MindShare Technology Series

SATA Storage Technology

Serial ATA

Don Anderson | MindShare, Inc.

MINDSHARE
BRINGING LIFE TO KNOWLEDGE

world-class technical training

Are your company's technical training needs being addressed in the most effective manner?

MindShare has over 25 years experience in conducting technical training on cutting-edge technologies. We understand the challenges companies have when searching for quality, effective training which reduces the students' time away from work and provides cost-effective alternatives. MindShare offers many flexible solutions to meet those needs. Our courses are taught by highly-skilled, enthusiastic, knowledgeable and experienced instructors. We bring life to knowledge through a wide variety of learning methods and delivery options.

training that fits your needs

MindShare recognizes and addresses your company's technical training issues with:

- Scalable cost training
- Just-in-time training
- Training in a classroom, at your cubicle or home office
- Customizable training options
- Overview and advanced topic courses
- Reducing time away from work
- Training delivered effectively globally
- Concurrently delivered multiple-site training

MindShare training courses expand your technical skillset

- PCI Express 2.0®
- Intel Core 2 Processor Architecture
- AMD Opteron Processor Architecture
- Intel 64 and IA-32 Software Architecture
- Intel PC and Chipset Architecture
- PC Virtualization
- USB 2.0
- Wireless USB
- Serial ATA (SATA)
- Serial Attached SCSI (SAS)
- DDR2/DDR3 DRAM Technology
- PC BIOS Firmware
- High-Speed Design
- Windows Internals and Drivers
- Linux Fundamentals

... and many more.

All courses can be customized to meet your group's needs. Detailed course outlines can be found at **www.mindshare.com**

bringing life to knowledge.

real-world tech training put into practice worldwide

*PCI Express ® is a registered trademark of the PCISIG

MindShare Learning Options

MindShare Classroom

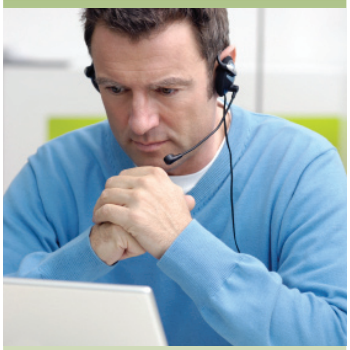
In-House Training

Public Training

Classroom Training

Invite MindShare to train you in-house, or sign-up to attend one of our many public classes held throughout the year and around the world. No more boring classes, the 'MindShare Experience' is sure to keep you engaged.

MindShare Virtual Classroom



Virtual In-House Training

Virtual Public Training

Virtual Classroom Training

The majority of our courses live over the web in an interactive environment with WebEx and a phone bridge. We deliver training cost-effectively across multiple sites and time zones. Imagine being trained in your cubicle or home office and avoiding the hassle of travel. Contact us to attend one of our public virtual classes.

MindShare eLearning

Intro eLearning Modules

Comprehensive eLearning Modules

eLearning Module Training

MindShare is also an eLearning company. Our growing list of interactive eLearning modules include:

- **Intro to Virtualization Technology**
- **Intro to IO Virtualization**
- **Intro to PCI Express 2.0 Updates**
- **PCI Express 2.0**
- **USB 2.0**
- **AMD Opteron Processor Architecture**
- **Virtualization Technology**

...and more

MindShare Press

Books

eBooks

MindShare Press

Purchase our books and eBooks or publish your own content through us. MindShare has authored over 25 books and the list is growing. Let us help make your book project a successful one.

Engage MindShare

Have knowledge that you want to bring to life? MindShare will work with you to "Bring Your Knowledge to Life." Engage us to transform your knowledge and design courses that can be delivered in classroom or virtual classroom settings, create online eLearning modules, or publish a book that you author.

