as a busmaster function of the adapter, where the busmaster chip is generally located on the motherboard where the bus control signals are produced, while the adapter uses this second busmaster in addition to the CPU through its DREQ signal. A single busmaster chip on the adapter itself makes this somewhat complicated procedure superfluous. The adapter busmaster can now produce all of the bus control signals itself and, thus, can address the 1/0 and memory address area in any number of ways. However, this requires bus arbitration by means of the expansion bus (or the slots), because with the DMA, only the arhitration between the CPU and the DMA controller is performed on the motherboard. Thus, the use of an external busmaster is more flexible and efficient than the DMA, but makes a more complicated arbitration necessary. It is, of course, possible that the external busmaster located on the adapter is a form of DMA controller which quickly exchanges data between the main memory and the adapter. The busmaster concept is, therefore, more generally understood than the Direct Memory Access, but DMA is only part of the busmaster concept.

24.5 Interrupts

Only optional interrupts are included in the PCI specification; they should be level-triggered and active-low. One interrupt line, $\overline{\text{INTA}}$, is assigned to each PCI unit. Only multifunctional units can also use the other three interrupt lines $\overline{\text{INTB}}$, $\overline{\text{INTC}}$ and $\overline{\text{INTD}}$. The PCI interrupts are formed in the CPU PCI bridge like the interrupt requests IRQx of the AT architecture. This usually occurs in a flexible way with help from the setup in the computer BIOS. Depending on which slot a PCI IDE hard disk adapter is installed in, the corresponding interrupt INTA for *this* slot must be set to the IRQ14 of the AT architecture. The INTA of a different slot (with, for example, a LAN adapter) can then be set to a different AT IRQx, for instance, IRQ11. Thus, for a single function PCI unit, only one interrupt (namely INTA) is available for each slot; whereas for a multifunction agent, one to four are available ($\overline{\text{INTA}}$ to $\overline{\text{INTD}}$). The actual IRQ activated by the PCI interrupt is set by software configuration. In this way, the inflexible setting of the IRQs, as in the AT architecture and also EISA/MCA, is no longer applicable. In addition, fewer contacts are needed – ISA/EISA/MCA slots contain 11 contacts without an increase in functionality. In a typical ISA/EISA/MCA adapter, ten contacts are not utilized; in a typical PCI adapter only three are not used (with which the adapter activates more than just an IRQ).

24.6 I/O Address Space

According to its specification, the PCI bus supports a 32-bit and also a 64-bit I/O address area. However, this only applies to true PCI units. You cannot reach an address greater than 64k with an 80x86 CPU because these processors can only produce 16-bit I/O addresses. The ports in a personal computer with a PCI bus are all located below 64k; the usage and address of each port have not changed when compared to ISA/EISA and MCA, depending on whether your PC is based on the AT, EISA or PS/2 architecture, or a standard expansion bus with either ISA/EISA or microchannel. ISA/EISA/MCA and PCI can, and should, be available alongside each other as PCI and the expansion bridge make this possible. You will find all of the valid I/O addresses and their corresponding usage in Chapters 21 (ISA), 22 (EISA) and **23** (MCA).

PCI stores two PCI registers in an I/O address area, which in EISA are reserved for the motherboard. These two 32-bit registers CONFIG–ADDRESS and CONFIG–DATA are used for accessing the configuration address area and are located at the addresses 0cf8h (ADDRESS) and 0cfch (DATA). If you wish to read or write a double word in the configuration area of a PCI unit, you must transfer the corresponding address to the CONFIG–ADDRESS register first. A write to the CONFIG–DATA register transfers the value to the specified location in the configuration address area of the addressed unit; a read transfers the value from this location. You can see the layout of the address registers in Figure 24.5.

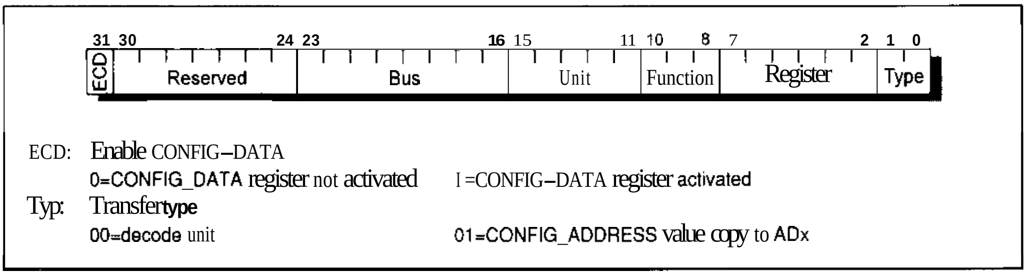


Figure **24.5:** *The CONFIG–ADDRESS register.*

If you set the ECD bit, the bridge carries out a configuration cycle for the 1/0 address Ocfch (corresponding to CONFIG–DATA). If the ECD bit is not set, a normal I/O cycle is performed which does not reach the configuration address area; it is switched through to the normal I/O address area. The *bus* entry stores the number of the PCI bus in a hierarchically structured system containing a maximum of 256 PCI buses. Such a hierarchically structured PCI bus system can be implemented by PCI–PCI bridges. *Unit* selects one of **32** possible PCI agents, *function* one of a maximum of eight functions within a PCI multifunction unit. Then, *register* addresses one of 64 possible dword entries in a specified configuration address area. Type is used to inform the bridge whether it is connected to the addressed unit (type = 0), and that after decoding of the unit and function entries it should send out a configuration cycle. If type equals Olb, then the unit is connected to the bridge and the bridge copies the contents of CONFIG–ADDRESS, unchanged, to the address/data bus. An access to the configuration address area through the two registers CONFIG–ADDRESS and CONFIG–DATA is known as *configuration mechanism #1.* In this way you can also instruct a PCI bridge to send out a special cycle in which the PCI bridge is specified in the address register for the bus entry, the entries for unit and function are written with a series of 1s and the register entry is written with a series of 0s. The next access to the data register then initiates a special cycle, where AD31–AD0 transfer the data register value.

In addition, there is also a *configuration mechanism* #2, but this is only provided for PC systems (as a processor-independent bus system, PCI can also be used in the Alpha or PowerPC, for example). Here, the PCI configuration area of a PCI unit is mapped into a 4k I/O address range between cOOOh and cfffh. This is achieved by using the activation register CSE (Configuration Space Enable) for the configuration area at the port address 0cf8h; you can see the structure in Figure 24.6, together with the layout of the forward register.

