X3T10/95-102

X3T10/95-102 1.0
November 30, 1994 9:59 am

Proposal for Enhanced IDE 95 Features

ATA-3 & ATAPI

Devon Worrell
November 30, 1994

WESTERN DIGITAL

073

1.0 Introduction

1.1 Purpose

The purpose of this document is to document a proposal from Western Digital. This proposal will add functionality to the IDE Specifications.

The issue is that more people in the world are now aware that more is to be gained by enhancing IDE than by promoting SCSI into the volume PC space. The obvious opportunity is to overlap the IDE channel, i.e. send one command per device. The motivations for overlapped IDE is multi-tasking OS. The next obvious opportunity is to add queuing, i.e. more than one command per device. This would allow for the DMA channels be multi-threaded.

We must realize that the existing single threaded dual channel DMA, single threaded dual channel IDE solution hasn't shipped and we need to get market focus behind it.....not focus all the interest on something that isn't available...ie. SWAT...Sell What is Available Today. The multi-threaded thing is at best a 2H95 product, more likely a 1H96 product.

1.2 Scope

This document will cover the key architectural points and other design issues. It is intended that this document be the basis for industry discussion.

1.3 Audience

This document is intended for use by Western Digital Engineers and Managers and attendees of ATA and SFF.

1.4 Technical Terms

There are a number of technical terms being used today to describe the capabilities that will be added to the interface. This of course is natural (Given that Marketing is involved), but could cause confusion when reading this proposal.

Multi-thread, Multi-threaded, Multi-threading	This must be the most misused term used today. It is used to describe everything from operating systems to lines at the bank. For the purposes of this document this term will be used to describe a very general capability of the IDE interface only. When used to describe the IDE Interface then it will mean that the interface can be used to transmit new commands to device(s) on the cable before the completion of any previous (pre-existing) command(s). This term will NOT be used to describe any capability of the Host or Devices attached to the IDE Cable.
Single Thread, Threaded or Threading	This will be used to describe the opposite of Multi-thread and will be used to describe a general capability of the IDE Interface only.
Overlap, Overlapped, Overlapping	This will be used to describe a capability of Device(s) attached to the IDE Cable.
Arbitration	When used within the context of this proposal, this will refer to the sharing of common signals or other resources (Registers)used

	in conjunction with the IDE interface, Host computers or Devices using the IDE interface.
Command Queuing	Simply put, allowing more than one command at a time to be accepted by the IDE Device.
Tagging	No not showing the world that you exist. This is synonymous with Command Queuing and is used interchangeably.
Controller	The electronics on a peripheral, that provides the IDE functionality.
IDE	Integrated Drive Electronics. This is the only acronym that I will use in this document. Although there are other names for the interface, such as ATA, ATA-2, and ATA-3, I will not be using them in this document. These are used by the Standards bodies as a pseudonym for the ubiquitous IDE Interface.
Semaphore	Used in the context of this proposal, a semaphore is any mechanism that is used to lock access to a common resource.
Task File, Task File Registers	This refers to he Registers that are used to control an IDE device. For the purposes of this proposal only those registers and signals that can be altered by either the Host or the Device are considered part of the Task File Registers. e.g. BSY, DRDY, DREQ etc. are NOT considered part of the Task File.

2.0 Goals

It is the intent of this proposal to provide some basic improvements to IDE. This package of improvements will be known as Enhanced IDE 95. The most basic improvement will be to allow commands to be sent to each of the two devices on one IDE Cable, independently. That is allowing commands to be processed in both devices at the same time. A further improvement will be to allow multiple commands to be sent to each of the devices on the IDE Cable. Although this may seem simple, the compatibility issues are extremely complicated.

2.1 Generic Goals

Keep it as simple as possible.
Allow Commands to be sent to each Device on the IDE cable independently.
Allow mixing of Legacy Devices and Newer capability Devices and still use the new capabilities.
Use DMA and independent commands as basis for greatest performance improvement.
Reduce the number of Interrupts to further improve performance.

Report Capabilities and Default to Legacy Mode.
Enable Capabilities that are backward compatible in Post, using Set Features.
Enable Capabilities that are not backward compatible on a command by command basis.
Single solution for ATA and ATAPI peripherals.
Use only the existing task file register space
Maintain compatibility with current silicon.
Support both PIO and DMA modes.
Support existing PCI Bus Master DMA
Support transfers of more than 512 bytes on any given DRQ Interrupt.
Allow error reporting after transfer on reads.
Allow for a minimum queue depth of 64.
Allow the drive to release the Task File Registers before command completion.
Support only simple queued commands.
Minimize Host polling requirements.

2.2 Other Possible Areas for Improvement

There are some other important improvements that Enhanced IDE 95 should address. Although these are important aspects of EIDE 95, this proposal goes no further than to point out the need.

One Interrupt for both Primary and Secondary Channels.
Parity on Data Transfers.
Reduced Command Set.
3.3v Interface.

3.0 Marketing considerations

Some would say that all these changes are not needed, after all there is the SCSI interface. Some would also say that these changes are producing an IDE interface that is no different in complexity from SCSI.

We all should realize the mistakes were made with SCSI. You must also realize that IDE runs to a different model. We would not use the comparison Simple versus Complex, instead we would use: More Control and Cheap versus Less Control and not so Cheap. Clearly IDE has always been and always will be much closer to an implementable standard due to it being a register based interface, whereas SCSI is a protocol based interface.

3.1 History

If you take a step back and realize that any interface is comprised of only four elements:

Physical:	Connectors
Electrical:	Signals, Voltages, Timings
Protocol:	Sequence, Bits, Registers
Commands:	Operations

If you apply this model against the 10 years since the original PC/AT hard disk controller shipped and then transitioned to IDE, you will see that progress had occurred where the PC market needed it to.

3.1.1 Physical

IDE has 3 physical interfaces:

- 3.5": 40 pin I/O connector, 4 pin power
- 2.5": 44 pin I/O that has 40 pins of I/O and 4 of power
- 1.8": 68 pin PCMCIA

3.1.2 Electrical

IDE has had a relatively constant electrical interface with the following areas of growth:

- PDIAG/DASP to address compatibility during transition from controller/ST-506 drives to IDE drives.
- DMARQ and DMACK to support DMA transfers.
- Faster PIO Data Register Timing.
  Mode 0: 600 ns per word - 3.33 Mbyte/sec.
  Mode 1: 383 ns per word - 5.22 Mbyte/sec.
  Mode 2: 240 ns per word - 8.33 Mbyte/sec.
  Mode 3: 180 ns per word - 11.1 Mbyte/sec.
  Mode 4: 120 ns per word - 16.66 Mbyte/sec.
- Faster Multiword DMA Data Register Timing.
  Mode 0: 480 ns per word: 4.16 Mbyte/sec.
  Mode 1: 150 ns per word: 13.33 Mbyte/sec.
  Mode 2: 120 ns per word: 16.66 Mbyte/sec.
- CSEL to ease installation and eliminate jumpers

3.1.3 Protocol

Until ATAPI, IDE has only added more advanced data transfer protocols and had relied upon the Command Register write, BSY bit, DRQ bit, and IRQ protocol for almost everything:

- Multi-sector PIO Read/Write.
- DMA Read/Write.
- Packet Interface. (ATAPI)

3.1.4 Command Set

Since the AT controller first shipped in 1984, command functionality has been added in the areas of:

- IDENTIFY to report capabilities.
- SET FEATURES to enable advanced capabilities.
- Multi-Sector PIO Read/Write.
- DMA Read/Write.
- Power management.
- Download microcode.
- Removability.
- Packet Interface. (ATAPI)
  - CD-ROM command sets.
  - Tape command sets.

3.2 Issues with SCSI

The major issues with SCSI within the PC market were:

- No effort was made to standardize the hardware registers used to integrate it into the PC. What the PC OEMs want are standard register sets, standard command sets, and embedded operating system support. This forced either:
  - Vendor supplied drivers, or
  - Layered device driver model with vendor supplied hardware driver (aka mini-port)
- A command set that was too complex and subject to too much interpretation.
- A protocol that was too complex for single-user machines.
- A cost point that was/is too high.

It should come as no surprise that the industry is trying to take the IDE cost benefits and extend the capabilities. The beauty of IDE is that it defines a register set, a protocol, a command set, an electrical interface, and a physical interface all within one document. PCI Bus Master DMA is obvious, as is moving from the domain of a single threaded channel to a multi-threaded channel to a queuing device.

3.3 Model for enhancing IDE

The obvious model is:

- Power up to Legacy/Compatibility mode.
- Report advanced capabilities via Identify Drive.

- Have the OS driver enable the advanced capabilities via Set Features command or use arguments to individual enhanced capability commands.
- Have the OS driver communicate with the IDE device via the advanced capabilities protocol/commands.

3.3.1 Timing Control

We all must admit that none of us like the BIOS controlled local bus IDE hardware, but a standard register set would have been impossible due to these devices running at different clock frequencies and the requirement that they be programmed to support fixed Mode 0/1/2/3/4 PIO IDE timings. Fortunately, PCI DMA model is 100% register standard, thus this will be easier for all.