We are proud to be the preferred training provider at an extensive list of clients that include:

ADAPTEC • AMD • AGILENT TECHNOLOGIES • APPLE • BROADCOM • CADENCE • CRAY • CISCO • DELL • FREESCALE
GENERAL DYNAMICS • HP • IBM • KODAK • LSI LOGIC • MOTOROLA • MICROSOFT • NASA • NATIONAL SEMICONDUCTOR
NETAPP • NOKIA • NVIDIA • PLX TECHNOLOGY • QLOGIC • SIEMENS • SUN MICROSYSTEMS SYNOPSYS • TI • UNISYS

MINDSHARE
BRINGING LIFE TO KNOWLEDGE

4285 SLASH PINE DRIVE COLORADO SPRINGS, CO 80908 USA
M 1.602.617.1123 O 1.800.633.1440 ravi@mindshare.com

www.mindshare.com

SATA Storage Technology

MINDSHARE, INC.

Donovan (Don) Anderson

Contributions by:
Mike Jackson
Duncan Penman

MINDSHARE PRESS

Visit MindShare, Inc. on the web: www.mindshare.com

Library of Congress Control Number: 2006934887

MindShare Press
Attn: Maryanne Daves
4285 Slash Pine Dr.
Colorado Springs, CO 80908
Fax: 719-487-1434

Set in 10-point Palatino by MindShare, Inc.

ISBN 978-0-9770878-1-5

First Printing April 2007

About This Book

Part One
SATA Overview

Chapter 1: The Evolution of Parallel ATA

Chapter 2: The Motivation for SATA

Chapter 3: SATA Overview

Chapter 4: Introduction to FIS Transfers

Part Two
FIS Transmission Protocols

Chapter 5: FIS Types and Formats

Chapter 6: Transport and Link Protocols

Chapter 7: FIS Retry

Chapter 8: Data Flow Control

Chapter 9: Physical Layer Functions

Chapter 10: Error Detection and Handling

Part Three
Command and Control Protocols

Chapter 11: The Command Protocol

Chapter 12: Control Protocol

Part Four
SATA II

Chapter 13: SATA II Features

Chapter 14: Native Command Queuing

Chapter 15: Port Multipliers

Chapter 16: Port Selectors

Chapter 17: Enclosure Services

Part Five
Physical Layer Details

Chapter 18: SATA Initialization

Chapter 19: Physical Layer

Chapter 20: Cables & Connectors

Chapter 21: Hot Plug

Chapter 22: Link Power Management

Chapter 23: BIST Features

Appendices

Appendix A: 8b/10b Encoding Tutorial

Appendix B: ATA & SATACommands

1 The Evolution of Parallel ATA

This Chapter

Background information regarding the parallel ATA (PATA) implementations is important for understanding the Serial ATA (SATA) implementation. The SATA design provides compatibility with the PATA legacy programming interface and with the commands defined for PATA. This chapter provides a base level of knowledge for the Parallel ATA implementation and is intended as a review or primer of the technology.

The Next Chapter

The next chapter introduces the motivation for implementing a serial version of ATA. Many recognized shortcomings of PATA are addressed in the SATA implementation. The chapter identifies these shortcomings and describes the solutions provided by SATA.

ATA transitions to Serial Interface

The parallel ATA (PATA) implementation, like many other IO interfaces, has a high-speed serial cousin simply named Serial ATA (SATA). SATA was designed for software compatibility with the ATA/ATAPI 6 specification. The motivation to migrate ATA to a serial interface includes a wide variety of issues, including:

