

Doing PCI Right

The Advantages of MITE™-Based PCI Data Acquisition

by

Tim Hayles, Sr. Software Group Manager, Data Acquisition Products
Ed McConnell, Data Acquisition Product Manager
Arthur Ryan, Senior Engineer

Preliminary Edition – as of August 14, 1997



Part Number 341312B-01

*Copyright 1997 National Instruments Corporation. All rights reserved.
Product and company names listed are trademarks or tradenames of their respective companies*

Doing PCI Right

The Advantages of MITE™-Based PCI Data Acquisition

Table of Contents

Introduction	1
A Closer Look at PCI	1
Hardware Standard	2
Architecture	2
What Does PCI Bring to Data Acquisition?	2
Benefits Overview	2
System Architecture Implications	3
Clock Speed, Transfer Rates, and Bus Size	3
Plug and Play	3
Memory Mapped vs. I/O Addressing	3
Interrupts	4
How Does Bus Mastering Improve System Performance?	4
Bus Mastering – A Common Example	4
Bus Mastering – A Technical Explanation	5
DMA Controller Operation	5
Which System Level Problems Are Not Solved with PCI?	6
Interrupt Chaining and Latency	6
Virtual Memory and Non-Contiguous Buffers	7
Data Size and Free Memory Dependencies	7
Reporting the Progress of an Active Data Acquisition Process	8
How Should PCI Devices Be Designed for Use in Data Acquisition Systems?	8
Interrupts	8
Implementing Bus Mastering	8
Restricting Memory	9
Scatter-Gather	9
Reprogramming the DMA controller	9
On-the-fly Scatter-Gather Bus Mastering	10
Cooperation Between Driver Software and Bus Master DMA Controller	11
System Performance Tests	11
Simple Data Transfer Test	12
Stream to Disk Test	13
Conclusion	14
Key Points to Remember When You Select a PCI-based DAQ Board	14
Resources	14

Doing PCI Right

The Advantages of MITE™-Based PCI Data Acquisition

by
Tim Hayles, Sr. Software Group Manager, Data Acquisition Products
Ed McConnell, Data Acquisition Product Manager
Arthur Ryan, Senior Engineer

Introduction

Peripheral component interconnect (PCI) bus is the latest, most advanced computer technology to take the computer industry by storm. It promises enormous improvements in data transfer speed and overall productivity. But how do you know that your data acquisition and instrumentation systems are benefiting from all the advantages of PCI? Although PCI promises lightning speed transfer rates, designers of PCI-based data acquisition (DAQ) and instrumentation devices must take special care to consider the affects of their design on the overall instrumentation system performance. PCI is simply a hardware standard, not an overall system standard. In fact, there are right ways and not so good ways to design PCI plug-in boards. This white paper takes a closer look at PCI with the intention of showing how PCI data acquisition and instrumentation boards should be designed so that the best overall system performance can be achieved. It will answer the following questions:

1. What is PCI?
2. What does PCI bring to data acquisition systems?
3. How does bus mastering improve system performance?
4. What system level problems are not solved by PCI, and what can be done to overcome them?
5. What is necessary for PCI to be used correctly in data acquisition systems?

If you are a designer or user of PCI-based DAQ products, you will benefit from this information.

An Intel report details a number of guidelines for efficient PCI device design because the “PCI card choice is of paramount importance in determining system performance. Much of the observed performance difference can be attributed to differences in PCI implementation. There are a number of rules to follow when making an efficient, high performance, PCI design While current systems may provide good performance even when the rules are not followed, future systems may not.”¹ How do you know if the PCI data acquisition board you are using will deliver good system performance – both now and in the future?

A Closer Look at PCI

Developed by Intel in 1992, PCI provides an industry-standard peripheral expansion architecture for computer bus design. In 1995, the PCI bus found its way into most computers because standard peripheral interface connections, such as the ISA bus, lacked the performance and capability of the microprocessors with which they were integrated. By tapping into the microprocessor local bus, designs capitalize on the latest microprocessor technology to achieve a higher level of performance.

PCI, the missing link between processors and memory, helps correct an imbalance in overall system throughput. Processor performance has dramatically increased in recent years; state-of-the-art microprocessors have 100 times more processing power than older microprocessors like the 80286. To keep up with the faster processors, system memory is also faster. DRAM access times have steadily decreased from 100 ns to 60 ns over recent years, and new DRAM architecture contributes to improve efficiency. All this microprocessor and memory evolution accelerates the performance between the microprocessor and system memory. However, the communication between peripheral devices, the microprocessor, and memory has lagged significantly. PCI mitigates the

¹ “Efficient Use of PCI,” Platform Architecture Labs, Intel Corporation, 04/22/97, http://support.intel.com/oem_developer/chipsets/pci/general/pci001.htm.

communication bottleneck as it is a processor independent, high-speed data transfer bus. Computer designers benefit as they are no longer constrained by a specific processor architecture, but rather they can build powerful microprocessor independent systems with the PCI bus serving as the data transfer highway. Hence, diverse systems, such as Pentium PCs, Alpha workstations, and Power Macintosh computers, utilize the PCI bus.

Hardware Standard

The unique nature of the PCI standard lies in the fact that it is a chip and board level standard. As a result, peripheral vendors can develop a PCI compliant ASIC for use on a PCI peripheral card or as an essential component in computer systems. The rapid development of PCI-based systems is accomplished by integrating different PCI components onto a single board. This chip and board level standard reduces cost because peripheral vendors develop one ASIC for different computer systems, and computer system vendors leverage off the development efforts of peripheral vendors.

Architecture

Figure 1 depicts the system architecture of most PCI-based computers. The CPU has a direct, high-speed path to system memory which facilitates rapid execution and data manipulation. A CPU to PCI interface, commonly referred to as a “bridge” links the CPU to the PCI bus, which is populated with peripheral devices requiring high bandwidth. Another bridge interface links the PCI bus to a peripheral bus populated with lower bandwidth devices such as the keyboard and serial port. In the case of Windows-based Intel (“Wintel”) systems, the legacy ISA bus fulfills the role of the peripheral bus.