3.4 ATAPI Disk

As long as IDE retains a single register set, it makes embedded OS integration much easier. With the advent of ATAPI compliant IDE CD-ROM, ATAPI compliant IDE Tape, and ATA compliant ATA disk, it seems natural for the following to occur:

- Definition of ATAPI compliant IDE disk
- Move IDE support into the layered device driver model, with ATA/ATAPI handled via a mini-port. This allows the OS to support Disk, Tape, and CD-ROM via a single TSD for SCSI or IDE.

3.5 Focus for this Proposal

Our impression is that PCI has killed VL.

Intel worked to get SFF to accept PCI Bus Master DMA IDE register level specification. The goal is a common register set in order to facilitate embedded operating system (i.e. MS) support. Why? If $200 CPU uses 10% of bandwidth for IDE PIO data transfer, then this cost is $20. Multimedia performance is shown to be directly related to amount of CPU bandwidth that can be allocated to it. DMA frees the CPU to do multimedia better.

PCI chipsets today are high pin-count, thus the die sizes are pad limited. This gives them lots of logic to work with as long as the function doesn't add pins. Lets just say that Intel is setting the PCI chipset feature bar and that their 486 and Pentium PCI chipsets have embedded EIDE in the CPU to PCI bridge. The design multiplexes PCI and IDE onto the same pins, thus it costs no pins, and uses up the excess die size...it is free! Unfortunately, SCSI can't be done the same way (i.e more gates, more pins) and I don't see it ever getting integrated into the chipset for this reason and the reason of embedded software support (i.e. Chicken/Egg situation... chipset vendor would want embedded OS support. Which means they would have to license architecture/design from existing SCSI LSI supplier. The Intel Pentium chipset includes the PCI Bus Master DMA IDE capability.

Additionally, the world is trying to add IDE capabilities to issue one I/O per IDE device, not one I/O per IDE channel. The key issues have to do with philosophy (some now, some later versus all later) and where to arbitrate (host hardware/software versus peripheral side) for control of the IDE task file, the IDE IRQ and the IDE DMARQ/DMACK. This topic has started within a working group of ATA (ATA-3). Basically there is a WD proposal that takes a phased approach and has host arbitration and a Quantum approach which takes a Big Bang approach and has peripheral arbitration.

- IDE is getting more capable and staying cheap.
- SCSI has become cheaper and easier, but it isn't going to get anywhere near as cheap as IDE since it isn't in the chipsets.
- ATAPI CD-ROM is now 2X and 4X. There are requests for ATAPI HDCD, ATAPI CD-R, and ATAPI changer specs.
- ATAPI Tape is now starting to happen
- PCI Bus Master IDE is starting to happen (one command/channel)

- The future direction for high speed IDE data transfer is DMA.
- A push is starting for Overlapped Commands and Queued Commands, which requires the host DMA engine go along.

4.0 Overview

Rather than delve into the complexities of this proposal, a general overview of the Reasons, Concepts and Specific Proposals is discussed in this section.

4.1 Key Points and General Approach

- Phase in the Changes. Although the changes to the interface should be phased in, the actual definition of them shall occur before any implementations are attempted.

  Start in 1995.
  More advanced capabilities in 1996.
- Provide solutions for the Customers of ANSI and SFF.

  Host Silicon providers (e.g. Intel PCI and DMA)
  BIOS (e.g. Phoenix)
  Drivers (e.g. Microsoft, IBM)
  System Vendors
- Focus on Overlapped operation
- Allow but de-emphasize more advanced Server and Array capabilities
- Obtain OS and Hardware solutions in a timely manor

  Support in Win 95 Release 2.
  Support in the PC'95 and/or PC'96 Hardware Design Guide.

4.2 Assumptions / Constraints used in the generation of this proposal

- Don't alter the key value that IDE currently has (Cost, Performance, True Register Set Standard).
- Phase in the improvements.
- Allow first implementations without any hardware changes.
- Always be backward compatible.
- ATAPI & IDE Features should be implemented identically where possible.
- Improve Performance only where complexity and cost are not increased.
- Overlapped operations will be mainstream, Queuing will not within the 1995-96 timeframe.
- Reducing number of host interrupts will be beneficial.
- System suppliers, Motherboard manufactures, BIOS and OS suppliers will decide what is acceptable and what is not.

4.3 Phased approach

The features should be phased in over one to two years. There are five distinct capabilities. Each of these can be used when all of the previous capabilities are implemented.

- ATAPI Overlapped Commands.

  Allow CD-ROM and Tape Drives to release the Task File Registers and commands to be sent to the faster IDE Disk Drives.
- IDE and ATAPI overlap using PIO and existing DMA.

  Add Overlapped command capability to the IDE Drives.
- New Overlapped PCI DMA Capability (Reduced Host Interrupts).

  Allow each Drive on the IDE cable to use the DMA Controller without any Host (Driver) intervention.
- Command Queuing and Tagging.

  Multiple commands for each Drive on the IDE cable.

- Advanced DMA for Servers and Arrays.
  Allow Tag information to be processed by the DMA Controller.

4.4 Basic Building Blocks

- Arbitration of the Task File Registers.
  - Release of Task File Registers. A new wrinkle in the protocol will allow the peripherals to clear the BSY signal before the completion of the command, thus allowing the registers to be used for sending command(s) to another device.
  - Selection Command (A2h). This allows the registers to be given back to the Device when needed.
- Arbitration of Interrupt Request (INTRQ) line.
  - Shared Interrupts.
  - Service Status bit for detecting who interrupted.
- Overlap Capable PCI DMA Drive.
  - DMA Ready Status.
- Arbitration of DMA Control (DMARQ, DMACK) signals.
  - Overlap PCI DMA Bridge Logic reads & writes Task File Registers directly.
  - Interlock function in PCI DMA Bridge Logic controls access to IDE Registers.
- Command Queuing
  - Communication of Tags (IDE Feature Register / ATAPI Tag Register.)
- Advanced PCI DMA.
  - Pointer to memory block used to select DMA control information for each Tagged command.