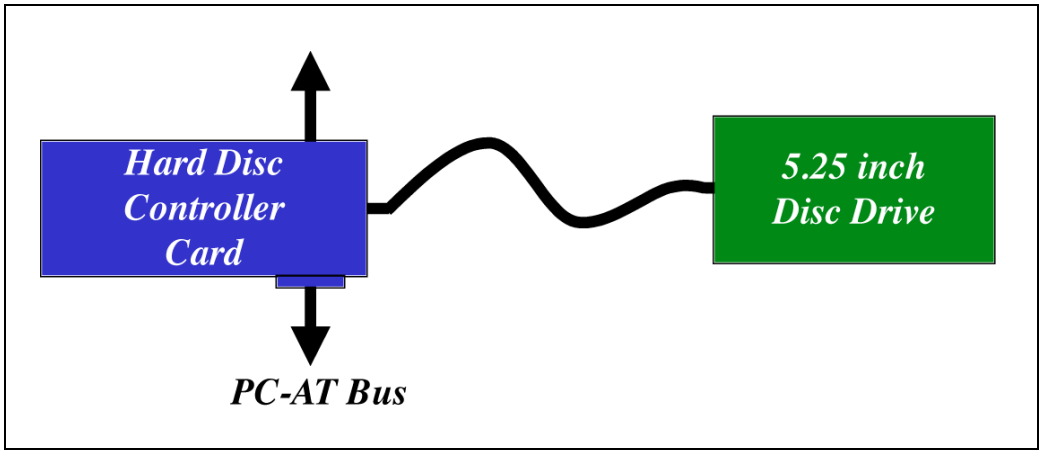
- Cost reduction through reduced pin count
- Lower voltages for smaller silicon sizes
- Improved transmission speed between the drive and Host Adapter
- Enhanced reliability
- Improved cables/connector
- Aggressive positioning of ATA for the server environment

The next chapter discusses the improvements and new features brought about by the SATA implementation.

Origins of ATA

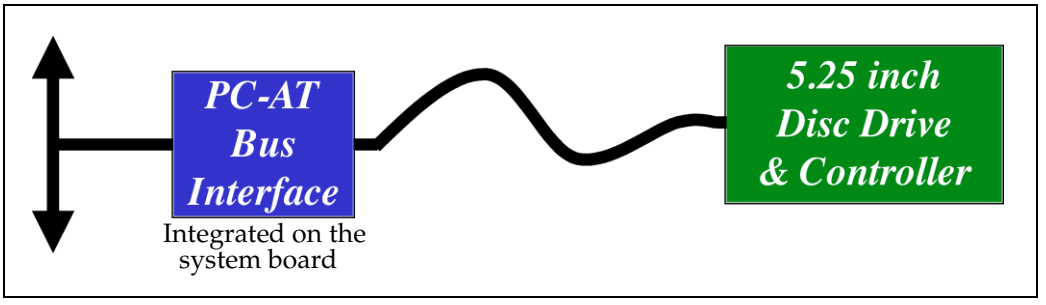
The origins of ATA can be traced back to Compaq Computer in the mid 1980s. At that time, Compaq was principally know for its portable computers. Compaq's first portables were based on the IBM PC and later the IBM PC-AT. Hard disc drives in PCs were initially implemented with a combination of hard disc controller cards that plugged directly into the PC expansion bus and the associated disc drives that were installed into drive bays. Figure 1-1 on page 10 depicts this hard drive controller and hard drive combination.

Figure 1-1: Controller/Disc Drive in Early PC Implementations



Compaq sought to create a portable computer that was smaller and lighter than the original portable (sometimes called luggables), which in some configuration weighed nearly 40 pounds. The Portable II design reduced overall size, in part, by integrating many of the IO functions directly onto the motherboard; thereby, eliminating the need for many expansion cards and associated expansion slots. These integrated functions included the floppy drive interface, serial interface, and parallel interface. Compaq also eliminated the hard disc controller as a plug-in adapter card and mounted it on top of the drive. This implementation of combining the disc controller and drive was called the Compaq Integrated Drive. In doing this, it was necessary to extend the PC-AT expansion bus to the drive bay. This was done via a ribbon cable and a small interface circuit called the Host Bus Adapter. See Figure 1-2 on page 11.

Figure 1-2: PC-AT Host Bus Interface (on motherboard) and Integrated Controller and Drive



The next generation of the Compaq Integrated Drive was fully integrated when the controller was incorporated inside a Wren disc drive housing, manufactured by Imprimis.