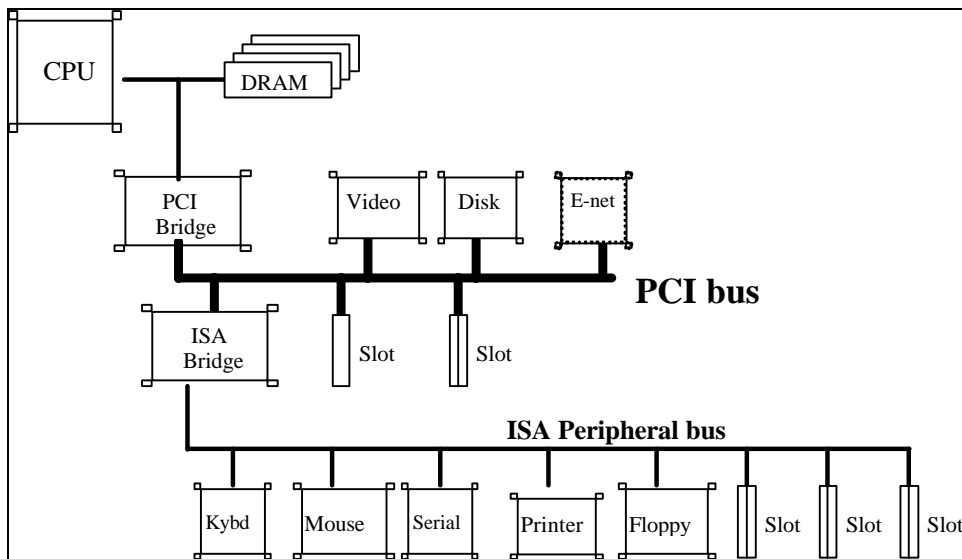


Figure 1. The PCI local bus design makes provisions for additional peripheral bus structures, such as the ISA bus, through the use of bridge chips.

What Does PCI Bring to Data Acquisition Systems?

Benefits Overview

PCI offers clear benefits for data acquisition, including

- Sustained transfer rates up to 120 Mbytes/s
- Choice of operating systems – Windows NT, Windows 95, Windows 3.1, Mac OS
- True Plug and Play installation and configuration
- Memory Mapped I/O
- Shared Interrupts
- Bus mastering for high speed data transfer between boards and memory
- High quality, high performance, and low cost

System Architecture Implications

PCI is a 32-bit data and 32-bit addressing, synchronous bus operating at clock frequencies up to 33 MHz. The PCI communication protocols are premised on transferring bursts of information. In a typical transfer, a peripheral device transmits an address for one bus clock cycle followed by a series of data values. One data transfer can occur on each clock cycle, which yields a maximum burst transfer rate of 132 Mbytes/s at 33MHz. Through bus mastering sustained transfer rates of 120 Mbytes/s are attainable. The PCI specification includes provisions to increase data throughput via 64-bit data transfers and increases the base clock frequency of 66 MHz.

The average bandwidth on older bus architectures, such as ISA, is 1-2 Mbytes/s. However, by using PCI DAQ boards you can achieve continuous system transfer rates of 120 Mbytes/s which will continue to increase with the introduction of more efficient PCI devices and chip sets. Assuming 16-bit data acquisition, sample rates of 20 MS/s are achievable without the need for on-board memory. Keep in mind that for certain, slower acquisition rate applications, a PCI DAQ board may not be necessary. For example, if your application requires a single DAQ board sampling at 40 kS/s, then an ISA board may be more than sufficient. In such applications, the update rate of the DAQ board is slower than either the cycle time on PCI or ISA. In the case of low bandwidth sampling, a customer is better off using one of the many ISA slots on typical Wintel machines instead of using one of the few PCI slots.

As will be shown, the PCI bus and PCI devices do not necessarily guarantee high data acquisition performance. Because PCI coexists with other components of a computer system, such as the operating system and other peripheral devices, designers of PCI-based DAQ products must develop products that optimize the performance of the entire computer system, not just the hardware. The most pressing issues to consider for high performance data acquisition are PCI bus master capability, interrupt utilization and OS memory management.

Many PCI DAQ products on the market support PCI slave-only functionality, but to take advantage of PCI benefits, a PCI DAQ board should have PCI master capability. Devices acting as bus masters will almost always transfer data more rapidly and efficiently than devices that act only as a slave and rely on another agent to initiate transfers. This improved efficiency is not only a quality of PCI but is inherent in a multi-master architectures. A bus master knows how much data it has and, presumably, where it needs to go; so it is easier for the bus master to control the data transfer and free the CPU for executing other tasks, such as numerical analysis, graphics, and network communication.

Plug and Play

PCI brings plug and play capabilities through a set of configuration registers that all PCI devices must contain. These registers permit a computer system to examine a PCI device and determine such things as the type of device (video, disk controller), how much address space the device requires and whether the device generates interrupts. Installation and configuration of PCI data acquisition devices is much easier and faster than ISA devices.

Memory Mapped vs. I/O Addressing

In developing PCI as a high speed peripheral bus, Intel included memory mapped I/O capability, and they recommend its use to ensure efficient PCI bus use. Direct I/O address operations are not recommended because such device programming decreases over all system performance. Peripherals used in Wintel computers typically use an I/O address space that is accessible through special CPU instructions. I/O reads and writes are generally issued by the CPU to address I/O space on PCI devices. I/O commands often serialize operations within PCI chip sets and within the CPU. This serialization will degrade system performance by increasing CPU utilization. In addition, I/O addressing is a method unique to Intel processors and is not widely implemented in other processors, thereby limited the extension of PCI devices from Intel computers to other processor based computers.

Another benefit of memory mapped I/O is ease of software programming. Accessing a memory mapped peripheral constitutes de-referencing a memory pointer in much the same way an array of data is accessed. This requires very little CPU involvement. Finally, device drivers written for memory mapped peripherals are easier to adapt to PCI systems with different CPUs.

KEY POINT: *For best system performance, select hardware with driver software that uses memory-mapped I/O, not direct I/O addressing.*

Interrupts

Using interrupts in ISA-based systems has always been a challenge. The ISA interrupt includes 16 interrupt levels. Over the years, by convention, many of the IRQ levels have been reserved for specific devices, and relatively few interrupts are available for general use. The PCI specification requires that PCI devices share interrupt signals, and this increases the availability of interrupts and alleviates the ISA interrupt constraints. When a PCI device is installed in a system, the interrupts service software for the device is added to a list of interrupt service routines (ISR) for all the PCI devices that share a common interrupt signal. When the CPU detects an interrupt assigned to the PCI bus, the CPU executes the first ISR in the list. This first routine determines if its associated PCI device asserted the interrupt. If so, the software services the interrupt. If not, the current interrupt software stops executing, and the CPU executes the next interrupt software in the list. This process is called interrupt chaining and its primary benefit is that PCI devices will never run out of interrupt resources.

How Does Bus Mastering Improve System Performance?