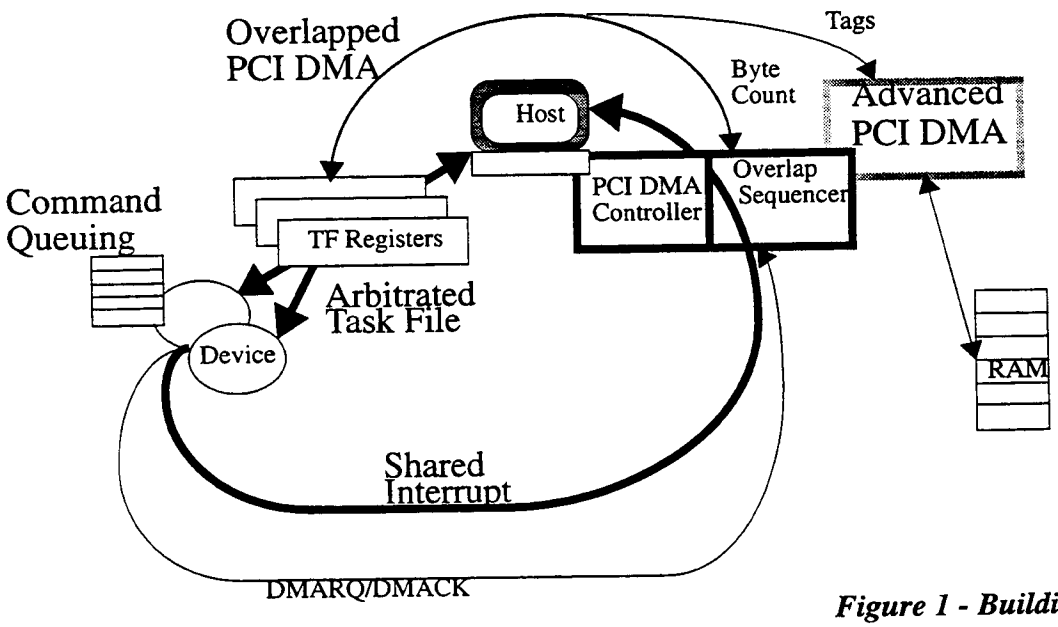

Figure 1 - Building Blocks

4.5 Capabilities

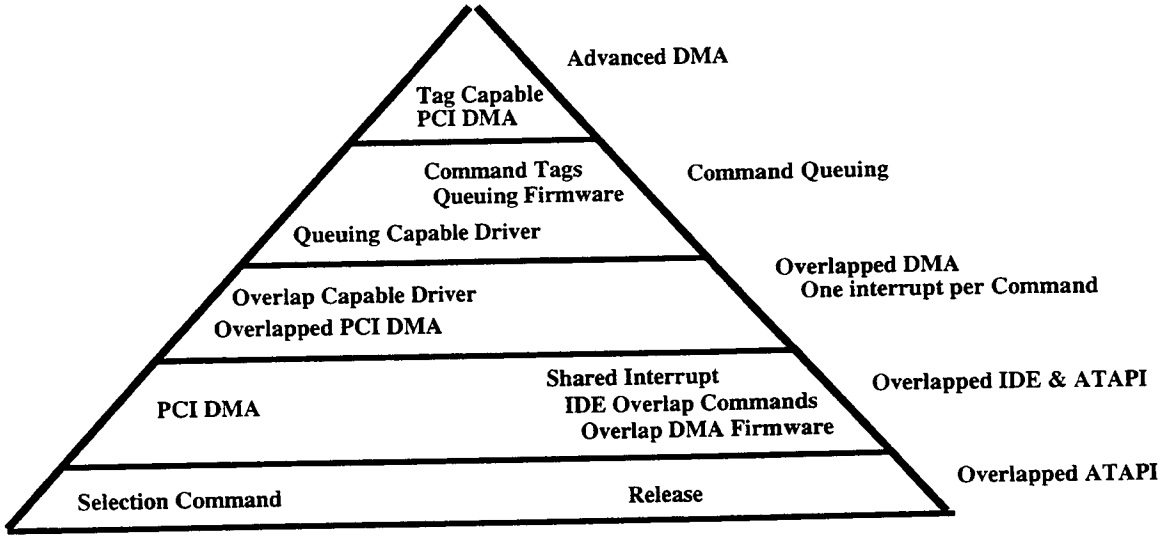

Figure 2 - Pyramid of Capabilities

4.5.1 ATAPI Overlap

- Allows a CD-ROM Drive to be attached to Primary Cable with no Performance Penalty.
- Uses existing Host & Drive (ATAPI) Hardware (No hardware changes needed.)
- ATAPI Drive Releases the Task File Ownership after acceptance of an ATAPI command.
- Overlap Mode is enabled on each command via ATAPI Features Register.
- Overlapped Commands are issued to an IDE (Legacy) Drive while an ATAPI Command is still processing.
- Interrupts are not Shared.
- Drive uses Interrupt & Service Status to gain Host's attention.
- Service Status set when any service is needed by the Device.
- Driver uses the A2h (Select) Command to give control of the Task File Registers back to the Device after an Interrupt and Sensing the Service status bit.
- The Interrupt Reason and DRQ value of IO=1, CoD=1 and DRQ=0 (Message In to Host) is used to indicate a Release Interrupt.
- Use existing PCI DMA.
- When DMA used, the Drive Interrupts when DMA is complete, only when there is still more data to be transferred (That command still has more data to transfer).
- When DMA used, Host (Driver) uses Byte Count from the Selection Command (A2h) results, to program the DMA Controller.

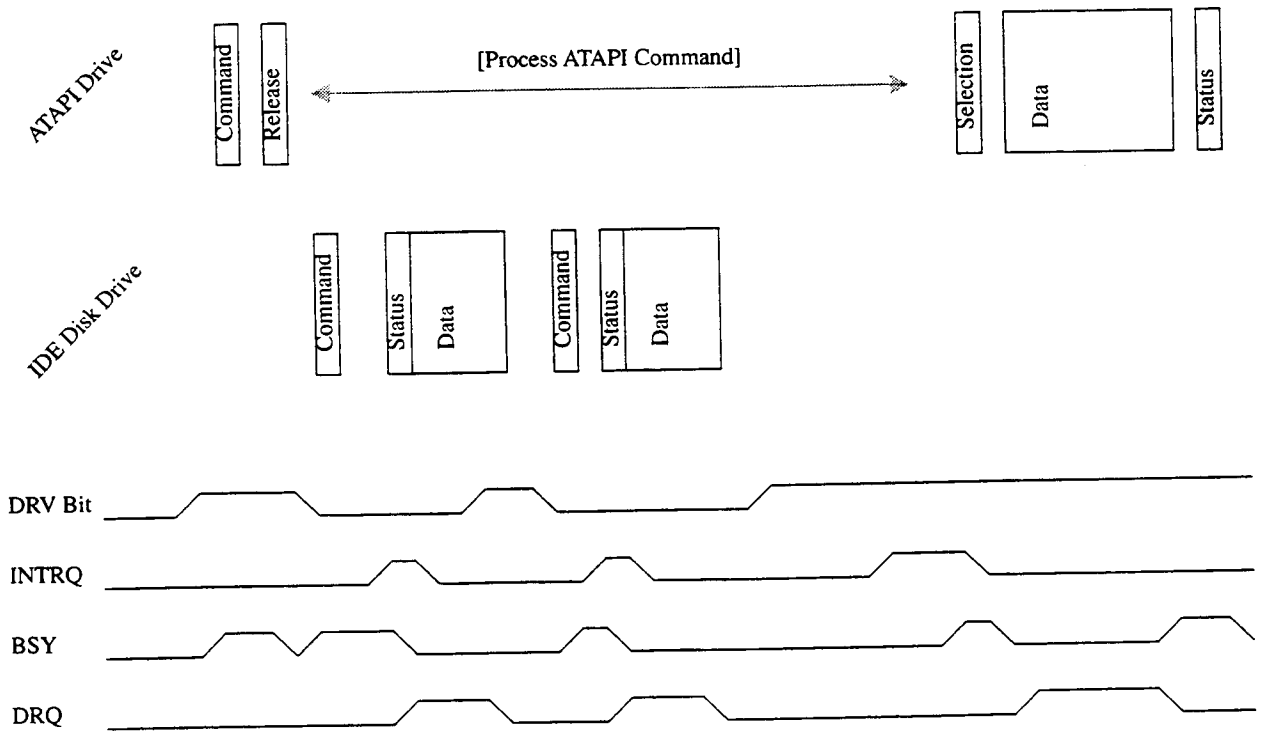

Figure 3 - ATAPI Overlap Sequence

4.5.2 IDE & ATAPI Overlap

- Two new IDE commands for Read and Write Overlapped.
- DMA or PIO specified in the IDE Feature Register.
- IDE Drive Releases Task File after acceptance of Overlap command.
- Task File arbitration performed by the Host (Driver).
- Shared Interrupts used to signal Service to Host.
- nIEN used to prevent interrupts while BSY or DRQ is set.
- IDE INTRQ signal becomes Open Collector.
- Both IDE Drives on the Cable must support the Shared Interrupt for it to be used.
- Overlap can still be used without Shared Interrupt.
- Pulsed Interrupt is used for non-PCI systems (Generates a Rising Edge for normal IDE interrupt logic).
- PCI Systems use a Low Level Interrupt signal.
- Shared Interrupt function is enabled at Post via the SET FEATURES Command.

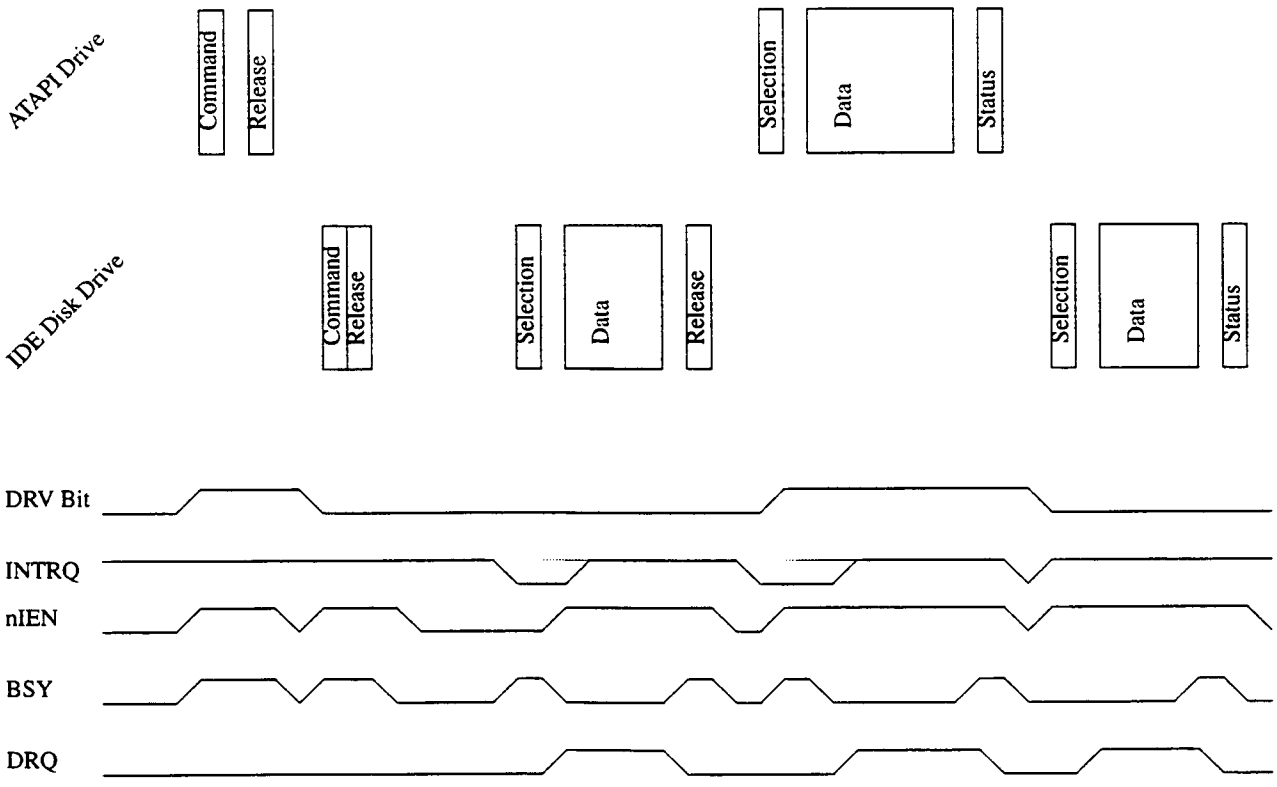

Figure 4 - IDE & ATAPI Overlap Sequence using Shared Interrupts

4.5.3 PCI DMA Overlap Capability

- Intercepts the Shared Interrupt from the IDE/ATAPI Device.
- New Status bit in the IDE Status Register to indicate DMA ready.
- Sequencer selects each Drive, senses DMA Ready & Service status bits.
- Arbitrates and Selects a Drive by issuing A2 command.
- Uses an Interlock to prevent Host (Driver) and Sequencer collisions.
- Blocks Interrupts while processing INTRQ or DMA transfers.
- Interrupts Host for unknown interrupt reasons.
- Host (Driver) Performs same function on systems that do not have the Hardware support for the PCI DMA Overlap.
- No change in the existing Drive DMA or DMARQ/DMACK logic.
- Bridge logic must include control registers for each of the two devices.
- Sequencer in bridge logic reads the Task File Status, Byte Count and Tag registers.