A value other than zero for key activates the mapping of the configuration area, that is, all 1/0 accesses with an address between cOOOh and cfffb initiate a configuration cycle. Otherwise, the I/O accesses to the 4k range between cOOOh and cfffb would be performed as normal I/O cycles. The *PCI bus* entry in the forward register at the port address Ocfah indicates the PCI bus for which the configuration cycle should be performed. A value of 00h means, for example, the bus immediately after the host bridge. Now, if you write a value to the 4k I/O range between cOOOh and cfffh, the address bits AD11–AD8 give the I/O address of the PCI unit, the *function* entry in the CSE register gives the function number within this PCI unit (if it represents a multifunction unit), and the address bits AD7–AD2 give the I/O address of the register index (or the dword register offset). The two address bits AD1 and AW of the I/O address are

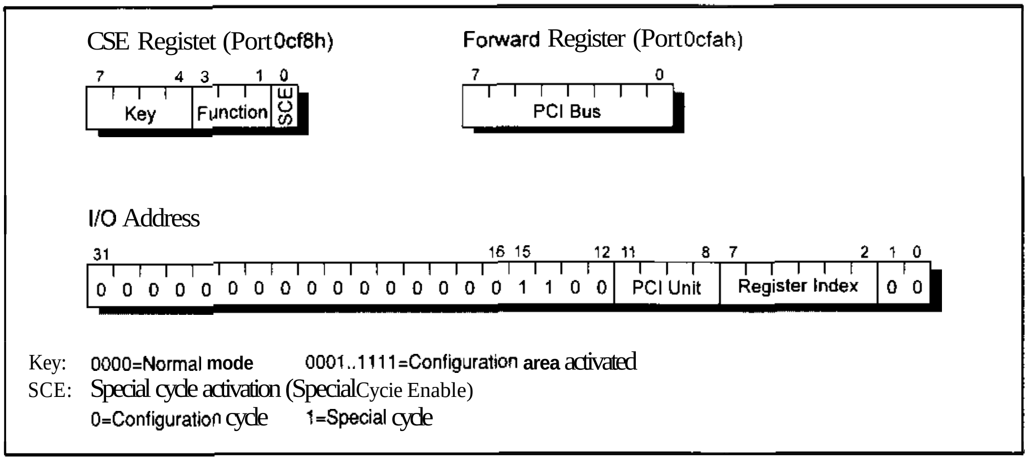

CSE Registet (Port 0cf8h)
7 4 3 1 0
Key
Function
SCE
Forward Register (Port 0cfah)
7 0
PCI Bus
I/O Address
31 16 15 12 11 8 7 2 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
PCI Unit
Register Index
0 0
Key: 0000=Normal mode 0001..1111=Configuration area activated
SCE: Special cycle activation (Special Cycie Enable)
0=Configuration cycle 1=Special cycle

Figure 24.6: Activation register, forward register **and** *I/O address* for *the configuration area*

ignored (it is better to set them to 00), address bits AD31–AD12 are equal to 0000ch or 0000 0000 0000 0000 1100.

In addition to these direct accesses to the configuration area of a PCI unit through the 1/0 ports, personal computers with a PCI bus, in general, also have an interface available through the BIOS interrupt 1ah. Its standardized functions are detailed in Section 24.9. First, however, we shall look at the layout of the configuration address area.

24.7 Configuration Address Space

A configuration area of 256 bytes is provided for every PCI unit (and every separate function in a multifunction unit), thus there are 64 ***registers*** for each 32 bits. A fixed ***header*** of 64 bytes is located at the start, which is predefined for each unit with this structure. The use of the remaining 192 bytes depends upon each individual PCI unit. The configuration software for such a unit then recognizes the use of this range. Figure 24.7 shows the layout of the configuration area and that of the header.

Authentic values for ***unit ID*** are between 0000h and fffeh; the value ffffh indicates that a PCI unit is not installed. In this way, the start routine of the PCI BIOS can identify all of the PCI units. The entries in the configuration area follow the little-endian format. The header itself is divided into two sections: the first 16 bytes (00h to 0fh) are the same for all PCI units; the layout of the subsequent 48 bytes can vary for different PCI units. These layouts are differentiated by the ***header*** entry (offset 0eh). Currently, only a single header type is defined (header = 00h), namely that you can see in the figure between offsets 10h and 3fh. The most significant header bit 7 indicates whether the unit is multifunctional (bit 7 = 1) or single function (bit 7 = 0). Note that the PCI specification only requires that the manufacturer ID, unit ID, command and status entries are available. The ***manufacturer ID*** is allocated by PCI SIG (the governing body that produced the PC1 standard). However, ***unit ID*** and ***revision*** are inserted by each manufacturer.

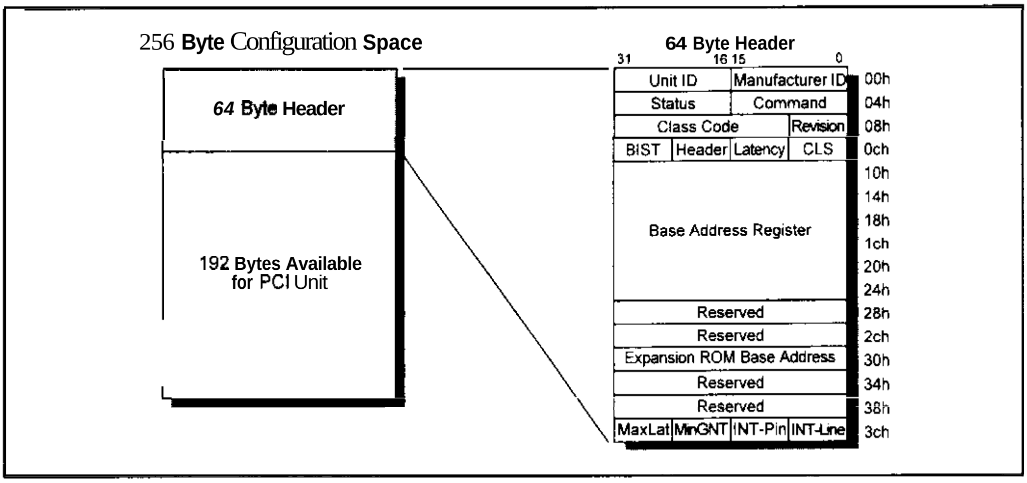


Figure *24.7:* **256-byte** *configuration area and* ***64-byfr header.***

The class code indicates the type of PCI unit. For this, the field is subdivided into three 1-byte sections. The most significant byte (at the offset Obh) indicates the basic class code, the middle byte (at the offset 0ah) indicates the subclass code, and the least significant byte (at the offset u09h) delivers a programming interface for the applicable unit. The authentic basic and subclass codes are shown in Table **24.3.**