The PCI bus operates as a multiple-master and multiple-slave bus. Bus master devices take charge of the bus and direct the flow of data via direct memory transfers (DMA), and slave devices simply accept data sent to them from another master or from the processor. A bus master uses DMA to generate memory addresses for writing data from the board to memory, or for reading data from memory to the board. Bus master boards relieve the microprocessor from the burden of transferring data between the board and memory thereby affording the microprocessor more time for other computational tasks.

Bursting data is integral to the PCI signaling protocol. Even single data transfers are treated as a burst transfer of one data element. Because PCI protocols treat bursts as part of the standard, vendors developing PCI peripherals deliver high performance without implementing optional, less widely accepted, or even costly protocols. Given the architecture depicted in the previous figure, PCI masters easily access system memory via the PCI bridge without direct intervention from the CPU. This independent access to memory is THE key to maximizing PCI performance.

PCI is a multiprocessor bus with many provisions for master mode capabilities. Unlike the ISA and EISA bus systems, which include a DMA controller chip for transferring data, PCI does not have a built in DMA controller. In fact, the PCI specification leaves the implementation of DMA up to the device manufacturer.

Bus Mastering – A Common Example

A common analogy that illustrates the difference between bus mastering and slave data transfer is the type of telephone system used by business to route calls. Either the company uses a central receptionist who receives all incoming calls and then routes the calls to the correct internal telephone, or the company uses direct inbound dialing (DID) which directly routes each in bound call to the correct internal telephone without the intervention of the receptionist.

In the first case, the receptionist acts as the CPU of the phone system and is responsible for directing all inbound calls from a caller to the correct employee. As the number of calls increases the receptionist becomes overwhelmed with many request, and the rate of each call routing decreases. As a caller, you would experience long hold times if the company is receiving many phone calls. This case is analogous to the transfer of data to and from a slave device and computer memory.

On the other hand, in a DID system, each employee has a unique telephone number that can be reached directly from an inbound call without the intervention of the receptionist. If you were to call a company that uses DID, you would experience the benefits of bus mastering. Compared to bus mastering, the caller controls the telephone lines

(the bus) and access the employee phone without disrupting the receptionist (CPU). The benefit is that the receptionist is free to do other tasks, and the calls (bus accesses) take place faster.²

Bus Mastering – A Technical Explanation

In theory, for non-bus master or microprocessor-controlled transfers, the fastest sustained transfer rate will top out at 13 Mbytes/s – assuming 10 clocks for each cycle. This is only one-tenth the theoretical maximum of 132 Mbytes/s. The reduction in speed is due to the various components of the bus architecture. In a transfer, the microprocessor instructs the PCI bus controller to transfer data between DRAM and various devices. This interaction requires more than one clock cycle. A FIFO in the controller serves as a buffer between the fast writes of the microprocessor and the slower reads of the DRAM, whose latencies are longer than one clock.

Although PCI can transfer data at one 32-bit word per clock, the microprocessor cannot copy data between system memory and the PCI controller that fast. The problem gets worse when the microprocessor is executing multiple tasks, such as updating video or writing to disk. FIFO buffering in the PCI controller improves the situation, but overall system performance is still limited when the microprocessor moves data between DRAM and the PCI controller. In the worst case, the microprocessor is involved on every data transfer and has less time for other tasks, as shown in Figure 2. PCI devices that do not control the PCI bus, but that depend upon the CPU to orchestrate the data transfer, are slave devices.

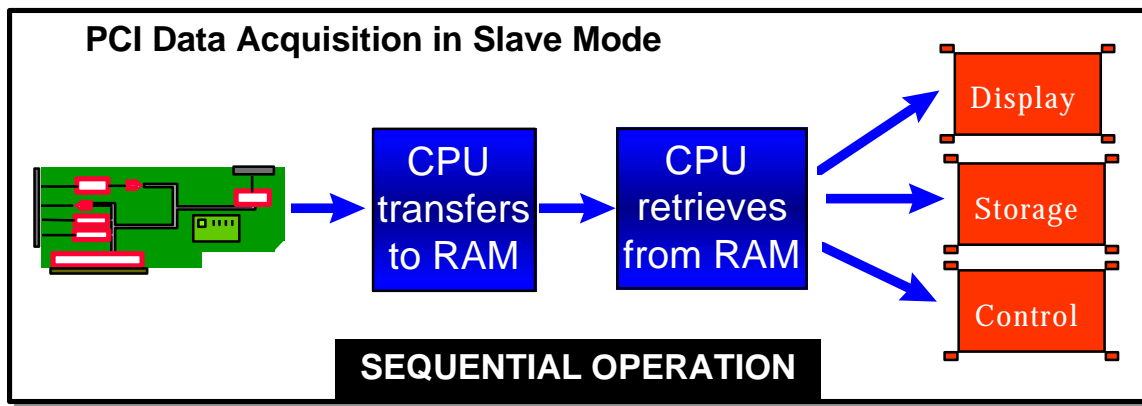


Figure 2. PCI data acquisition boards operating in slave mode reduce the overall system performance data transfer is a serial operation dependent upon CPU intervention.³

PCI bus masters, equipped with DMA, are more efficient because they can communicate to system memory without the microprocessor. DMA minimizes latency in servicing a PCI device because the controller responds more quickly than interrupts, and transfer time is shorter. Using DMA to coordinate the transfer of data between board and computer memory frees the microprocessor so that it can operate on more computational tasks. Also, in systems where the processor primarily operates out of a cache, data transfer is actually occurring in parallel, thus increasing overall system utilization. A typical bus master DMA controller knows how much data to move and where it is located. An advanced, well-designed bus master DMA controller knows the depth of the PCI controller FIFO and moves data at the right pace so as not to waste PCI bandwidth.

DMA Controller Operation

For each channel, the DMA controller saves programmed addresses and counts in registers. It maintains copies of this information in the current address and count registers. When DMA is started by writing to the base registers and enabling the DMA channel, the current registers are then loaded from the base registers. With each DMA transfer, the value in the current address register is driven onto the address bus, and the current address register is automatically incremented or decremented. The current count register determines the number of transfers

² "Data Acquisition Gains New Highs with PCI," Muneeb Khalid, *Evaluation Engineering*, May 1997.

³ National Instruments, "How Windows NT and PCI Deliver New Data Acquisition Solutions," Seminar, February 1997 Edition, part number 350317A-01.

remaining and is automatically decremented after each transfer. When the value in the current count register goes from 0 to -1, a *terminal count* (TC) signal is generated, which signals the completion of the DMA transfer sequence. This termination event is referred to as “reaching terminal count.”

For multiple buffer data transfers, the DMA controller must be reprogrammed when a DMA channel reaches TC. Thus, time is needed for the microprocessor to respond and reprogram the controller – but far less time than for the CPU to service device I/O interrupts.