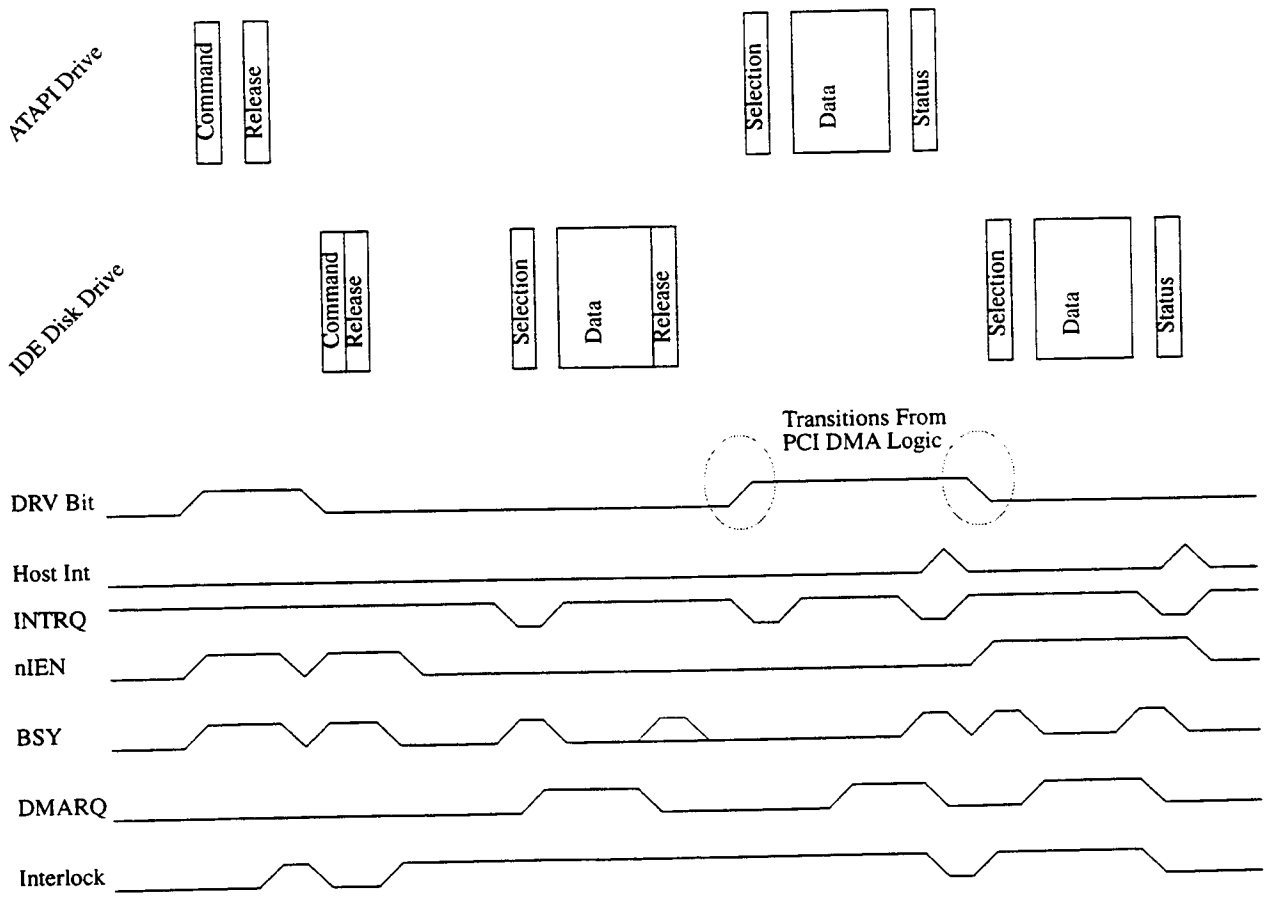

Figure 5 - Overlapped PCI DMA Sequence

4.5.4 Command Queuing

- Tags are used to identify each command
- ATAPI uses the Tag register to communicate the Tag To and From the Device
- IDE Drives use the IDE Feature Register to send the Tag on a command, then use the ATAPI Tag register when asking for service.
- For Both ATAPI and IDE the Tag is placed in the ATAPI Tag register after the A2h Command is issued.
- Simple Queuing only is implemented.
- Errors abort all commands in the Queue.

4.5.5 Advanced DMA

- Layered on Overlapped PCI DMA & Overlapped IDE / ATAPI Command capability.
- Advanced DMA Controller reads Tag from Task File.
- Uses a memory block with one DMA Control entry for each Tag and Device.

5.0 Paths

Although what has been discussed thus far are the goals, they will not be implemented all at once. The goals will be met in groups. These groups represent a phased implementation approach, with less complicated but most beneficial features implemented first. Other more complicated features or those features that require host hardware modification will be phased in over a longer time frame.

Although some features will be phased in, the actual definitions for all the enhanced functions will be standardized at the same time.

A real issue is silicon development cycles, with drive vendors wanting to get this worked out soon as there are new chips on the operating table and we need closure.

5.1 Phase 1: Overlapped Commands to ATAPI Devices

These are the first improvements that will be implemented. It does not require any hardware changes to the Host or the ATAPI Device. This capability will be the foundation for all the other overlap and Queuing capabilities. This phase will allow the OEMs to place one IDE Hard Drive and one ATAPI CDROM on a single IDE Cable, thus reducing the overall cost of the system by at least $0.50 or more.

One Fast Device and one slower ATAPI Device on one IDE Cable.
Doesn't require host LSI changes.
The Host arbitrates the DMA hardware usage by the IDE & ATAPI Drives.
Requires OS Driver changes.
Requires minor ATAPI Device Firmware changes.
Only the ATAPI Device needs to be capable of the Overlap function.

5.2 Phase 2: Overlapped IDE channel & Reduced Host Interrupts for DMA

This is where most of the performance improvements will be created. This phase supports both IDE Disk drives and ATAPI CDROM/Tape. The capabilities added here will form the foundation for Command Queuing and more advanced DMA capabilities. During this phase there will be some skewing in the use of Host Silicon and Devices that can implement the overlap capabilities. Although the full improvement in performance will not be obtained when this occurs, some improvement is still expected.

Mixing of Legacy and Enhanced Drives possible.
Mixing of Non-enhanced Hosts and Enhanced Drives keeps most of the performance gains.
Doesn't require host LSI changes when using existing PCI DMA or PIO.
Reduced Host Interrupts requires new PCI DMA Bridge logic.
The Host arbitrates the DMA hardware usage by the IDE Drives.
Only very minor peripheral LSI changes (Devices share the Interrupt signal.)
Requires OS Driver changes (Should have already been done in the first phase.)
Requires minor IDE Drive Firmware changes (ATAPI Drive changes occurred in phase one.)

5.3 Phase 3: Command Queuing

In this last phase, Queuing of commands in the Faster IDE Disks is implemented. In addition, optional PCI DMA bridge logic could be implemented that will allow the Queued commands to be processed out of order without interrupting the Host Driver when the Tag changes.

All Drives must be Shared Interrupt capable, no mixing allowed.
Optional DMA channel handles Tag and Drive changes automatically (No Driver intervention).
Requires Host LSI changes if Automated DMA is used.

Requires OS driver changes.
Requires Drive Firmware changes.

5.4 OS Improvements

Although not part of any of the above "Phases" the Operating System and its drivers are an integral part of any performance improvements that can be achieved. Once the definition for the Overlap capabilities is complete the net releases should implement ALL the new capabilities. In addition there are other areas that Western Digital believes are important for the Operating Systems to consider.

- Get rid of the monolithic IDE disk/CD-ROM driver and move fully to the NT like layered I/O model.
- Support disk, CD-ROM, and tape TSDs.
- Rename SCSI Manager and call it I/O Manager.
- Run ATA and ATAPI hardware via mini-ports.
- Allow the ASPI interface to communicate with the ATA and ATAPI mini-ports so that CDHD, CR-R, and CD changer can be supported by 3rd party applications.
- Support PCI Bus Master DMA IDE.
- Support for Overlapped IDE channel.
- Support for Command Queuing.