For many previously defined basic and subclass codes, clearly defined programming interfaces already exist (such as VGA or SCSI), so that here no programming interface through the least significant byte (at the offset 09h) of the class code entry is required. For all other units, no such programming interface has been defined. For this reason, all detailed classes have the entry 00h in the programming interface field. The instruction entry in the header makes it possible to control the unit, that is, how it reacts to PCI cycles. Figure 24.8 shows the structure of this entry. If you write the value 0000h to the instruction register, you deactivate the applicable PCI unit; then, it will only react to a configuration cycle.

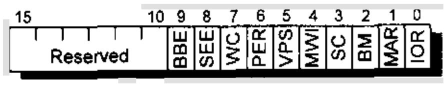
A set BBE bit enables so-called fast.back-to-back,, cycles for different targets. In this way, fewer dummy cycles are required between two PCU bus transactions. The SEE bit activates (SEE = 1) or deactivates (SEE = 0) the driver for the SERR signal. Should the system control also be informed of address parity errors, both SEE and PER must be set. The WC bit must be set for PCI units which perform address/data stepping. Address/data stepping means that not all necessary address/data signals are activated at one time. To reduce the load on the internal driver during activation of the lines, they are gradually set to a high level by means of blocks over a number of clock cycles. With this, the PCI units must wait a short time until all of the potentials have stabilized. If the PER bit is set, the PCI unit will react to a parity error; with a reset PER bit, all parity errors are ignored. By setting the VPS bit, you instruct a VGA compatible PCI unit to ignore all accesses to the palette register. If VPS = 0, the unit will react normally. A set MWI bit permits the unit to send out a memory write access with invalidation; if MWI is reset, it must use a normal memory write access instead. By resetting SC, you can instruct the unit to ignore all special cycles; if SC = 1, then the unit will also recognize special cycles, and will react

Basic code	Meaning	Subcode	Meaning
00h	Unit was produced before the class code definition was defined	00h	A previous units except VGA
		01h	VGA
01h	Controller for mass storage	00h	SCSI controller
		01h	IDE controller
		02h	Floppy controller
		03h	IPI controller
		80h	Other controller
02h	Network controller	00h	Ethernet
		01h	Token ring
		02h	FDDI
		80h	Other controller
03h	Video controller	00h	VGA
		01h	XGA
		80h	Other controller
04h	Multimedia unit	00h	Video
		01h	Audio
		80h	Other unit
05h	Memory controller	00h	RAM
		01h	Flash memory
		80h	Other controller
06h	Bridge	00h	Host
		01h	ISA
		02h	EISA
		03h	MCA
		04h	PCI-PCI
		05h	PCMCIA
		80h	Other bridge
07h–eh	Reserved		
ffh	Unit does not belong to any class 00h-feh	none	

Table 24.3: Basic ***and subclass codes***

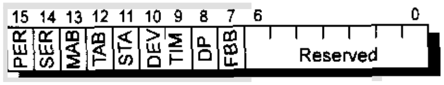
accordingly to them. If a unit should operate as a busmaster, you must set the *BM* bit. When *MAR* is set, the unit will react to an access of the memory address area, with a set *IOR* bit, it will react to I/O cycles. In addition to the instruction register, a status register is also provided; it indicates the status of the applicable unit for a PCI operation. The layout of the register is given in Figure 24.9.

If the unit discovers a parity error it sets the *PER* bit. The *SER* bit is set if the unit activates the $\overline{\text{SERR}}$ signal. If a unit operating as a busmaster stops a transaction, it must set the *MAB* bit. This is similar for *TAB*, except that here, the target has stopped the operation. MAB and TAB are set by the busmaster. STA, on the other hand, is set by a unit operating as the target if it has initiated a target abort. The two *DEVTIM* bits set the time characteristic for the $\overline{\text{DEVSEL}}$ signal. The *DP* bit is only implemented for the busmaster, and is set if PERR is activated, the unit is operating as a busmaster, and the PER bit is set in the instruction register. Finally, the *FBB* bit



BBE:	Fast back-to-back cycles (Back-to-Back Enable)	
	0=deactivate	1=activate
SEE:	$\overline{\text{SERR}}$ Enable	
	0=deactivate	1=activate
WC:	Wait cycle control	
	0=no address/data stepping	1=address/data stepping possible
PER:	Parity error (Parity Error Response)	
	0=ignore	1=perform
VPS:	VGA palette snoop	
	0=normal reaction	1=no reaction
MWI:	Memory write access with invalidation	
	0=deactivate	1=activate
SC:	Special cycle	
	0=ignore	1=include
BM:	Busmaster	
	0=no	1=yes
MAR:	Memory address area	
	0=deactivate	1=activate
IOR:	I/O address area	
	0=deactivate	1=activate

Figure 24.8: Instruction register.



PER:	Parity error			
	0=no parity error	1=parity error found		
SER:	System error			
	0=no system error	1=system error signalled		
MAB:	Master abort			
	0=no master abon	1=master abort received		
TAB:	Target abon			
	0=no target abon	1=target abort received		
STA:	Target abort			
	0=no target abort	1=target abon signalled		
DEVTIM:	DEVSEL timing			
	00=fast	01=medium	10=slow	11=reserved
DP:	Data parity error			
	0=no parity error	1=parity error found		
FBB:	Fast back-to-back cycles			
	0=not supported	1=supported		

Figure 24.9: Status register.

implemented for targets indicates whether the target supports fast back-to-back cycles (FBB = 1) or not (FBB = 0).