Emergence of IDE (Integrated Disc Electronics) Drives

Other hard disc drive vendors began manufacturing integrated drives similar to Compaq's proprietary solution. These disc drives became known as IDE drives. What these initials actually mean is not clear. A few possibilities include:

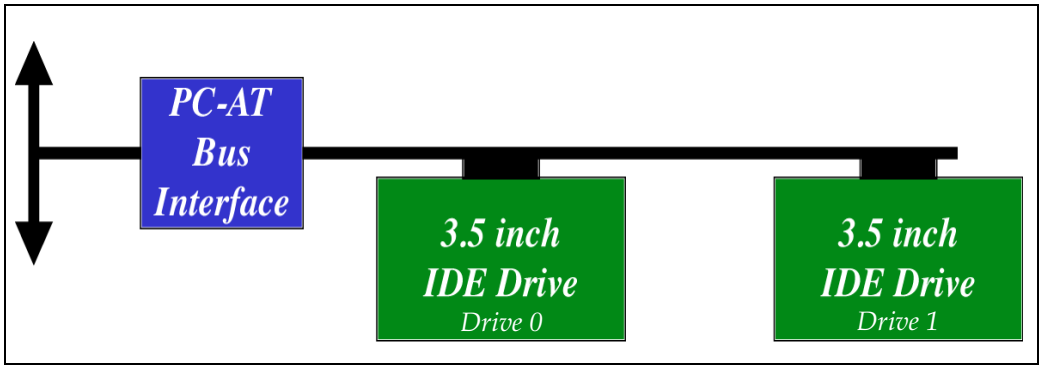
- Integrated Disc Electronics
- Integrated Drive Electronics
- Integrated Drive Element
- Intelligent Drive Electronics

"My personal favorite is Integrated Disc Electronics, but what IDE originally stood for probably falls under the category of "Who cares?"."

Support for Two Drives

Another feature of the IDE drives is the ability to attach and address two devices via a single Host Adapter Interface and shared ribbon cable. See Figure 1-3 on page 12. Note that jumpers or switches on each drive must be set to designate which drive would respond as drive zero (0) and which would respond as drive one (1). See the section entitled, "Device Register - Selecting the Target ATA Device" on page 21 for a description of how these drives are independently addressed.

Figure 1-3: Two Drives Can Share the Same IDE Interface



Compatibility Problems

One of the early problems associated with IDE was that no standard existed for implementing these drives. Consequently there were a number of compatibility problems. Perhaps the most common problems occurred when two drives from different manufacturers were connected to the same IDE cable.

The ATA Standard

Eventually a working group within the Common Access Methods group was formed to define a standard interface specification for the interconnect between the IBM PC-AT bus and IDE drives. The specification was ultimately called AT-Attachment or ATA. The first ATA specification was finally published as:

ANSI (American National Standards Institute) X3.221-1994

Today there is also the ATA working committee named T13 that defines the direction of ATA. T13 is chartered by the:

INCITS - Technical Committee for the InterNational Committee on Information Technology Standards

The ATA Signalling Interface

The ATA interface is represented in block diagram form in Figure 1-4 on page 14. The interface consists of 32 signals, 7 ground pins, and 1 reserved pin position for the key (See Figure 1-5 on page 16). Fundamentally, the ATA Host Bus Adapter (HBA) merely extended the IBM PC-AT expansion bus to the disc drive where the disc controller electronics reside. The interface consists of several major signal groups:

- IO Address-related signals
- Control signals
- Data lines
- DMA-related signals

These signal groups along with the other miscellaneous signals are listed and described in Table 1-1 on page 14. Note that many of the signals are not used in current ATA implementations. Power to the drive is delivered by a separate power connector.

2 The Motivation for SATA

Previous Chapter

Background information regarding the parallel ATA (PATA) implementations is important to understanding the Serial ATA (SATA) implementation. The SATA design provides compatibility with the PATA legacy programming interface and with the commands defined for PATA. The previous chapter provides a base level of information as a review and or a primer on the parallel implementation.