DMA controllers also have mechanisms for automatically reprogramming a DMA channel when the DMA transfer sequence completes. These mechanisms include *autoinitialization* and *buffer chaining*. The autoinitialization feature repeats the DMA transfer sequence by reloading the DMA channel's current registers from the base registers at the end of a DMA sequence and re-enabling the channel. Buffer chaining, also known as scatter-gather, is useful for transferring blocks of data into noncontiguous buffer areas or for handling double-buffered data acquisition. With buffer chaining, a channel interrupts the microprocessor and is programmed with the next address and count parameters while DMA transfers are being performed on the current buffer. Some DMA controllers minimize microprocessor intervention further by having a chain address register that points to a chain control table in memory. The DMA controller then loads its own channel parameters from memory. Generally, the more sophisticated the DMA controller, the less servicing the microprocessor has to perform.

Which System Level Problems Are Not Solved with PCI?

Although the PCI bus extends many performance and ease-of-use advantages to users of PCI-based computers, problems and limitations still exist because the PCI bus is only a hardware specification, not a system level specification. Device designers use software to correct the system performance limitations inherent in PCI.

Interrupt Chaining and Latency

According to the PCI specification, “the system vendor (computer vendor) is free to combine the various interrupt signals from the PCI connector in any way to connect them to the interrupt controller.” This is ultimate flexibility for the designer, but it may inadvertently limit the overall system performance especially in PCI/ISA combination machines. For example, in a typical PCI/ISA combination design, all the INTA# are connected and all the INTB# are connected (Figure 3). The combined traces are routed to a programmable interrupt router which assigns the interrupt combination to an existing ISA interrupt. The disadvantage of interrupt chaining is an increase in interrupt latency depending on where a specific interrupt service routine resides in the chain.⁴

Although interrupt chaining solves the problem of allocating limited interrupts resources, doing so comes at a cost. The interrupt latency (the elapsed time from a device invoking an interrupt until its associated interrupt service routine executes) varies depending on where the interrupt service routine for the device is located in the chain. Long and non-deterministic interrupt latencies are hinderances in time critical applications. Not only that, but the CPU must execute the interrupt chain on every interrupt, wasting processing cycles. Overall, interrupt chaining is beneficial because it solves the ISA interrupt resource problems. However, the increased interrupt latencies associated with chaining necessitates that PCI devices issue interrupts infrequently and only when necessary.

KEY POINT: *Because system vendors are free to implement PCI interrupts in a combined PCI-ISA system in many different ways, the overall system performance can vary between computers if the only way to program the device is through interrupts. PCI-based DAQ devices should be bus masters and issue interrupts infrequently – only when necessary.*

⁴ Shanely, Tom and Don Anderson, *PCI System Architecture*, Mindshare, Inc., Addison-Wesley Publishing, 1995, page 216.

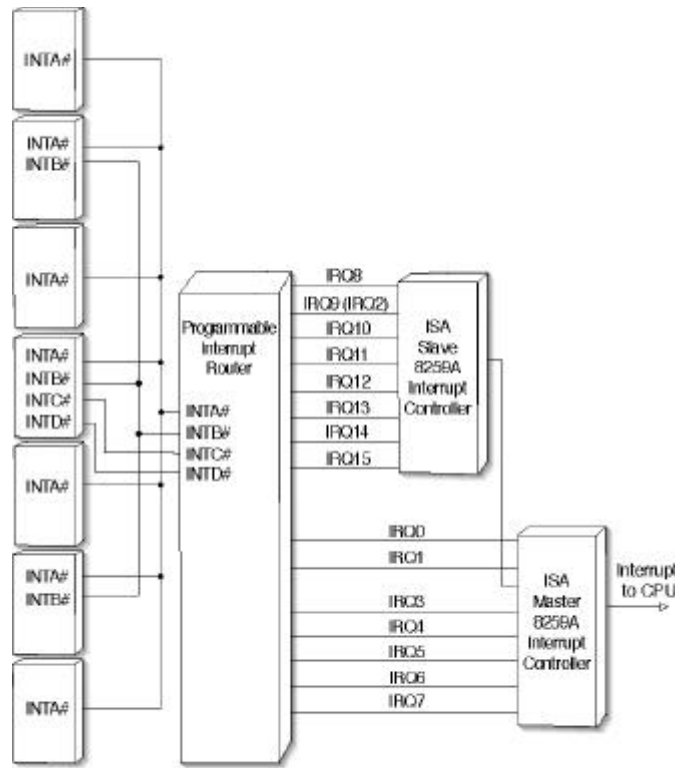


Figure 3. In combined PCI-ISA bus computers, the PCI interrupts are often chained together and associated with an ISA interrupt.⁵

Virtual Memory and Non-Contiguous Buffers

Bus masters, DMA controllers, and virtual memory managers are not a “marriage made in heaven.” Instead, they are at odds with one another. Virtual memory systems benefit the PCI software engineer and the user by providing an unlimited amount of physical memory. A “swap file” on a the computer hard drive, a secondary storage system other than main memory, functions as a virtual data buffer, but the actual memory location written to or read from by a data acquisition program may or may not be in physical memory at the time of access. If not, the virtual memory manager issues a page fault and loads from hard disk (pages in) the correct memory set from the swap file. If physical memory is in short supply at the moment of access, other pages may have to be paged-out to the swap file in order to make room in physical memory. Programs refer to virtual memory addresses, not physical addresses, and page faults occur when the virtual address does not match any physical addresses.

DMA controllers, on the other hand, reference only physical addresses. In order for a DMA controller to peacefully coexist with the virtual memory manager of an operating system, the driver software for the data acquisition board must assume the responsibility of monitoring the data location and managing the data transfer between physical and virtual memory. Before data can be transferred, the virtual memory manager must be instructed to move the data buffer from virtual memory to physical memory (page in). Enough physical memory must be available to contain the entire buffer. In addition, the physical memory must be “locked down” so that it is not moved to another memory location if the virtual memory manager needs memory of another operation.

Data Size and Free Memory Dependencies

The standard page size used by most virtual memory-based operating systems is 4 Kbytes. This means that once the data buffer is paged in, the buffer will be “scattered” across physical memory in tiny chunks of 4096 bytes each. Occasionally the virtual memory manager will produce larger memory blocks, but it is not obligated to do so

⁵ Shanely, Tom and Don Anderson, *PCI System Architecture*, Mindshare, Inc., Addison-Wesley Publishing, 1995.

because this reduces overall system performance. The PCI specification does not improve the relationship between DMA controllers and virtual memory managers. Rather the specification simply permits board vendors to design their own DMA controllers (bus masters) which may have features that eliminate the problems associated with virtual memory managers and DMA controllers, such as the ISA 8237 DMA controller.

