

PCI-ATA-DMA

Version 1.0

May 24, 1995



Technical Editor:

Curtis E. Stevens
Phoenix Technologies
2575 M^cCabe Way
Irvine, Ca. 92714
Phone: (714) 440-8000
Fax: (714) 440-8300
Curtis_Stevens@PTLTD.COM

Phoenix Technologies Ltd.

THIS SPECIFICATION IS MADE AVAILABLE WITHOUT CHARGE FOR USE IN DEVELOPING COMPUTER SYSTEMS AND DISK DRIVES. PHOENIX MAKES NO REPRESENTATION OR WARRANTY REGARDING THIS SPECIFICATION OR ANY ITEM DEVELOPED BASED ON THIS SPECIFICATION, AND PHOENIX DISCLAIMS ALL EXPRESS AND IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND FREEDOM FROM INFRINGEMENT. WITHOUT LIMITING THE GENERALITY OF THE FOREGOING, PHOENIX MAKES NO WARRANTY OF ANY KIND THAT ANY ITEM DEVELOPED BASED ON THIS SPECIFICATION WILL NOT INFRINGE ANY COPYRIGHT, PATENT, TRADE SECRET OR OTHER INTELLECTUAL PROPERTY RIGHT OF ANY PERSON OR ENTITY IN ANY COUNTRY. USE OF THIS SPECIFICATION FOR ANY PURPOSE IS AT THE RISK OF THE PERSON OR ENTITY USING IT.

Copyright Phoenix Technologies, May 1995

Phoenix Technologies LTD
2757 M^cCabe Way
Irvine, Ca. 92714

Phone: (714) 440-8000
Fax: (714) 440-8300

Revision History			
Rev	Date	Author	Description
1.0	24 May 1995	Curtis E. Stevens	Initial Release

Introduction

The ability of ATA devices to utilize Direct Memory Access (DMA) and the inability of the Peripheral Control Interface (PCI) bus to perform DMA operations has exposed several problems. This paper addresses two questions that are becoming particularly important for system designers:

- BIOS-Memory Manager Interface — How can the BIOS provide a DMA adapter with *physical* memory addresses needed for disk access?
- BIOS-PCI-DMA adapter usage — Is PCI DMA treated as a PnP resource, or part of a PCI device?

ATA-DMA

Background Two years ago DMA was not a viable method for accessing ATA drives because it was slower than conventional PIO (Programmed Input Output). The advent of “F” mode DMA for ISA and “B” mode DMA for EISA improved the performance of DMA-based ATA drives, making DMA competitive with PIO. These new DMA modes gave systems using this I/O capability a theoretical maximum transfer rate of 8MB/s. At that time the fastest PIO mode was mode 2, which also delivered a theoretical maximum of 8MB/s.

Limitations of ATA PIO Both of these transfer methods were limited to 8MB/s because of the performance of the ISA bus. VL and PCI solved the 8MB/s limitation, but a new issue is going to move the industry toward DMA. The ATA-PIO interface, using an unterminated 18-inch cable, is electrically limited to 16 MB/s transfer rates. In the future, the ATA interface is expected to expand DMA transfers to 25MB/s. Drives which support fast DMA and fast PIO are already on the market, and drives which only support fast DMA are coming.

Phoenix and ATA DMA Recognizing the performance limitations of PIO, Phoenix chose to implement “mode 1” DMA when the first PCI bridge chips started including “fast” DMA channels. One early system equipped with Phoenix support and an Intel 82378, achieved

a Coretest speed of 4+ MB/s¹. Today, using “mode 2” DMA and a CMD 646, we are achieving Coretest speeds in excess of 14 MB/s.

An Issue with DMA However, there is another concern with DMA — Early DMA “solutions” worked on systems running only DOS, but failed on systems running Windows. The problem is caused by memory managers which are capable of presenting a *virtual* memory map that is quite different from the *physical* memory map, even for DOS. When Windows places the microprocessor in virtual-86 mode, the memory addresses passed to INT 13h are *virtual*, while DMA requires *physical* addresses. The presence of a memory manager requires the BIOS to either eliminate DMA support entirely or to interact with the memory manager to get the actual physical addresses when running Windows.

The Phoenix DMA Interface

Operating Systems that Don’t Use INT 13h

Although the BIOS starts a PC’s operation with a series of INT 13h calls from INT 19, many operating systems quickly take over, bypassing the BIOS. They do not use INT 13h because they require multi-tasking/protected-mode operation, and INT 13h can only work in a real-mode single-task environment. For example, Windows NT, SCO Unix, Linux, and OS/2 do not use the INT 13h interface and are, therefore, not effected by changes to INT 13h. As a result, the Phoenix DMA solution needs to work only for DOS/Windows.