This Chapter

This chapter introduces the motivation for implementing a serial version of ATA. Many recognized shortcomings of PATA are addressed in the SATA implementation. The chapter identified the problems with PATA and described the solution provided by SATA.

The Next Chapter

The next chapter provides a comprehensive overview of the SATA features and protocols.

Motivation and Design Goals for SATA

ATA, like so many other IO interfaces, now has a high-speed serial interface. Many compelling reasons led to this serial interface including the need for:

- Lower pin count
- Higher performance
- Simple drive configuration
- Better cables/connectors
- Greater reliability
- Low voltage support
- Migration to more server implementations

An extremely important aspect of a serial interface implementation is to maintain software compatibility with ATA. The following sections discuss the problems with Parallel ATA (PATA) and how SATA solves these problems, while maintaining software compatibility.

Lower Pin Count

The parallel ATA (PATA) implementation includes a large number of pins as illustrated in Figure 1-5 on page 16 (Chapter 1). Note that some of these pins are not used today or are not required, leaving 26 signal pins that are required. The problems with high pin count are numerous:

- high cost of implementing larger chips to accommodate the high pin count on both drives and HBA
- more board space taken up due to high pin count
- the 40 pin connectors take up much space and are costly
- cables take up more space, are costly, cumbersome to route, and inhibit air flow through a system

In short, many problems associated with PATA result directly from the high pin count. A serial interface with fewer pins solves many of the PATA-related problems. See Chapter 20, entitled "Cables & Connectors," on page 361 for details.

Performance

PATA implementations are limited by the physical connectors and cables originally used in conjunction with the IBM PC-AT bus that operated at transmission rates of 8MB/sec. Today PATA operates at transmission rates of 133MB/sec. To accomplish the higher PATA transmission rates ground pins are inserted between each signal, thereby doubling the connector size to 80 pins. Further increases in performance using the PATA cables and connectors will be more than difficult.

PATA's limited transmission capability is not suitable for the next generation of hard drives that will require much higher transmission rates. SATA's high-speed serial interface operates at 1.5Gbits/sec, which yields a maximum transmission rate of 150MB/sec (Generation 1). The second generation of SATA doubles the transmission rate to 300MB/sec and plans exist to increase performance even further.

No Drive Configuration Required

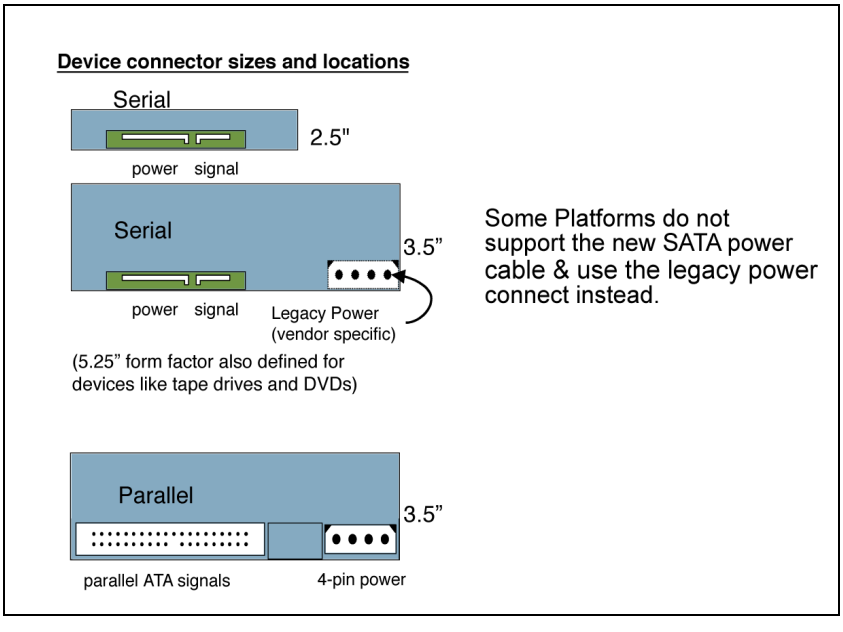
As described in "Support for Two Drives" on page 11, ATA drives required configuration jumpers to support two drives that share the same ATA bus. One drive must be selected as drive zero and the other as drive one. Failure to set the drives correctly typically prevent either drive from working.