6.0 Key Issues

Before launching into technical discussion of various solutions, it is necessary to provide a foundation of understanding for the reader. This can be thought of as a list of problems, but even more than that, a tutorial.

6.1 Tutorial and Problem Definition

The IDE interface of today has evolved from the simple Winchester controllers, where there was only one set of registers and up to two ST506 or ESDI drives. In this environment there was only one controller for the peripherals and as such the interface was created as "Single Threaded."

Here we find the first of the problem areas for enhancing the interface to provide some level of "Multi-threading" capabilities. Indeed IDE controllers use a protocol to talk with the host, that provides a simple semaphore for locking access to the IDE Interface Control Registers (Task File.) This is made up of the Busy (BSY) signal and the Data Request (DRQ) signals. These signals are used to control "Ownership" of some of the Task File Registers. When BSY or DRQ is set (1) only the peripheral is allowed to Read and/or Write to the controlled registers.

6.1.1 Task File ownership

Logic conventions are: A = signal asserted, N = signal negated, x = does not matter which it is.
Dark Gray are registers where ownership is controlled by BSY & DRQ.
Light Gray are Registers that are not defined for use by IDE.

Addresses					Functions	
CS1FX	CS3FX	DA2	DA1	DA0	Read (DIOR-)	Write (DIOW-)
A	N	0	0	0	Data	
A	N	0	0	1	Error Register	Features
A	N	0	1	0	ATAPI Interrupt Reason / Sector Count	
A	N	0	1	1	Sector Number	
A	N	1	0	0	ATAPI Byte Count LSB / Cylinder Low	
A	N	1	0	1	ATAPI Byte Count MSB / Cylinder High	
A	N	1	1	0	Drive Select	
A	N	1	1	1	Status	Command
N	A	0	0	0	Floppy A Status	Unused
N	A	0	0	1	Floppy B Status	Unused
N	A	0	1	0	Unused	Floppy Digital Output
N	A	0	1	1	Floppy ID / Tape Control	RESERVED
N	A	1	0	0	Floppy Controller Status	RESERVED
N	A	1	0	1	Floppy Data Register	
N	A	1	1	0	Alternate Status	Device Control
N	A	1	1	1	Change / Drive Address	Unused

Table 1 - Registers Controlled by BSY & DRQ

As you can see it is impossible to issue a new command for a number of reasons:

Can't write new command parameters into the Task File while BSY or DRQ is set.
Either BSY or DRQ is always set from the time the command is issued until it is completed.

Changing the DRV bit will cause commands to be aborted in existing IDE Hard Drives.

6.1.2 Sharing of IDE Hardware Signals

Provided that the problems of sending overlapped commands is solved, a new set of issues is exposed. These center around the sharing of signals from the IDE Device to the Host. These signals are used to Generate an Interrupt (INTRQ) and Control DMA operations (DMARQ, DMACK).

6.1.2.1 Sharing INTRQ

For example with one or more commands being processed in more than one device at the same time, when the drive needs to signal to the host that data or status is available, it uses INTRQ. This signal in IDE devices today is driven high when an interrupt is desired. In addition it is only driven by the device that is selected via the DRV bit. Thus there is no way, currently to allow each device to generate an interrupt.

There are two basic approaches to solve this problem. Either the Signal can be inverted and each peripheral can then drive it low (Pulse or Level for PCI) to signal an interrupt, or the devices can decide between themselves which will be allowed to drive the signal. This proposal will use the former as it is believed that the arbitration by the devices themselves is far to complicated for any benefit that is derived. This would also force significant hardware changes in the devices. Note that the other proposals would make use of DASP and PDIAG for device to device arbitration, and this would then prohibit the use of these signals for future enhancements (e.g. Parity.)

6.1.2.2 Sharing DMARQ & DMACK

When it comes to the DMA control signals the problem again can be solved in one of two ways. Unlike the INTRQ signal the DMARQ can not just be driven by each of the devices at the same time. Only one device at a time must use the signal to communicate with the DMA logic on the mother board. In addition the DMA logic must also be kept in sync with the specific location in a specific command from a specific device. The easiest way to do this of course is to use the Host computer to setup the DMA logic and select the device that will be using it.

A more complicated technique would be to program the Host DMA Logic with information on all the outstanding commands. Then to have the devices themselves arbitrate for the use of the DMA control signals, then provide a communication path from the Device to the Host DMA Logic to tell it which device, command and location within the command is currently using the control signals. Western Digital has a real problem with this level of complexity. Adding all this logic to the host DMA will add significant cost and as such will never be "Main Stream." In addition this would not allow a phased approach to implementation, which Western Digital believes is important.

Thus the correct solution is to not change the existing DMA logic in the Devices, and allow either the Host Driver or the PCI DMA Overlap Logic to change the DRV bit.

6.1.3 Other DMA issues

An overall issue of DMA, is that of using the interface for any other operations while the DMA is in progress. DMA makes actual reads and writes to the Data Register of the device performing the DMA. Thus it would be impossible to issue new commands to the device that is performing active DMA. In this proposal the DMA as well as the arbitration is performed by the Overlapped PCI DMA Bridge Logic. Today when DMA is used the command is issued to the device, and no further interrupts are generated until the DMA is complete.

In this proposal the DMA operation will be changed to be much closer to that of the PIO protocol used today. This will add some extra interrupts if the Host does not support the Overlapped PCI DMA Bridge Logic, but will still allow DMA and overlap to be used. This it is believed will still provide significant performance improvements.

6.1.4 Other Problem areas with IDE

Although the Overlapped capability is one of the major areas that need to be addressed, there are other weaknesses as well.

Status is not provided after the data is transferred.
The data transfers are always 512 bytes or some fixed multiple.

7.0 Proposal Details

This proposal makes the attempt to combine discussions of the solutions into just a few categories. In each of these categories all the pertinent information related to the Host Driver, Host Silicon, Drive Firmware and Drive Silicon will be covered. Rather than just cover the device side implementation as is usual for X3T10, all the interrelated functionality needs to be discussed.

This proposal is broken up into four separate capabilities:

- Overlapped Operation
- Shared Interrupt
- Command Queuing
- Advanced DMA.

8.0 Overlapped Operation

This section will discuss the overall capability of "Overlap." This is the foundation for all the other capabilities. There are two very slightly different variations of the Overlap Capability for the Devices. One for IDE and another for ATAPI. In the ATAPI world there are plenty of registers available in the Task File for the Link Layer information. In the IDE world all the bits are dedicated to passing parametric information and as such none (too few) are available to extend the functionality. This proposal requires the Overlap Capability to be enabled on a command by command basis. Although this is strictly not needed for just the simple overlapping of commands, when the Tags are introduced later, this becomes more important. Rather than try to change the method of overlap between simple overlapped and queued commands, this proposal makes them identical with the exception of the use of the Tags when sending the command to the Device.

8.1 Task File Changes for Overlap

8.1.1 Overlap for IDE commands

The overlap is enabled by using two new IDE Commands, READ Tagged and WRITE Tagged commands only. All other commands can not be overlapped. There is no real reason for commands that do not access the media to be overlapped. These commands operate quickly and thus do not need overlapping. If overlapping was to be supported for IDE Commands, then there would need to be a Command by Command enable, so that there would be no backward compatibility issues when dropping back to an "Older Style" driver in sever error conditions. As there is no place in the Task File for an Enable bit, this approach of the two new commands was taken.

8.1.2 Overlap for ATAPI Commands

As has already been discussed, the overlap capability is enabled on a command by command basis. There exists in ATAPI a register dedicated to this function, the Features Register. Thus each command is changed into an overlapped style command via the Overlap Enable bit in the ATAPI Features Register. As this register is used for every ATAPI command, they can all can be overlapped.

Figure 6 - ATAPI Feature Register

D7	D6	D5	D4	D3	D2	D1	D0
Reserved						***Overlapped Command Enable***	DMA Enable

8.1.3 Status register for Overlapped Operation

Figure 7 - Status and Alternate Status Registers for Overlapped Operation

D7	D6	D5	D4	D3	D2	D1	D0	
BSY	DRDY	***DMA Ready***	DSC	DRQ	CORR	***SERVICE***	CHECK or ERROR	Read

Bit 7 BSY — Busy is set whenever the drive has access to the Command Block.

Bit 6 DRDY — Indicates that the drive is capable of responding to a command. This bit shall be cleared at power on. Devices that implement tagged commands shall clear this bit when they are not ready to accept another host command into their

		queue.
Bit 5	***DMA Ready***	***This bit signals that the Device is ready to start a DMA data transfer. This is used to communicate to the Overlap Capable PCI DMA Logic that this service interrupt is going to transfer data via DMA.***
Bit 4	DSC	Seek Complete indication, used for overlapped Seek operation.
Bit 3	DRQ	Data Request - Indicates that the device is ready to transfer a word or byte of data between the host and the drive. The information in the ATAPI Interrupt Reason will also be valid during a Packet Command when the DRQ is set.
Bit 2	CORR	Indicates if a Correctable Error occurred.
Bit 1	***SERVICE***	***This bit signals that the Device is requesting service or interrupt. This bit will be set when the interrupt is requested and not cleared until the SELECTION command is issued.***
Bit 0	CHECK or ERROR	Indicates that an error occurred during execution of the previous command.