Reporting the Progress of an Active Data Acquisition Process

The increased data throughput of PCI yields larger data sets. As has been described, the amount of data transferred by DMA is proportional to the amount of physical memory available for the freeing of and paging in of a data buffer. In continuous monitoring and control or high speed applications, the PCI DAQ board may be transferring more data than is physically possible to store in system memory. The larger data transferred across the PCI bus puts burden on the bus master chip to find enough free memory of the data transfer. Creatively designed driver software, along with intelligently designed hardware, overcomes the buffer size and free memory limitations.

How Should PCI Devices Be Designed for Use in Data Acquisition Systems?

Because the PCI specification is only a hardware standard and does not establish how a hardware and software dependent system should be built, designers of PCI data acquisition boards must take special care to not only develop hardware that meet the PCI specification, but also software drivers that optimize the use of PCI DAQ boards in virtual memory and PCI-ISA systems. Otherwise, you cannot be assured to get the best performance from your computer.

Interrupts

In alleviating the interrupt resource problem inherent in the ISA bus architecture, the PCI specification introduces the possibility of potentially unbounded latencies, as measured from the time of the issuance of the interrupt to the time that the correct software services the ISR. In practice, however, software engineers have been chaining ISA timer tick interrupts for years. Plus, non-real time operating systems, such as Windows 95 and 3.1, acknowledge they do not call a registered interrupt routine in a bounded fashion. Thus, interrupt chaining is not a new challenge. But, in order for a DAQ board to be a good citizen on the PCI bus system, the driver software should a) program the board to generate interrupts only when necessary, and b) write a primary interrupt handler so that it executes as fast as possible.

Limiting interrupt generation reduces interrupt traffic on the bus. To accomplish this, intelligently designed hardware is required. Intelligent design includes a self-programming DMA controller that does not need the CPU to execute an ISR to monitor or reprogram the DMA controller with memory locations or transfer counts. In addition, intelligent hardware design makes possible the insertion of an interrupt anywhere during a transfer, instead of asserting an interrupts on a recurring condition, such as transfer completion of a contiguous memory block, as is the case with most PCI-based data acquisition devices.

Fast interrupt handler execution minimizes the time between the occurrence of the interrupt and the time to execute the corresponding ISR. For example, an ISR routine must rapidly determine if it belongs to the asserted interrupt. If the ISR does not correspond to the interrupt, control must be quickly passed back to the operating system which activates the next ISR in the chain. Even if the ISR corresponds to the interrupt, assigning the actual interrupt handling to a secondary handler will improve overall system performance. Intelligently designed hardware makes for efficient software design which make possible a primary interrupt handler that deterministically recognizes interrupts and passes ISR execution to a secondary interrupt handler, rather than using the primary handler to service the interrupt, such as empty a FIFO, in order for the interrupt to be released.

Implementing Bus Mastering

Being a bus master device, Figure 4, requires more than simply moving data from one memory location to another. In virtual memory systems, such as Window NT and Windows 95, large virtual memory buffers exist as small blocks of physical memory scattered throughout the entire memory system. The operating system keeps track of how these small blocks link together to form the large buffer. Transferring large amounts of sampled data between the device and the fragmented memory can decrease the overall system performance of your computer. However,

three software driver techniques are used by hardware vendors for overcoming the challenge of transferring data between fragmented memory locations.

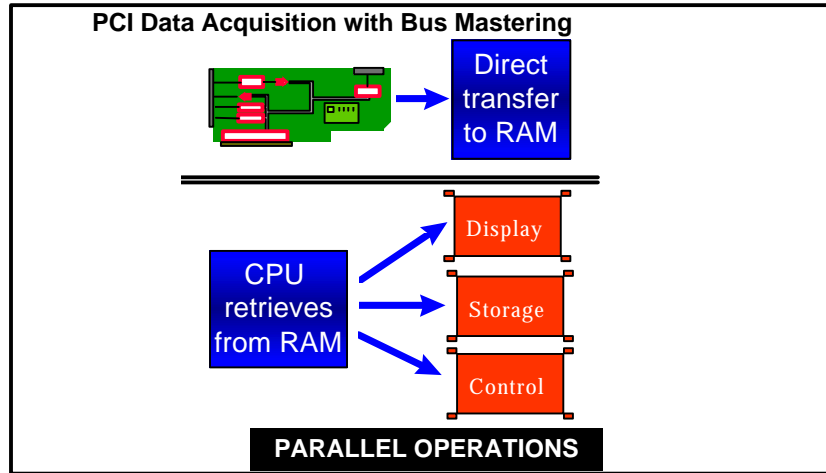


Figure 4. Bus master PCI data acquisition boards unburden the CPU by transferring data directly to system memory, leaving the CPU available for more appropriate activities. Parallel acquisition and data processing are possible. Bus mastering gives a 10 to 1 performance increase over slave mode transfers.⁶

Restricting Memory

First, the “quick and dirty” way to overcome physical memory fragmentation and act as a PCI bus master is to circumvent the virtual memory manager and reserve a portion of system memory specifically for data acquisition. In Windows 95 systems, the memory can be reserved during boot time. Although this method sets aside a contiguous block of memory that a simple PCI bus master can use for data transfer, restricting memory is not a “friendly” solution for virtual memory-based operating systems, such as Windows 95 and NT. The restricted memory is not flexible because it cannot be resized programmatically during data acquisition. Additionally, the operating system is no longer aware of the removed memory, even if the reserved memory is only 500 KB. Restricting memory at boot time is tantamount to removing a memory SIMM from the your computer. In addition, the you must know in advance how many data samples are to be acquired and stored. For continuous acquisition and monitoring applications in which data is acquired in mass and over hours or even days, reserving a block of memory dramatically limits the system flexibility. A better method is to work with the virtual memory manager by designing both hardware and software with this goal in mind.

Scatter-Gather

Scatter-gather DMA is the second method of implementing bus mastering – rather than restrict memory to the data acquisition hardware, you work with the virtual memory manager. Because the virtual memory manager scatters the data buffer accross physical RAM in 4 KB chunks, the DMA controller must gather the chunks together for the transfer. The DAQ board driver software obtains the starting physical addresses of each chunk and the number of bytes in each chunk. Scatter-gather DMA has been a method of data transfer in virtual memory manager systems for many years and with the ISA 8237 DMA controller. Although many hardware vendors use scatter-gather DMA, there are different implementations.