The header entry CLS (cache line size) defines the cache line size of the system in units of 32 bytes. It is necessary, for example, for the memory write access with invalidation instruction. The latency entry indicates how long a PCI bus operation can take: the effective time span amounts to the latency+8 PCI clock cycles. The most significant bit in the BIST (Built-In Self-Test) register indicates whether the unit can carry out a self-test (bit 7 = 1). If that is the case, you can issue BIST by writing the value 1 to bit 6. BIST then returns a termination code to the four least significant BIST bits 3–0. Any value other than zero indicates an error. The header entry INT-line in an AT PC indicates which hardware interrupt line IRQx is connected to the interrupt pin of the unit or function. The values 0–15, corresponding to IRQ0–IRQ15, are valid. The interrupt routing logic of the PCI bridge then activates the corresponding input of the PIC. Which interrupt pin the unit or function actually uses is defined by the INT pin: a value of 1 means $\overline{\text{INTA}}$, a value of 2 $\overline{\text{INTB}}$, etc. If the unit does not use interrupts, you must enter the value 0. The two read-only registers *MinGNT* and *MaxLat* give the minimum and maximum latency values required by the manufacturer of the PCI unit, so that the unit can optimally use the PCI bus. Adapter and PCI units frequently contain an I/O or memory area, which they use, for example, for storing data, program execution, etc. With the help of the base address *register(s)*, PCI now allows these I/O and memory areas to be arranged in any I/O or memory address area. In Figure 24.10, you can see the structure of the base address register for a 32/64-bit memory base and a 32-bit I/O hase. The memory address area can contain 32 or 64 bits, depending

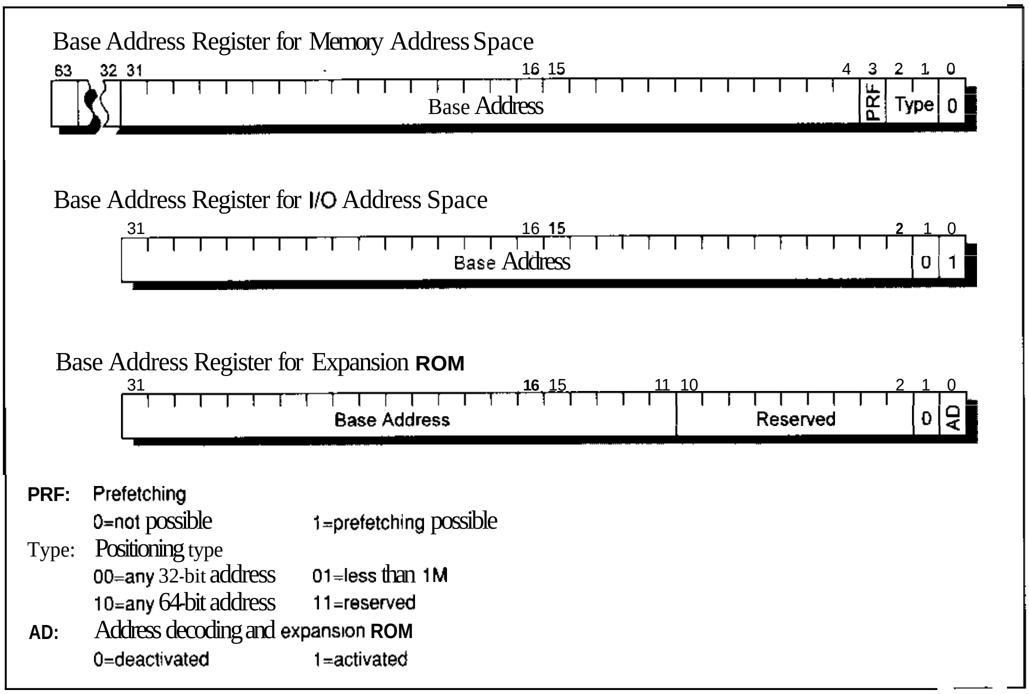


Figure 24.10: Base address *register* for memory and *I/O* ports, and also expansion ROM *base* address.

on the implementation; the I/O address area, however, always has only 32 bits (of which the 80x86 CPUs still only use the least significant address word ADIS–ADO).

Bit 0 differentiates a memory base address (bit 0 = 0) from an I/O base address (bit 0 = 1). Depending on the size of address, three to six base addresses are possible, the corresponding entry in the header contains 24 bytes. First I would like to explain the entry for a memory base. To be able to perform a remapping of an address area, the POST routine must naturally know how large the area to be formed should be. For this purpose, all bits in the applicable base register which, in practice, represent an offset within the address area to be formed, are hard-wired with the value 0. The POST routine can then simply determine the size of the area, whereby it writes the base address with a series of 1s and then reads back the base address value. All bits which return the value 0 are located within the address area to be formed; all bits set to 1 should be affected by the remapping. If, for example, bits 15 to 4 return the value 0 and the bits 31 (or 63) to 16 return the value 1, the area to be mapped encompasses 64 kbytes, because the 16 least significant bits can address 2^{16} bytes = 64 kbytes (the four least significant bits 3–0 cannot be changed, but are included for determining the size of the area). The remapping is performed, whereby the POST routine overwrites the address bits which returned the value 1 with the required base address. As you can see, the remapping is performed in blocks of 16 bytes, 32 bytes, 64 bytes, . . . , 1 kbyte, 2 kbytes, etc. As the base address entry in the header can include a number of base addresses, it also enables a fragmented remapping to be realized. If the *PRF* bit is set, the unit permits prefetching, that is, the bridge can read data in advance from the unit into a buffer without disturbing the unit or causing detrimental side-effects to occur. The 2-bit type entry indicates the address area where the mapping can be performed. The value 00b means that a 32-bit register can be mapped anywhere in the 32-bit address area; the value 10b has a similar meaning, only here the register and the address area each contain 64 bits. The value Olb is provided for 32-bit registers which must be formed in a 20-bit address below 1M.

The remapping for 1/0 addresses is achieved in a similar way, except that here only the two least significant bits 1 and 0 remain unchanged (bit 0 = 1 indicates that mapping in an I/O address area will be carried out). Thus, 1/0 address areas can be mapped in units of 2 bytes, 4 bytes, etc. A remapping of 2 bytes usually means that a single 32-bit 1/0 port will be placed in a suitable position in the I/O address area.