8.1.4 Task File Register usage for ATAPI Devices implementing Overlap

Addresses	D7	D6	D5	D4	D3	D2	D1	D0	Registers
1F0									Data
1F1	Sense Key				MCR	ABRT	EOM	ILI	Error Register
							Overlap	DMA	ATAPI Feature Register
1F2							IO	CoD	Interrupt Reason Reg
1F3	*Tag*								Tag Register
1F4	Byte Count that will be used for PIO or for *DMA operations*								Byte Count Low
1F5									Byte Count High
1F6				DRV	Head				Drive Select
1F7	BSY	DRDY	*DMA*	DSC	DRQ	CORR	*Service*	ERR	Status
	0xA0 or 0xA2								Command
3F6	BSY	DRDY	*DMA*	DSC	DRQ	CORR	*Service*	Check	Alternate Status
						SRST	nIEN		Device Control
3F7		L		DEV	Head Selected				Change / Drive Address

Table 2 - ATAPI Overlapped Commands, Register Usage

8.2 Using the Select Command (A2h)

The Arbitration of the Task File Registers is performed by logic outside of the Devices attached to the IDE Cable. The basic premise of this proposal is that the Device releases the use of the Task File Registers when it is processing the command and no longer needs the registers. This of course makes it difficult to place the arguments for the Interrupt into the registers as the device no longer owns them. The Select command essentially hands the registers back to the device so that the correct parameters can be placed into them. These parameters include the Byte Count, Interrupt Reason and Command Tag values.

When an overlapped command requests service the Host Driver or PCI DMA Logic is responsible for determining which device should be serviced, and then issuing the Select. This causes the device to place information on the reason for the service into the Task File registers. Note that for both IDE and ATAPI devices the results of the Select Command are the same.

Addresses	Register	Contents
1F0	Data	
1F1	Error Register	If the Status indicates an Error then this is Valid
1F2	Interrupt Reason	Contains IO and CoD
1F3	Tag for Command	Contains the Tag for the command requiring Service
1F4	ATAPI Byte Count LSB	Number of bytes that need to be transferred, both for PIO or for DMA
1F5	ATAPI Byte Count MSB	
1F6	Drive Select	Same before and after "Select"
1F7	Status	DRQ along with IO and Cod determine the reason for the Service Request
3F0	Floppy A Status	Unused
3F1	Floppy B Status	
3F2	Unused	
3F3	Floppy ID / Tape Control	
3F4	Floppy Controller Status	
3F5	Floppy Data Register	
3F6	Alternate Status	Same as Status register
3F7	Change / Drive Address	Same before and after "Select"

Table 3 - Registers after the Selection Command

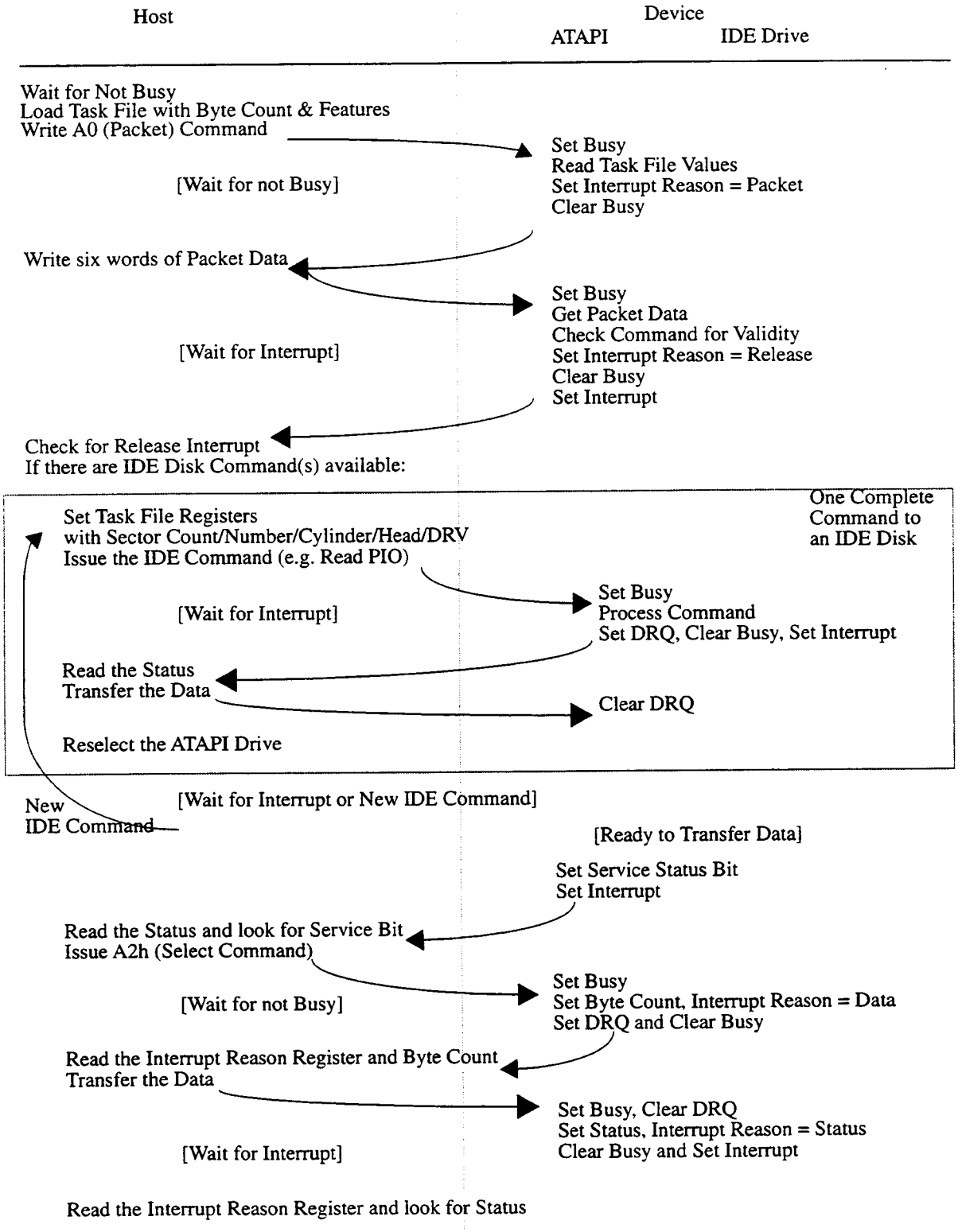

Figure 8 - ATAPI Overlap Flow Diagram

9.0 Shared Interrupts

The sharing of the Hardware IDE signal for requesting an interrupt (INTRQ) to the host is enabled by a SET FEATURES command. Note that this feature must be enabled for both devices on the cable. If either of the devices does not support Sharing, or if the set features command to enable the capability is aborted, the capability must be disabled in both devices.

When Shared Interrupts are enabled the device will invert the INTRQ line and drive it with an open drain driver. When an interrupt is to be generated, the device will pulse the interrupt line low for 1 microsecond or continue to drive it low until the Status Register is read (PCI systems.) In addition when shared interrupts are enabled the SERVICE bit in the Status and Alternate Status registers will indicate that an interrupt was generated. This status bit will remain set until the condition that caused the interrupt is cleared. This could be Sending a Selection Command (A2h) or starting a data transfer for an IDE style command. Because this is a capability that is enabled once, it must be kept backward compatible with existing IDE commands and Drivers. Thus commands that are not using the Selection protocol (Not overlapped) will be using this type of interrupt. This will not affect the interrupt itself, but may impact on the clearing of the SERVICE bit.

9.1 Enabling Enhanced Capabilities

The Set Features command is used to set some interface timing and protocol modes. These modes are set at Post by many BIOSes. The contents of the ATA Features Register indicates the function to be performed.

Table 4 - Contents of the Feature Register for Set Features Command

Bit Byte	7	6	5	4	3	2	1	0
0	Set (1)/ Clear (0) Feature	Reserved			Feature Number			

Feature Number	Set Feature Commands	Support
5Dh	Enable Shared Interrupts (Edge Sensitive system)	Phase 2 and 3
5Eh	Enable Shared Interrupts for a PCI system (Level)	Phase 2 and 3
DDh	Disable Shared Interrupts	Phase 2 and 3

Table 5 - Feature Number Description for Set Feature Command

10.0 Overlapped PCI DMA

The overlapped PCI DMA Logic emulates in a simple form what the Host Driver would do when an interrupt is detected. Thus the IDE or ATAPI Device is unaware of any difference between a driver and the PCI DMA Logic.

To allow the PCI DMA Logic to more easily detect that DMA transfer is required, a new status bit has been added, called DMA Ready. This bit in conjunction with the Service bit and INTRQ signals to the PCI DMA Logic that it should start a DMA Operation for this device. Note that the DMA Logic must have previously been programmed for the transfer.

Use of the DMA Ready status and Service bits is enabled when overlapped Command are issued with the DMA enable bit set in the IDE or ATAPI Feature Registers.

Whenever the PCI DMA Logic senses a condition where it can not handle the interrupt (e.g. No DMA was setup for this device, or the DMA Ready and Service bits were not set) the Interrupt is passed through to the Driver. The Driver is then responsible for providing the same level of functionality as in the PCI DMA Logic. Note also that if the Host doesn't support the Overlapped DMA Capability, then the driver will provide that functionality.