Reprogramming DMA Controllers with Interrupts – Most scatter-gather bus master implementations used by some DAQ vendors rely on interrupts to notify the CPU when the DMA controller has completed a data transfer of one block of data to system memory. The CPU reprograms the bus master chip with the memory address of the next available scattered memory location. Although this method conforms well to a virtual memory architecture, many interrupts are generated, demanding CPU intervention. Furthermore, in systems where the PCI bus master board shares interrupts with other devices, the overall system performance will be adversely reduced as the CPU becomes more overwhelmed with activities.

⁶ National Instruments, “How Windows NT and PCI Deliver New Data Acquisition Solutions,” Seminar, February 1997 Edition, part number 350317A-01.

Reprogramming the DMA controller still limits the overall system performance of PCI. Consider a double-buffered DAQ application in which the DMA controller transfers 64 KB of data to different memory locations, one beginning at address location 10000 and the other beginning at address 20000. Each time one transfer is completed, the DMA controller generates an interrupt which causes the microprocessor to reprogram the DMA controller for the next address. The microprocessor involvement includes:

- Step 1:** Programming the DMA controller to transfer data to memory beginning at location 10000.
- Step 2:** Starting the DMA transfer.
- Step 3:** Waiting for DMA completion interrupt.
- Step 4:** Reprogramming the DMA controller to transfer data memory beginning at location 20000.
- Step 5:** Starting the DMA transfer.
- Step 6:** Waiting for DMA completion interrupt.

To see the impact DMA reprogramming has on system performance, consider a PCI device which supplies data to the PCI bus at 30 Mbytes/s. The PCI bus is able to transfer the data at 35 Mbytes/s per DMA transfer. However, a delay occurs each time the microprocessor reprograms the DMA controller. As a result, the sustained system throughput will be less than 30 Mbytes/s depending on how much time the microprocessor needs to reprogram the DMA controller.

On-the-Fly Scatter-Gather Bus Mastering – The preferred bus master implementation is a DMA controller designed to accept memory addresses and sizes “on the fly,” Figure 5. This type of bus master implementation, also known as link chaining, is a combination hardware and software solution whereby the controller is intelligently designed to reprogram itself without the use of interrupts and CPU intervention. In link chaining DMA mode, the DMA controller is not reprogrammed for new memory locations. Instead, it configures itself automatically by reading a series of linked data records stored in some small memory buffer. The record contains the source address of the data, the destination address of the data, the number of bytes to transfer, and the address of the next linked data record. The DMA controller then reads and executes each data link record in the list. The microprocessor needs only to program the DMA controller with the address of the first link data record, then the DMA controller does the rest.

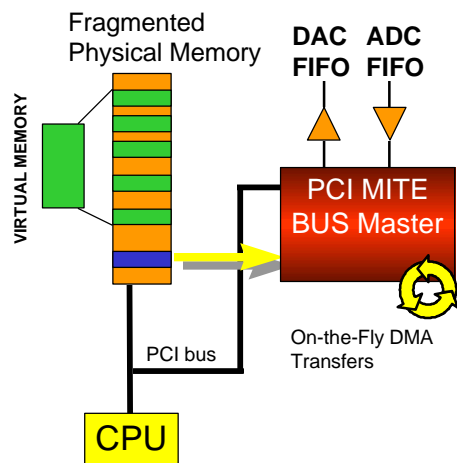


Figure 5. On-the-Fly scatter-gather bus mastering is the most efficient method of data transfer in virtual memory systems. The Bus master knows the scatter memory locations and transfers data without burdening the CPU or dedicating memory to only data acquisition.⁷

The advantage of an on-the-fly scatter-gather bus mastering is efficient operation with virtual memory managers and no CPU intervention. In theory, an arbitrary buffer up to the limits of system memory can be transferred with no negative impact on overall system performance. No longer are numerous interrupts generated for each scattered data buffer. Instead, the bus master judiciously asserts interrupts pertinent to the data acquisition process, such as, an interrupt informing the CPU that all the directives have been executed. This method of bus mastering is the fastest mode of data transfer because the controller has direct access to the data, knows where the location of the data, and does not require CPU intervention. Also of benefit is that all the system memory is available for not only data acquisition but also for other applications to use.

⁷ National Instruments, “How Windows NT and PCI Deliver New Data Acquisition Solutions,” Seminar, February 1997 Edition, part number 350317A-01.

Cooperation Between Driver Software and Bus Master DMA Controller

For continuous acquisition and control applications where the amount of data is greater than the storage buffer, the DAQ driver software must read from and write to the buffer during the acquisition process. The driver must know how much data has been acquired at any given moment. A sample counter on the board counts the number of analog-to-digital converter (ADC) conversions. Unfortunately, the sample counter does not reveal the amount of data in system memory. One or more FIFO buffers sit in the data path between the ADC and system memory to buffer data in case the service latencies are too long. This avoids data loss in the event that data can not be transferred from the board to system memory fast enough. Some data may be waiting in the FIFO. In continuous acquisition counters roll over and resume counting at zero. They are monitored by the CPU to detect a roll over case. The best way to determine that amount of data in system memory is to directly query the DMA controller.

A properly implemented PCI bus master chip should include a count and memory address register so that the driver software can perform a single, 32-bit read and determine the amount of data in system memory. Memory addresses are more advantageous than sample counters because they are unique to the system and do not roll over. Driver software determines the exact location of the newest data point in a buffer by reading the memory address. To fully decouple the data set size from the size of the data acquisition buffer the DMA controller must fill and refill the buffer. Thus, the driver software needs a way to determine the number of times the buffer has been refilled. The best solution is an advanced DMA controller that is programmed by the device driver to assert an interrupt only at the end of the buffer. An interrupt service routine counts the number of time the buffer is filled.

The PCI specification leaves bus master implementation up to the DAQ board designer. Now the responsibility of delivering data acquisition solutions that deliver the best performance is up to the board vendor. The lack of DMA support on the PCI prompted National Instruments, a leader in data acquisition solutions, to develop the MITE™, a custom ASIC that performs advanced, efficient DMA block transfers. PCI DAQ boards built with the MITE ASIC deliver the best overall data acquisition system performance.

The MITE ASIC acts as an intelligent bus master chip capable of on-the-fly scatter-gather data transfers. Three DMA controllers are available and can be assigned for use in analog input, analog output, and counter/timer data transfer. The on-board DMA controllers operate in several modes, including ring-buffer transfers, large contiguous block transfers, and noncontiguous transfers (on-the-fly scatter-gather, link chaining) by reading and executing DMA instructions stored in system memory. The noncontiguous transfer capability is especially important in virtual memory operating systems. The MITE DMA controller has a very flexible self-configuration capability that is used to count the number of times a DAQ buffer has been filled. Even if the DAQ board samples at 1 million samples/s into a 100,000 sample buffer, the process can run indefinitely and will generate only 10 interrupts/s! The PCI DAQ boards using the MITE sustain data transfers between different blocks of memory without needing to reprogram the DMA controller.