As the last entry in the header, you will also find the expansion ROM base address. With this, you can shift a ROM expansion to any position in a 32-bit memory address area. The remapping is achieved in exactly the same way as for a 32-bit base address, only here the 21 most significant bits are provided for the remapping. Thus, ROMs can occur and be remapped in units of 2 kbytes, 4 kbytes, etc. By setting AD, you activate the expansion ROM, that is, addresses are decoded; the value zero deactivates the ROM expansion. In this way, a PCI unit can operate with or without a ROM expansion selected. Only if AD is set does the remapping address in bits 32–11 have any significance.

24.8 PCI-specific BIOS Routines

You can comfortably access the configuration address area with ten functions of the BIOS interrupt 1ah. They are listed in the following section along with brief explanations.

- **PCI BIOS** available?

The function determines whether or not a PCI BIOS is available.

Register	Call value	Return value
AH	b1h	00h=available, 01h–ffh=not available[1]
AL	01h	Configuration mechanism, special cycles[2]
BH		Version (main level)
BL		Version (sub level)
CL		Number of the last PCI bus in the system
EDX		String "PCI" ('P' in DL, 'C' in DH etc.)
Carry		Status: 1=no PCI BIOS[1]

[1] P C BIOS only available, if EDX also equals "PCI"

[2] Bit 0=1: configuration mechanism #1, bit 1=1: configuration mechanism #2, bit 4=1 special cycle through configuration mechanism #1, bit 5=1: special cycle through configuration mechanism #2

- **PCI unit** search

The function determines, by means of the unit and manufacturer identification, and also an index N, the PCI address of the unit, that is, bus number, unit number and function number of the Nth unit which fulfils the search criteria.

Register	Call value	Return value
AH	b1h	Result code in accordance with Table 24.4
AL	02h	
BH		Bus number
BL		Bits 7.3: unit, bits 2.0 function
CX	Unit	
DX	Manufacturer	
SI	Index	
Carry		1=error, 0=o.k.

- **PCI** class code search

'The function determines, by means of a specific class code, the PCI address of the Nth unit which contains this class code.

Register	Call value	Return value
AH	b1h	Result code in accordance with Table 24 4
AL	03h	
BH		Bus number
BL		Bits 7 3 unit, bits 2 0 function
ECX	Class code	
SI	Index	
Carry		1=error 0=o k

- Initiate special cycle

The function sends out a special cycle to a specific PCI bus in the system. The 32-bit value in EDX is sent on the address/data bus ADx during the data phase.

Register	Call value	Return value
AH	b1h	Result code in accordance with Table 24.4
AL	06h	
BH	Bus number	
EDX	Data	
Carry		1=error, 0=o.k.

- Configuration byte/word/double word read

The three functions read a byte (8 bits), word (16 bits) or double word (32 bits) from the configuration area of a PCI unit.

Register	Call value	Return value
AH	b1h	Result code in accordance with Table 24.4
AL	08h (byte), 09h (word), Oah (dword)	
BH	Bus number	
BL	Bits 7.3: unit, bits 2..0: function	
CL/CX/ECX	Read byte, word, dword	
DI	Register number	
Carry	1=error, 0=o.k.	

- Configuration byte/word/double word write

The three functions write a byte (8 bits), word (16 bits) or double word (32 bits) into the configuration area of a PCI unit.

Register	Call value	Return value
AH	b1h	Result code in accordance with Table 24.4
AL	Obh (byte), Och (word). Odh (dword)	
BH	Bus number	
BL	Bits 7.3: unit, bits 2..0: function	
CL/CX/ECX	Write byte/word/dword	
DI	Register number	
Carry		1=error, 0=o.k.

All functions return a result code in register AH. The five possible PCI BIOS result codes are listed in Table 24.4.

Code	Meaning
00h	Successful completion
81h	Function not supported
83h	Invalid manufacturer identification
86h	Unit not found
87h	Invalid register number

Table 24.4: PCI BIOS result codes

24.9 PCI Slots

PCI frees itself not only from ISA – as microchannel did a few years previously – but with its slot geometry and contact layout, it is entirely independent from all existing bus systems. Only the dimensions of the slots bring microchannel to mind. PCI is laid out for an address and data bus width of 32 bits without compromise; you are wasting your time looking for 8-bit and 16-bit segments. Instead, an expansion increase to 64 bits is planned. You can see the construction and signal layout of a PCI slot in Figure 24.11. The multiplexing scheme for the data and address bits can clearly be seen. In total, a maximum of three PCI slots can be available. However, as each slot can be filled by a PCI adapter containing a number of functional units and, in addition, PCI units can also be included on the motherboard, the quantity of possible PCI devices is not restricted to three. With a PCI clock speed of 33 MHz and the high current driving capabilities which adapters request compared to on-board units, more than three external adapters would overload the PCI components, or would become too error-prone.

The 32-bit section contains 124 contacts, of which, however, only 120 are actually used. The remaining four contacts are blocked by a code bridge. Owing to the different slots, PCI takes into account the technical developments initiated for the 3.3 V technology used in power saving notebooks or green PCs. In this way, the power consumption in a CMOS circuit can be halved. This is because the power consumption is proportional to the square of the supply voltage. Some highly integrated circuits go one stage further, and can no longer even tolerate the normal 5 V supply, because their internal structure (above all, the channel width of the MOSFETs) would burn. A further PCI demand with respect to lower power consumption is that every PCI unit must operate with a clock between 0 MHz and 33 MHz. In this way, for example, the clock signal can be switched off 10 MHz) to reduce the power consumption by more than 999%.

The two present contacts $\overline{\text{PRSNT1}}$ and $\overline{\text{PRSNT2}}$ are individually, or jointly, connected to ground or left open by an installed PCI adapter, in order to indicate that an adapter is installed and what its power consumption is. If no adapter is installed, then both contacts naturally remain open. The possible configurations are shown in Table 24.5.