To prevent the Host from accessing the IDE bus while any possible contention could occur, the PCI DMA Logic contains an Interlock mechanism. This is nothing more than a simple semaphore. When the Host commands the PCI DMA Logic to start overlapped operations, the Semaphore is seized by the DMA Logic. When the Driver wishes to gain access to the IDE Bus, it requests that the Semaphore is released and then waits until it is.

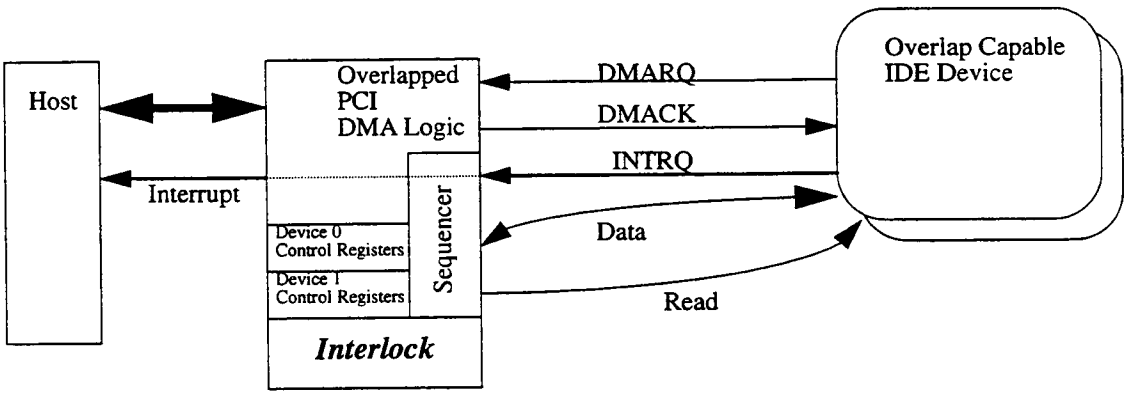

Figure 1 - Overlapped PCI DMA Block Diagram

10.1 Error Handling with Overlapped DMA

Error handling is no different than is used today.

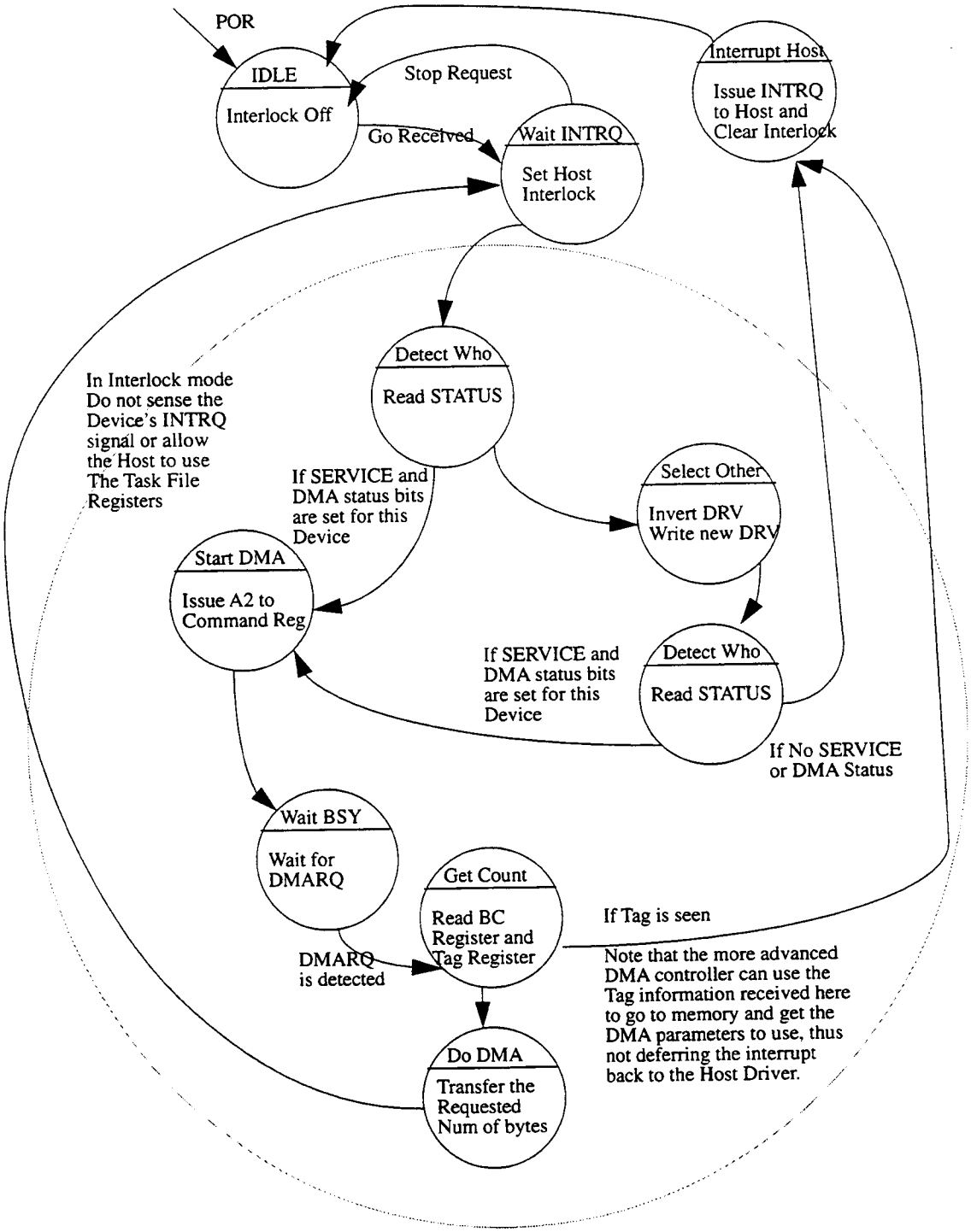

Figure 9 - State Diagram, Overlapped PCI DMA Support

11.0 Command Queuing / Tagging

This document is intended to define the additional functionality required to implement ATA tagged command queuing with minimal impact to backward compatibility with existing ATA BIOS and OS drivers. Adding support for tagged commands provides a simple, backward compatible, method for significantly enhancing the performance of ATA disk drives.

11.1 Performance of Command Queuing

The performance improvements achieved by adding tagged queuing are due primarily to the added ability of the device to receive multiple commands into it's internal queue and then order the execution of these commands to optimize their rotational position on the media.

Because rotational delays are typically two to three times typical seek delays, the ability to allow the device to prioritize commands based on rotational priority can improve drive I/O performance by 50% to 150%.

11.2 Tagged Queuing for IDE

In order to support an IDE backward compatible tagging capability the function of the following bits within the task file have been redefined: DRDY, DSC and Index.

11.3 Tags in IDE

Although the stated goal is for the Tagging capability to be implemented the same for IDE as in ATAPI, the register definitions prevent it. In ATAPI the Tag register overlaps the IDE SECTOR NUMBER Register. Thus the tagging protocol will be the same for IDE and ATAPI, but the placement of the Tag will be slightly different.

For IDE the Tag will be placed into the IDE Features Registers. For ATAPI the TAG Register will be used.

11.4 IDE Tag Registers

Figure 10 - IDE Tag Register

D7	D6	D5	D4	D3	D2	D1	D0	
Tag Value						Reserved	DMA	Write Only

Bit 0 DMA — The DMA bit is used just like the least DMA bit in the ATAPI Feature register. If the DMA bit set (1) then DMA will be used for this command, otherwise normal PIO will be used.

Bits 7-2 TAG — This contains the Tag for the command. This allows for values from 0 - 63. All 64 tag values are legal.

11.5 IDE Style Tagged Commands

As the Tag is supplied in the IDE FEATURE Register, not all the commands could be queued. This forces the decision that not all commands can be queued. There are only two functions that will be able to benefit from queueing, Read and Write. Thus there are two new commands to provide command queuing.

READ TAGGED 0xA6
WRITE TAGGED 0xA7

Addresses	D7	D6	D5	D4	D3	D2	D1	D0	Register
1F0									Data
1F1		UNC	MC	IDNF		ABRT		AMNF	Error Register
	Tag							DMA	IDE Feature Register
1F2	Count								Sector Count
1F3	Sector								Sector Number
1F4	Cylinder								Cylinder Low
1F5									Cylinder High
1F6		L		DRV	Head				Drive Select
1F7	BSY	DRDY	DMA	DSC	DRQ	CORR	Service	ERR	Status
	0xA6 or 0xA7								Command
3F6	BSY	DRDY	DMA	DSC	DRQ	CORR	Service	ERR	Alternate Status
						SRST	nIEN		Device Control
3F7		L		DEV	Head Selected				Change / Drive Address

Table 6 - IDE Overlap Command Register Usage

Addresses	D7	D6	D5	D4	D3	D2	D1	D0	Register
1F0									Data
1F1		UNC	MC	IDNF		ABRT		AMNF	Error Register
	Tag							DMA	IDE Feature Register
1F2							IO	CoD	Interrupt Reason Reg
1F3	Tag								Tag Register
1F4	Byte Count that will be used for PIO or for DMA operations								Byte Count Low
1F5									Byte Count High
1F6		L		DRV	Head				Drive Select
1F7	BSY	DRDY	DMA	DSC	DRQ	CORR	Service	ERR	Status
	0xA6 or 0xA7								Command
3F6	BSY	DRDY	DMA	DSC	DRQ	CORR	Service	ERR	Alternate Status
						SRST	nIEN		Device Control
3F7		L		DEV	Head Selected				Change / Drive Address

Table 7 - IDE Overlap Command Register Usage after A2h (Select) Command

Tagged Commands use the following sequences for normal execution. The host system selects the device and monitors DRDY until the device is ready to accept another command (DRDY=1). The host then sets up the task file for the command including the Tag field. Once the Task file is configured the host writes to the command register to issue the tagged command.