To ensure a unified PCI architecture for all our products, National Instruments PCI boards use the MITE ASIC. A unified PCI architecture benefits both the hardware developer and you. For the developer, time to market is reduced because a the PCI interface works with a variety of computer platforms, including Intel, Sun, HP, and Macintosh. More capable products are achievable that extends the computer-based test, measurement, monitoring, and control solutions into new application areas. A unified architecture give you a migration path to new computer systems. You save time and money, because you no longer have to buy a new board if you need to use a different type of computer. You have a greater selection and flexibility to choose the data acquisition solution that best meets your needs.

System Performance Tests

National Instruments test experience shows a dramatic speed boost when PCI DAQ boards are used in bus master mode. In our labs, we have seen at least a 3:1 improvement in performance over slave-only operation and as high as 10:1 with the latest generation of PCI chip sets. But the real benefit of PCI bus mastering is evident when real world application tests are performed.

Simple Data Transfer Tests

The following test demonstrates the performance difference between a 1.25 MS/s, multifunction DAQ board which is available for both PCI and ISA. The tests involved acquiring 400,000 samples and graphing the data on the screen – a very CPU intensive activity. The tests were performed on both 200 MHz Pentium and 180 MHz Pentium Pro computers with 32 MB RAM running Windows 95 and LabVIEW 4.1. The program continuously acquires data and increases the sample rate until a buffer overflow error occurs indicating that data was lost. For example, in Figure 6, a 1.25 MS/s PCI DAQ board configured in slave mode samples at 720 kS/s (58% of the board’s capacity) before data is lost. The Pentium 200 MHz CPU can not respond to the data transfer interrupts and update the graph fast enough. The transfer rate is slower than an equivalent ISA board using DMA. But the same PCI DAQ board operating as a bus master does not require CPU intervention and maintains a sustained transfer rate of 1.25 MS/s (100% of the board’s capacity), and all the data is transferred.

Pentium 200 MHz Test Results

<u>Board Type</u>	<u>Board Name</u>	<u>Sustained Sample Rate</u>	<u>Efficiency</u>
ISA with DMA	AT-MIO-16E-1	775 kS/s	62%
PCI slave	PCI-MIO-16E-1	720 kS/s	58%
PCI bus master	PCI-MIO-16E-1	1.25 MS/s	100%

Pentium Pro 180 MHz Test Results

<u>Board Type</u>	<u>Board Name</u>	<u>Sustained Sample Rate</u>	<u>Efficiency</u>
ISA with DMA	AT-MIO-16E-1	1.1 MS/s	88%
PCI slave	PCI-MIO-16E-1	920 kS/s	74%
PCI bus master	PCI-MIO-16E-1	1.25 MS/s	100%

Table 1. The processor speed has an affect on the achievable transfer rate, but the presence or lack of a PCI bus master device (or DMA) impacts system performance much more.

The test results indicate that the type of processor (Pentium vs. Pentium Pro) impacts the total system performance, Figure 7. The faster the processor, the faster the CPU can perform multiple tasks, such as service DAQ board interrupts and update the screen. But the increase in CPU speed is still not enough to overcome the burden a PCI DAQ board in slave mode places on the system. In fact, the attained performance increase is greater on the ISA bus than on the PCI bus. The Pentium Pro caused the sustained transfer rate of the ISA board (AT-MIO-16E-1) to increase 42%, where as the transfer rate of the PCI slave board (PCI-MIO-16E-1 in slave mode) only increased 34%. The data is further evidence of the benefit of transferring data with DMA.

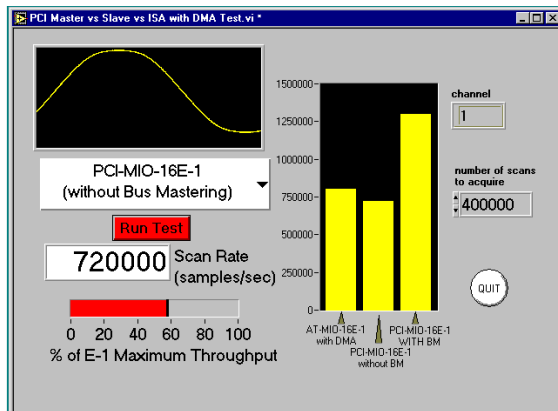


Figure 6. Micron, Pentium, 200 MHz, 32 MB, LabVIEW 4.1, Windows 95

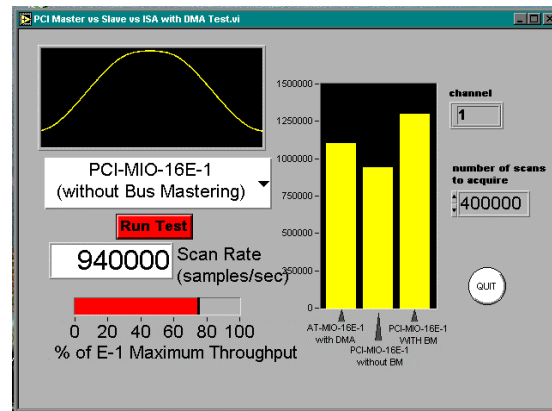


Figure 7. Micron, Pentium Pro, 180 MHz, 32 MB, LabVIEW 4.1, Windows 95. A faster processor increases the overall system performance, but bus mastering still dramatically outperforms PCI slave boards.

Stream to Disk Test

Another useful test involves streaming data to disk. Disk operations are always burdensome on the CPU, especially if the hard disk is fragmented or has little free space. The following test was performed using a Pentium Pro 200 MHz, 32 MB computer running Windows 95 and LabVIEW 4.1. Fifty million samples are acquired and written to a five million sample buffer in blocks of 100,000 samples. The data is stored as raw data (binary), as this does not require the CPU to scale the data. If scaled data (volts, temperature, strain, and so on) is to be save to disk, the achievable transfer rate will decrease as the CPU must also execute a scaling calculation. A 100 Mbytes (50 Msamples * 2 bytes/sample) data file is created, and the 5 Msamples buffer is completely filled 10 times during the acquisition process.