The pins identified as *I/O* represent special supply pins for a universal adapter, which can be operated with both 5 V and 3.3 V. Such adapters can be inserted into any slot. Note that in

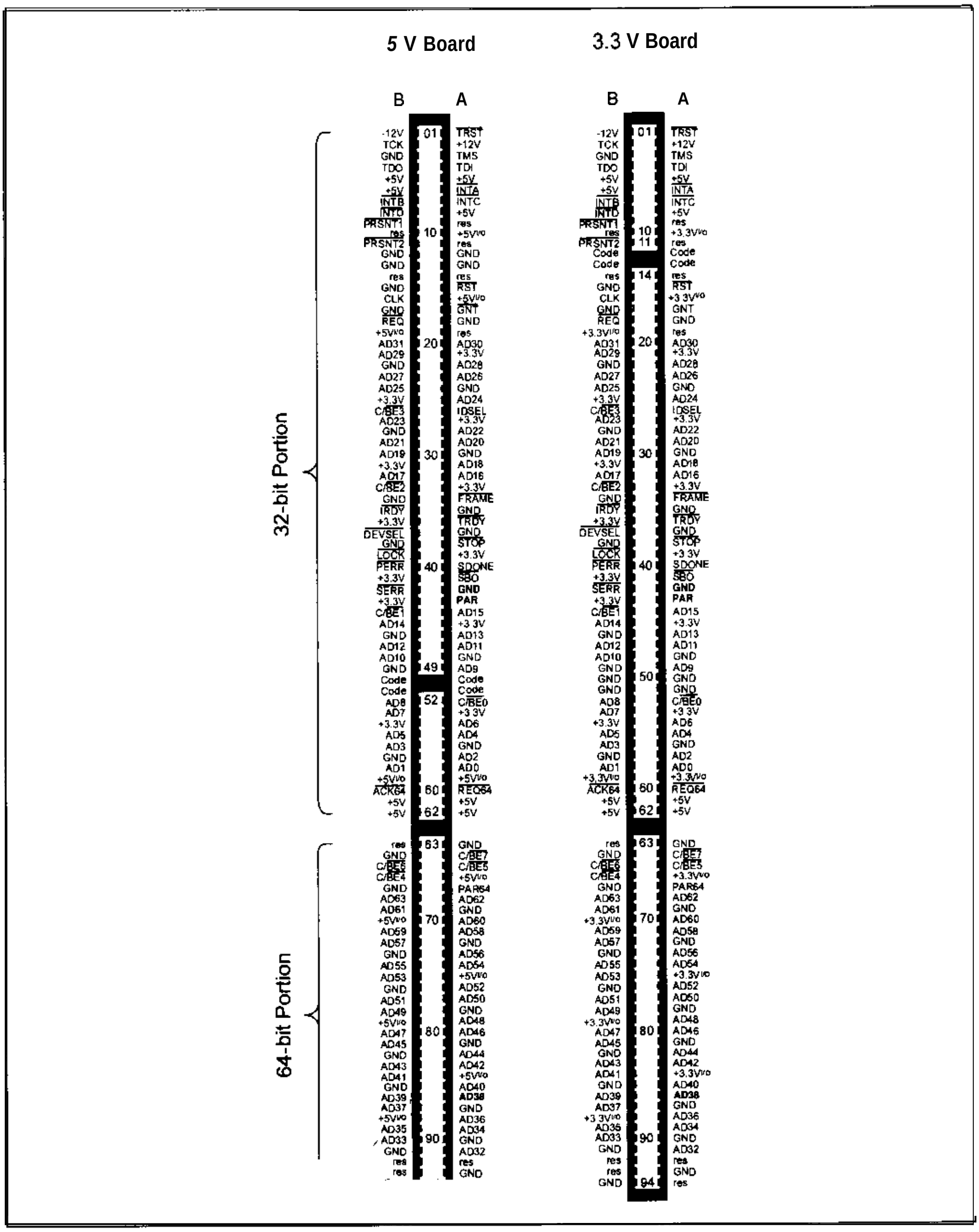


Figure 24.11: The PCI slots. PCI defines two different slot geometries for 5 V, 3.3 V and universal adapter (for 3.3 V and 5 V). They are differentiated by different code bridges so that an adapter cannot be incorrectly installed. In addition, a 64-bit segment is provided.

$\overline{\text{PRSNT1}}$	$\overline{\text{PRSNT2}}$	Meaning
open	open	no adapter installed
GND	open	adapter with max 25 W power consumption
open	GND	adapter with max 15 W power consumption
GND	GND	adapter with max 7 5 W power consumption

Table 24.5: Present contacts

addition to the special +5 $V^{I/O}$ and +3.3 $V^{I/O}$ pins, the usual +5 V (5 V slot) and normal 3.3 V pins (3.3V slot) also exist.

The 64-bit section is separated from the 32-bit part by a coding bridge. It is optional and stores the most significant double word of a quad word value or a quad word address. Note that the control and status pins $\overline{\text{REQ64}}$ and $\overline{\text{ACK64}}$ for the activation of the 64-bit extension are included in the 32-bit part. Due to the fact that the contacts are close together, the 64-bit PCI slot is very compact, despite its total of 188 contacts. The address/data multiplexing reduces the quantity of contacts, even with the 64-bit slot, to less than that in a microchannel slot (maximum of 202 pins).

In addition to the 64-bit extension, a few signals in the standard 32-bit slot are also optional. It is not absolutely necessary for them to be implemented in order to comply with the PCI specification. The required and the optional pins and signals are listed in Table 24.6.

Required:
AD31–AD0, C/$\overline{\text{BE3}}$–C/$\overline{\text{BE0}}$, PAR, $\overline{\text{FRAME}}$, $\overline{\text{TRDY}}$, $\overline{\text{IRDY}}$, $\overline{\text{STOP}}$, $\overline{\text{DEVSEL}}$, IDSEL,
$\overline{\text{PERR}}$, $\overline{\text{SERR}}$. $\overline{\text{REQ}}$. $\overline{\text{GNT}}$, CLK, $\overline{\text{RST}}$

Optional:
AD63–AD32, C/$\overline{\text{BE7}}$–C/$\overline{\text{BE4}}$, PAR64, $\overline{\text{REQ64}}$, $\overline{\text{ACK64}}$, $\overline{\text{LOCK}}$, $\overline{\text{INTA}}$, $\overline{\text{INTB}}$, $\overline{\text{INTC}}$,
INTD, SBO, SDONE, TDI, TDO, TCK, TMS, $\overline{\text{TRST}}$

Table 24.6: Required and optional PCI pins and signals

With a so-called shared slot you will find an ISA, EISA or microchannel slot along the PCI slot. A true ISA/EISA/MCA adapter would normally be inserted in the ISA/EISA/MCA part of a shared slot, or a true PCI adapter would be inserted in the PCI part of the same slot. These shared slots should allow the number of slots to be reduced to a minimum and, in this way, permit ISA/EISA/MCA adapters to be used without increasing the space required (which is necessary for building very compact notebooks or portables). Thus, PCI has learnt from IBM's experiences with microchannel.