The host may abort a specific tagged command in the peripheral's queue by issuing a new command to the peripheral

with the tag used for the command to be aborted. Peripherals can abort tagged commands using the existing aborted command sequence with the additional requirement that the peripheral identify the aborted command with the command's tag.

11.6 Non Tagged Commands

Upon receipt of a non-tagged command the device shall terminate without status any commands in progress, clear all entries in the devices command queue and immediately process the non tagged command using the non tagged protocol.

Implementors Note: Mixing tagged and non tagged commands in the same command stream can create backward compatibility problems. As a rule the OS drivers supporting tagging should ensure that all commands issued by the tagging driver have completed before releasing control to other drivers in the system which may not be compatible with the tagging protocol.

11.7 Effect of Reset on Queued Commands

Upon receipt of any of the three IDE reset conditions; Power On Reset, assertion of the RESET signal or toggling the SRST bit in the device control register, the device shall terminate the current command without status and clear all entries from the devices tagged command queue.

In the case of SRST the device shall remain configured according to the rules governing the SET FEATURES command.

11.8 Errors and Command Queuing

All errors when Queuing is being used will cause ALL the commands that have yet to be processed to be aborted. The fact that the commands have been aborted is implied in the protocol and no specific status will be generated to indicate that these commands have been aborted. Note that this aborting of all commands on an error applies to ANY fatal or hard error that is detected and reported to the Host. This technique will greatly simplify the error recovery procedures that the Driver and Device will need to implement. This obviates the need for any contingent allegiance conditions or any specific command abort capabilities.

11.9 Advanced DMA

This capability is intended to allow the host to only field one interrupt per command. The Host's DMA controller must be capable of keeping information on ALL active commands for both IDE devices.

11.10 Device to Device Arbitration

There are two possible ways of handling the sharing of the DMA control signals. One would be for the devices to arbitrate for them, the other would be for the devices to signal to the DMA controller that they wish service, and then for the DMA controller to select one for DMA operations.

11.11 Signaling of Drive, Tag and Location in the Data Stream to Host DMA Logic

11.12 Error Handling for Advanced DMA

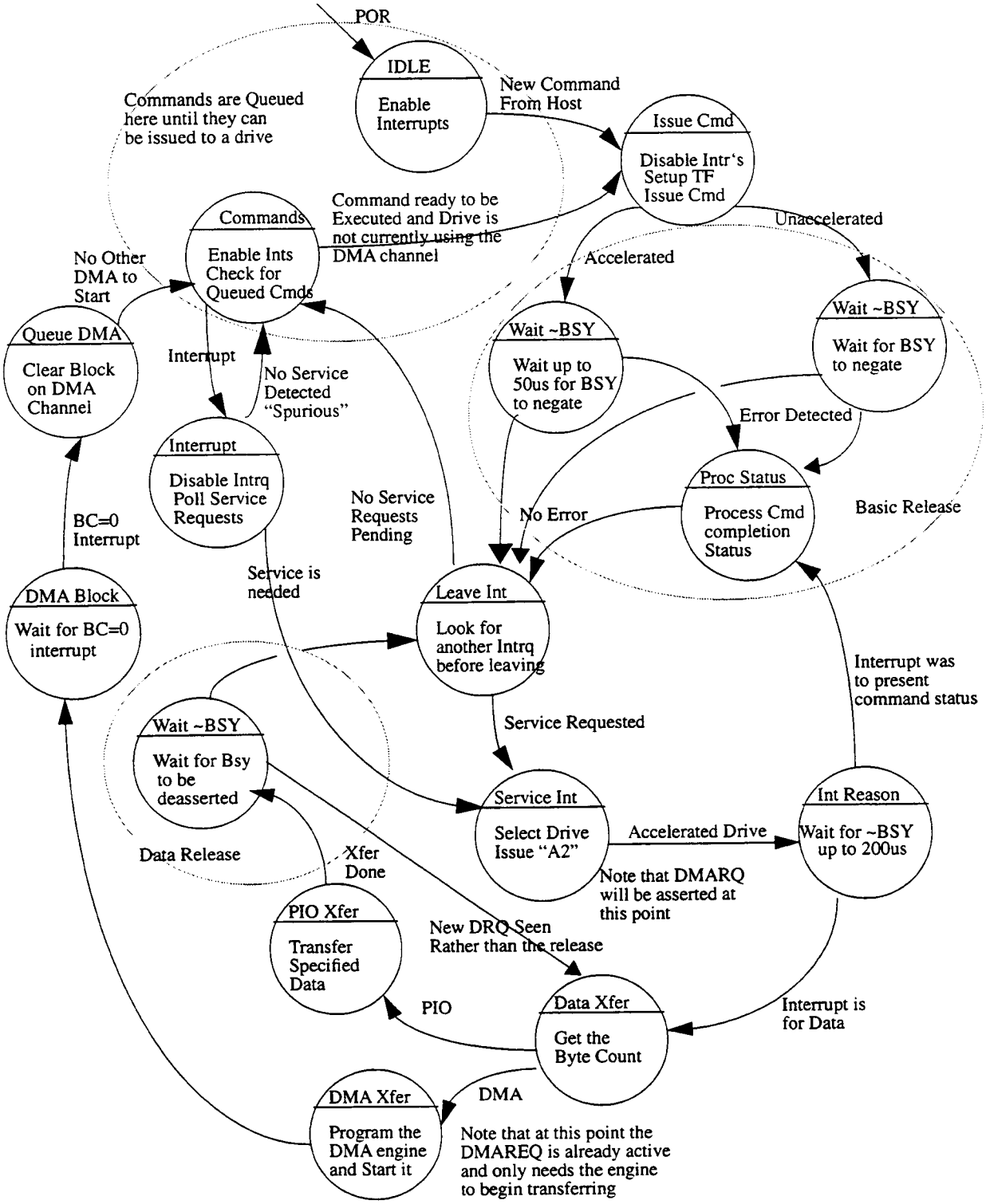

Figure 11 - Host State Diagram, Command Queuing & Shared Interrupts

11.15 Error Handling for Overlapped and Queued Operation

When an error on a command occurs that must report a "Fatal" condition to the host, all queued commands that have not yet been executed will be removed from the command queue and not executed. There will be no errors returned to the host for those commands. This is done to remove the Contingent Allegiance that is used in SCSI to handle getting the Status from the Request Sense. This would not be necessary for Queued IDE commands, but is being done to keep the interface and protocol consistent.

In overlapped operation there will be intermediate command status, as well as the final command completion status. The intermediate status is supplied to indicate if the command was accepted. If the command is not accepted, then there will be no further status supplied. The intermediate status is the status at the point that the device releases the Task File registers back to the host, prior to executing the command. Thus this status can only relate to the validity of the command and not any command execution.

12.0 Comparisons

12.1 Comparisons with Other Proposals

12.2 Shared Interrupt vs. Drive to Drive Arbitration

- Requires very little drive hardware changes.
- Not as complicated as Drive to Drive Arbitration.
- Allows PCI DMA controller to trap interrupt and perform arbitration or pass the interrupt through to the Host Driver when a function is not supported in hardware.
- Reduces communication from the drive to the DMA controller after arbitration.
- Allows some Overlapped commands in a mixed Legacy and Overlap Capable environment.

12.3 New Opcodes vs. using all Existing Opcodes in different "Modes"

- Using new Opcodes prevents any older driver from breaking.
- Allows some redefinition of the Task File Registers.
- Only two commands simplifies the Drive Firmware.
- Enables the Function on a command by command basis automatically.
- Does not break existing prefetch hardware.
- Does not break existing Drive Auto DRQ logic when using Queuing.
- Provides simple Drive Hardware Decode and Sequence logic.

12.4 PCI DMA hardware Arbitration vs. Drive to Drive Arbitration

- Only the Host Hardware changes.
- When the Host Hardware does not support the feature, the Host Driver can provide the capability transparently to the IDE Drive.
- Allows for various levels of performance without changing the Drive Hardware or Firmware.
- Advanced PCI DMA can be implemented without any Drive Hardware Changes.
- No potential race condition on the DMARQ line during arbitration.
- Can be used in mixed Legacy and Enhanced environment.

13.0 Issues still unresolved

There are still several areas that remain unresolved.

- How to generate an interrupt for Non-accelerated devices for the A2 command. Would have to re-enable interrupts in general and could get an interrupt from the other device and not for the A2 completion. If this happened the Task File would still be Busy.