INT 13h Uses VDS Services

Phoenix recommends basing an INT13h-DMA solution on existing IBM virtual DMA services. IBM defined a memory manager/ virtual DMA interface dating back to at least December, 1989. The virtual DMA services (VDS) are only enabled when 40h:7Bh bit 5 is enabled. **ALL** IBM-compatible memory managers must set this bit when they are active. Microsoft, Qualitas, and Quarterdeck support this bit, and they claim that Windows will not function if

¹Phoenix uses the Coretest speed evaluation tool in these examples as a method of measuring the BIOS overhead for accessing the device.

this bit is not supported. IBM specifies that this bit will be turned on whenever the machine is in virtual-86 mode and turned off when the machine is returned to real mode. The memory manager provides VDS services by extending INT 4Bh. The INT 13h-DMA Flowchart below diagrams the flow of control.

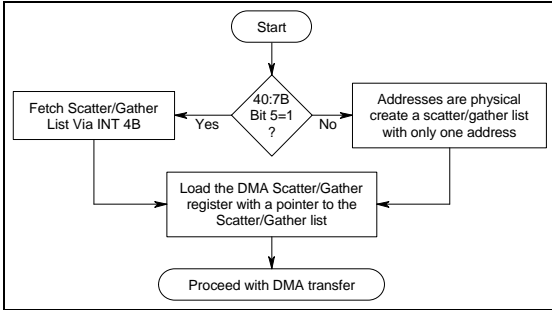


Figure 1 -- INT 13h Flowchart

The Phoenix INT 13h-DMA interface makes the following assumptions:

- INT 13h is used only by DOS/Windows and INT 19h.
- There is a real-mode service that memory managers provide for retrieving *physical* addresses.

Phoenix uses the VDS service to both *Lock* memory and retrieve *physical* addresses. The memory must be locked because INT 13h makes an INT 15h Fn 90h call while it is waiting for data. This call allows DOS/Windows to work on other tasks while the drive is retrieving data. The physical addresses are returned in the form of a scatter/gather list. INT 13h allows the Operating System to fill a 64k buffer. The resulting scatter/gather list associated with this buffer can have a maximum of 16 entries. This limit is created by memory managers which have a minimum page size of 4k. Several DMA controllers require that memory addresses do not cross a 64k boundary, so it is possible that this list doubles in size to 32 entries at 8 bytes per entry (or 256 bytes long). **This list is allocated at the end of the conventional 1k Extended BIOS Data Area.** Even though the Phoenix ATA-DMA implementation works in several environments, any environment that does not respect the use of 40h:7Bh bit 5 can cause the INT 13h interface to fail. For this reason, the end user needs a way of disabling INT 13h

DMA. Phoenix provides this capability via the “XFER MODE” disk drive SETUP menu entry. When the user autotypes the drive, the XFER MODE will be set to the maximum capability supported by both the drive and the platform. If a user wishes to operate in an environment that is incompatible with the Phoenix DMA implementation, he can manually change the XFER MODE to something other than DMA.

PCI

The PCI bus does not support the signals necessary for performing ATA-DMA operations. This has significant performance implications for high speed data transfer in today’s faster I/O systems. However, new technology provides a unique solution to the DMA problem. Many chip manufacturers are now including a full DMA adapter as an integral part of their Fast PIO chips. The figure below shows the flow of data when a PCI-DMA chip is present.

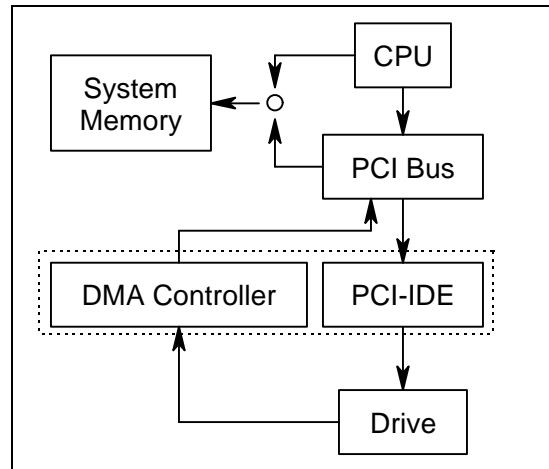


Figure 2 -- DMA Data flow

The question is whether this new DMA adapter should be treated like a PnP device, or simply as a PCI device. Phoenix is treating the DMA adapter as part of the PCI device, eliminating the need for a PnP device node. OS’s will see the DMA adapter as part of the PCI-ATA device.