24.10 PCI Adapters

PCI includes three different types of boards, namely 5 V, 3.3 V and the universal board. The 5 V boards have a recess at the pin positions A50/B50 and A51/B51, thus they only fit into 5 V slots. In a similar way, the 3.3 V adapter has a recess at the pin positions A12/B12 and

A13/B13. As the universal board can be installed in both 5 V and 3.3 V slots, two recesses are provided, namely at A50/B50, A51/B51 (5 V coding) and A12/B12, A13/B13 (3.3 V coding).

PCI adapters which do not support the JTAG boundary scan test must short-circuit their T W and TDI pins. Frequently, a single scan path is implemented across the complete motherboard and the PCI slots. This makes checking simpler because only a single input and a single output are necessary. If TDO and TDI are not connected together in an adapter without the JTAG test, the scan path is interrupted.

24.11 PCI Signals

Here, I would like to list the PCI contacts and explain the meaning of their corresponding signals. As PCI supports busmasters located on PCI adapters in the same way as EISA and microchannel adapters do, all pins are bidirectional. To make the data and signal flow clearer, I have assumed that the CPU (or another unit on the motherboard) represents the current busmaster for the given transfer direction. The pins are divided into their corresponding 32-bit and 64-bit groups, and are listed alphabetically.

24.11.1 Standard 32-bit Section

$\overline{\text{ACK64}}$ (I/O)
Pin 860

An active acknowledge 64-bit transfer signal with a low level indicates that the PCI unit has identified itself as the target for the current bus cycle, and that it can perform the requested 64-bit transfer.

AD31–ADO (I/O)
Pins A20, A22–A23, A25, A28–A29, A31–A32, A44, A46–A47, A49, A54–A55, A57–A58, B20–B21, B23–B24, 827, B29–B30, 832, B45, B47–B48, 852–853,855–856, 858

The 32 address and data pins form the multiplexed PCI address and data bus. Each PCI bus operation consists of an addressing phase, during which the pins AD31–AD0 transfer an address, and one or more data phases where data is hansferred.

C/$\overline{\text{BE3}}$–C/$\overline{\text{BE0}}$ (I/O)
Pins A52, B26, 833, B44

The bus command and byte-enable signals are hansferred on these four pins. Like AD31–AD0, they form a multiplexed bus. During the addressing phase, the signals C/$\overline{\text{BE3}}$–C/$\overline{\text{BE0}}$ indicate the type of bus cycle. With this, the possible combinations of C/$\overline{\text{BE3}}$–C/$\overline{\text{BE0}}$ have the following meaning:

```
(0000) INTA sequence         (1000) Reserved
(0001) Special cycle         (1001) Reserved
(0010) I/O read access       (1010) Configuration read access
```

```
(0011) I/O write access    (1011) Configuration write access
(0100) Reserved            (1100) Memory multiple access
(0101) Reserved            (1101) Dual addressing cycle
(0110) Memory read access  (1110) Line memory read access
(0111) Memory write access (1111) Memory write access with invalidation
```

During the data phase, the byte-enable signals $\overline{\text{BE3}}$–$\overline{\text{BE0}}$ define which bytes of the 32-bit data bus transfer authentic values. Note that between the valid bytes «gaps» can occur (for example, the combination $\overline{\text{BE3}}$ = 0, $\overline{\text{BE2}}$ = 1, $\overline{\text{BE1}}$ = $\overline{\text{BE0}}$ = 0 is valid).

CLK (O)
Pin B16

The pin gives the PCI clock signal for all PCI operations. In accordance with the specification, it has a frequency of between 0 MHz and 33 MHz.

$\overline{\text{DEVEL}}$ **(I/O)**
Pin 837

An active device select signal with a low level indicates that the decoding logic has identified the applicable PCI unit as the target of a bus operation.

FRAME **(I/O)**
Pin A34

The $\overline{\text{FRAME}}$ signal is sent by the active PCI master. The reduction of $\overline{\text{FRAME}}$ to a low, that is, active, level starts the addressing phase. The master deactivates FRAME again, to indicate the last data phase of a bus operation.

$\overline{\text{GNT}}$ (O)
Pin A17

An active grant signal with a low level indicates to the corresponding unit that the arbitration logic has handed over the bus, and that it can now use the PCI bus as the master. Each master has its own $\overline{\text{GNT}}$ input.

IDSEL **(O)**
Pin A26

The initialization device select signal is used as a chip select signal during the accessing of the configuration address area, to determine the unit to be configured and to address it.

INTA, $\overline{\text{INTB}}$, $\overline{\text{INTC}}$, $\overline{\text{INTD}}$ (I, I, 1, I)
Pins A6–A7, 87–88

Every PCI functional unit can issue a maximum of four hardware interrupts by means of a low level signal at these pins. A single function unit can only activate INTA, a multifunction unit can also activate $\overline{\text{INTB}}$ to $\overline{\text{INTD}}$, depending on the requirements and layout. Hardware interrupts are level-higgered with an active low level. For a PC, INTA to INTD of the respective units are associated with the hardware interrupts IRQ0 to IRQ15 on compatibility grounds.

$\overline{\text{IRDY}}$ **(I/O)**
Pin B35

The initiator ready signal indicates that the initiator (busmaster) is ready, and that it can complete the current data phase. For a write access, the initiator activates the $\overline{\text{IRDY}}$ signal, to show that the data on the bus is valid. For a read access, IRDY indicates that the initiator can take the read data. Thus, $\overline{\text{IRDY}}$ corresponds to the processor READY signal, except that it is sent out by the initiator (master) and not the target (slave).Only if $\overline{\text{IRDY}}$ and TRDY are active at the same time is the data phase actually completed.