Pentium Pro 200 MHz Test Results

<u>Board Type</u>	<u>Board Name</u>	<u>Sustained Acquisition to File</u>	<u>Efficiency</u>
ISA with DMA	AT-MIO-16E-1	700 kS/s	56%
PCI slave	PCI-MIO-16E-1	200 kS/s	16%
PCI bus master	PCI-MIO-16E-1	1.25 MS/s	100%

Table 2. Streaming data to disk demands CPU attention. The performance boost from a PCI bus master board is dramatic.

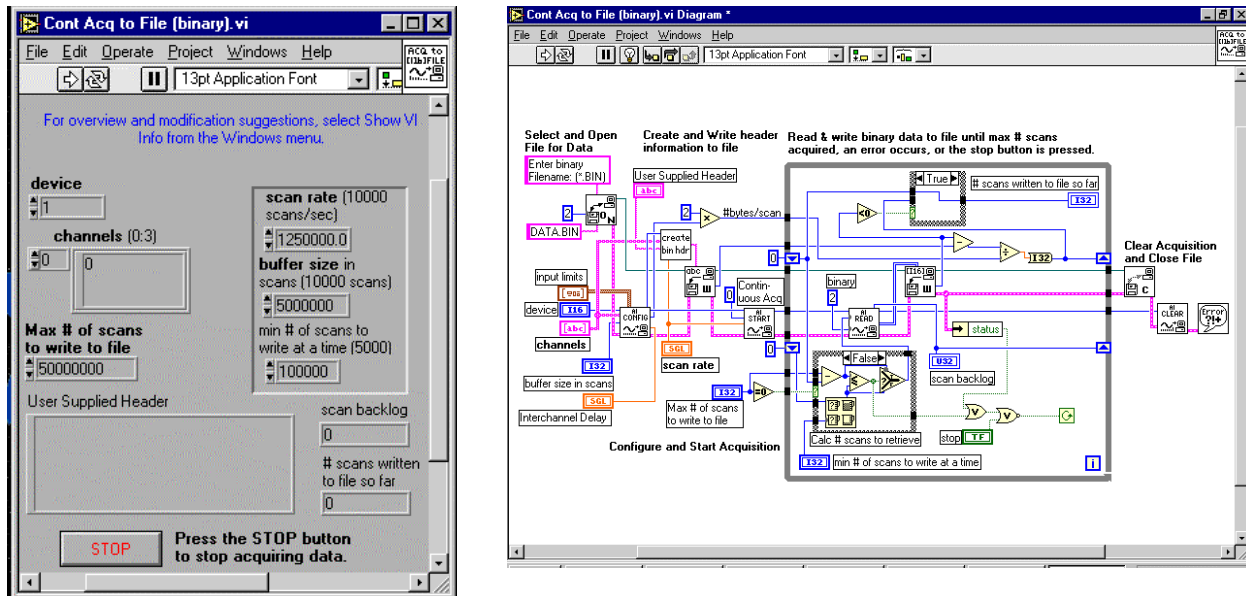


Figure 8. This stream data to disk program ships as a standard example with LabVIEW so that you can quickly determine rate at which you can save data to disk with your computer.

The benefits of bus mastering and DMA transfer of data is even more dramatic when streaming data to disk. The system using the ISA DAQ board streamed data to disk at 700 kS/s, using only 56% of the sampling capability of the board. But the same system using PCI DAQ board operating as a slave device stored data at only 200 kS/s, a low, only 16% of its acquisition capacity. The same PCI board used as a bus master effortlessly streamed data to disk at 1.25 MS/s – what you would expect from a bus master device operating across the wide PCI bus. For this application, if faced with the choice between an ISA board with DMA or a PCI slave device, you would achieve better overall system performance if you used the ISA board.

Conclusion

As virtual instrumentation systems continue to be used as legitimate alternatives to expensive, stand-alone instruments, the capability of computer technology to deliver the functionality and performance of sophisticated instruments becomes more and more essential. The recent advances in CPU speed, memory, operating systems, and I/O bus systems propel computer-based measurements and control systems into application areas once serviced by traditional, vendor-defined instruments. None of the recent advanced in computer technology has been as beneficial and dramatic as the acceptance and use of the PCI bus as an instrumentation bus and the use of virtual memory operating systems, such as Windows 95 and NT. But, as INTEL concluded in their PCI report, the “proper use of the PCI bus will maximize system performance while minimizing the load on the PCI bus and the CPU.”⁸ Even though, the PCI specification provides rich support for bus mastering and is laced with suggestions about its benefits, DAQ board developers have the responsibility to develop products that work in harmony with virtual memory managers and that make the most efficient use of CPU processor time and PCI specifications.

Key Points to Remember When You Select a PCI-Based DAQ Board:

1. Because of the many PCI-ISA interrupt issues, select a PCI DAQ board that issues interrupts infrequently, and only when absolutely necessary.
2. For best system performance, select hardware with driver software that uses memory mapped I/O, not direct I/O addressing.
3. Select PCI data acquisition products that use on-the-fly scatter-gather bus mastering to be sure that you are getting the most performance out of both your data acquisition board and Pentium-based computer.
4. Select PCI hardware from a vendor whose driver is designed to work in harmony with virtual memory operating systems, such as Windows NT and 95.

Resources

Khalid, Muneeb, “Data Acquisition Gains New Highs with PCI,” *Evaluation Engineering*, May 1997.
Kimery, James, “DMA Optimizes VXI System Performance,” *Instrumentation Newsletter*, Fall 1997, National Instruments customer publication.

McConnell, Edward, “DMA Fundamentals for Maximizing PCI bus Performance,” *ECN*, April, 1996.
PCI Specification Version 2.1

Platform Architecture Labs, “Efficient Use of PCI,” Intel Corporation, 04/22/97,
http://support.intel.com/oem_developer/chipsets/pci/general/pci001.htm.

Ryan, Arthur, “The Impact of PCI on Data Acquisition and Test Applications,” *Proceedings of the Telecommunication and Industrial PCI Conference*, November 1996.

Shanely, Tom and Don Anderson, *PCI System Architecture*, Mindshare, Inc., Addison-Wesley Publishing, 1995.
National Instruments, *How Windows NT and PCI Deliver New Data Acquisition Solutions Seminar Handbook*, February 1997 Edition, part number 350317A-01.

⁸ Platform Architecture Labs, “Efficient Use of PCI,” Intel Corporation, 04/22/97,
http://support.intel.com/oem_developer/chipsets/pci/general/pci001.htm.



Tel: (512) 794-0100 ❖ Fax: (512) 794-8411 ❖ E-mail: info@natinst.com ❖

www.natinst.com

PRELIMINARY as of 081497

Part Number 341312B-01

Copyright 1997 National Instruments Corporation. All rights reserved.
Product and company names listed are trademarks or tradenames of their respective companies