LOCK **(I/O)**
Pin B39

An active $\overline{\text{LOCK}}$ signal with a low level defines an atomic access which extends over a number of bus operations. Accesses to non-locked elements can still be performed, that is, $\overline{\text{LOCK}}$ only blocks accesses to the PCI element and does not completely block the PCI bus for other accesses. The $\overline{\text{LOCK}}$ signal of the 80x86 CPUs, on the other hand, blocks the whole local CPU bus, and thus the transfer to another master.

PAR **(I/O)**
Pin A43

The parity pin transfers a parity bit so that even parity is achieved across AD31–AD0 and C/$\overline{\text{BE3}}$–C/$\overline{\text{BE0}}$, that is, the quantity of all 1s in AD31–AD0, C/$\overline{\text{BE3}}$–C/$\overline{\text{BE0}}$ and PAR is even.

$\overline{\text{PERR}}$ **(I/O)**
Pin B40

For all PCI operations, with the exception of special cycles, an active parity error signal with a low level indicates that a data parity error has occurred.

$\overline{\text{PRSNT1}}$, $\overline{\text{PRSNT2}}$
Pins B9, B11

The two present contacts $\overline{\text{PRSNT1}}$ and $\overline{\text{PRSNT2}}$ are individually, or jointly, set to ground or left open by an installed PCI adapter to indicate that an adapter is installed and what its power consumption is (Table 24.5).

$\overline{\text{REQ}}$ (I)
Pin B18

An active request signal with a low level indicates to the arbitration logic that the applicable unit wishes to use the bus as the master. Each master has its own $\overline{\text{REQ}}$ output.

$\overline{\text{REQ64}}$ **(I/O)**
Pin A60

An active request **64-bit** transfer signal with a low level indicates that the current busmaster wishes to perform a **64-bit** transfer.

$\overline{\text{RST}}$ (O)
Pin A15

An active reset signal with a low level resets all connected PCI units

$\overline{\text{SBO}}$ (I/O)
Pin A41

An active snoop backoff signal with a low level at this pin indicates an inquiry hit to a modified cache line. $\overline{\text{SBO}}$, together with SDONE, supports a write-through or write-back cache which is located in the area of the PCI bridge and lies within the CPU address area.

SDONE (I/O)
Pin A40

An active snoop done signal with a high level shows that the current inquiry cycle has been completed.

$\overline{\text{SERR}}$ (I/O)
Pin B42

For all PCI operations, an active system error signal with a low level indicates an address parity error for special cycles, or another serious system error.

STOP (I/O)
Pin A38

By an active $\overline{\text{STOP}}$ signal with a low level, the target of a master (initiator) indicates that the master should stop the current operation (target-abort).

TCK, TDI, TDO, TMS, $\overline{\text{TRST}}$ (I, 0 , 0 , O, O)
Pins A1, A3, A4, B2, B4

These five pins form the interface for the JTAG boundary scan test in accordance with IEEE 1149.1. Through TDI (Test Data Input), test data or test instructions are input and through TDO (Test Data Output), output, synchronous to TCLK (test clock). An active TMS (Test Mode Select) signal activates the TAP control, an active $\overline{\text{TRST}}$ (test reset) rests them.

TRDY (I/O)
Pin A36

The target ready signal indicates that the addressed PCI unit (the target) is ready and that the current data phase can be completed. During a write access, the target activates TRDY, to indicate that it can take the write data. For a read access, TRDY shows that the target now has the read data ready. Thus, $\overline{\text{TRDY}}$ corresponds to the processor $\overline{\text{READY}}$ signal. Only if $\overline{\text{TRDY}}$ and $\overline{\text{TRDY}}$ are active at the same time is the data phase actually completed.

64-bit Expansion

AD63–AD32 **(I/O)**
Pins A68, A70–A71, A73–A74, A76–A77, A79–A80, A82–A83, A85–A86, A88–A89, A91, B68–B69, B71–B72, B74–B75, B77–878, B80–B81, B83–B84, B86–B87, B89–B90

The 32 address and data pins form the expansion of the multiplexed PCI address and data bus to 64 bits. Every PCI bus operation consists of an addressing phase, during which the pins AD63–AD32 transfer an address, providing $\overline{\text{REQ64}}$ and a DAC command are active; otherwise, AD63–AD32 are reserved. During the data phase(s) the most significant double word of a 64-bit quad word are transferred through AD63–AD32, if $\overline{\text{REQ64}}$ and ACK64 are both active.

C/$\overline{\text{BE7}}$–C/$\overline{\text{BE4}}$ (I/O)
Pins A64–A65, B65–B66

The bus command and byte-enable signals are similar to C/$\overline{\text{BE3}}$–C/$\overline{\text{BE0}}$ and are transferred through these four pins. Thus, they form a multiplexed bus. During the addressing phase, signals C/$\overline{\text{BE7}}$–C/$\overline{\text{BE4}}$ indicate the bus type in a similar way to C/$\overline{\text{BE3}}$–C/$\overline{\text{BE0}}$ (refer to the appropriate section). In the data phase, the byte-enable signals $\overline{\text{BE7}}$–$\overline{\text{BE4}}$, together with $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, define which bytes of the 64-bit data bus transfer authentic values.

PAR64 **(I/O)**
Fin A67

The parity pin transfers a parity bit so that even parity is achieved through AD63–AD32 and C/$\overline{\text{BE7}}$–C/$\overline{\text{BE4}}$.

24.12 Excursion: A Modern PCI Chipset – 82430FX

The 82430FX for the Pentium is a typical PCI chipset. It comprises the PCI system controller 82437FX (TSC) with integrated PCI, L2-cache and DRAM controller, the data path chip 82438FX (TDP) and the PCI–ISA–IDE bridge 82371FB (PIIX = PCI ISA IDE Xcelerator) with ISA controller and EIDE interface. The following briefly discusses the various components.

24.12.1 The Components and Their Cooperation

To form a PCI system with the 82430FX chipset you need a system controller 82437FX, two data path chips 82438FX (because one chip only has a 32-bit data path) and – of course – a processor. The 82430FX is especially tuned for the Pentinm, therefore I will use the Pentium as the example processor. Typical PCI boards usually have several ISA slots in addition to the PCI contacts. The required PCI–ISA bridge is implemented by the PIIX chip 82371FB. There are also PCI–EISA bridges available, but I will restrict myself to ISA because the PCI/ISA combination is the most widely used one. Figure 24.12 shows a typical PCI/ISA system configuration with the 82430FX chipset.