There is no need for Drive Select switch/jumper on SATA drives because each drive resides on its own bus. Each SATA drive responds only as drive 0.

Cables and Connectors

PATA cables and connectors have many short comings including: high pin count that adds to their cost, increases space requirements, cables are cumbersome to manage, typical cable length is limited to 18 inches. In addition to these issues, PATA connectors use pin connections that are prone to bending and difficult to plug in the best of conditions. Finally, PATA connectors have no support for hot plugging. Figure 2-1 contrasts the PATA and SATA drives and connectors.

Figure 2-1: Comparison of SATA and PATA Connectors

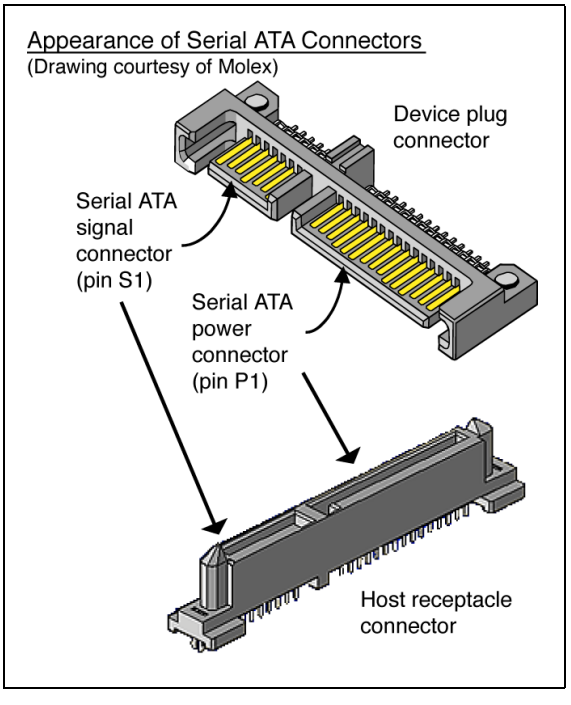


The SATA cable and connectors have the following characteristics that improve upon PATA:

- Signal cable has 7 conductors (two differential pair for noise immunity and three ground pins).
- SATA maximum cable length specified by the 1.0a standard is 1 meter (39.37 inches) compared with 18 inches for PATA.
- SATA cables are smaller, easier to manage, and less costly than the PATA solution.
- Connectors have contacts rather than pins.
- Connectors designed for easy connection, sometimes called blind mating.
- Connectors designed to support Hot Plug (aka, Hot Docking).
- SATA connectors are more easily adapted to backplane implementations

Figure 2-2 illustrates the SATA device (drive) connector and the connector at the Host Adapter. The connections can be made directly such that a drive can be plugged into a bay in backplane fashion or a cable can be routed between the two connectors. See Chapter 20, entitled "Cables & Connectors," on page 361 for additional information.

Figure 2-2: SATA DATA and Power Connectors



Reliability

The ATA bus does not incorporate any native transmission-related error detection and reporting capability. However, the UltraDMA transfers operating at transmission rates of 33Mhz and higher do support CRC error checking, but only during the transmission of data.

In contrast, every packet of information sent across the SATA bus includes a Cyclic Redundancy Check (CRC). These checks are designed to detect every single and double-bit error that occurs. There exists an extremely high probability that CRC will detect virtually every error.

When an error is detected the hardware is designed to perform an automatic retry, by sending the failed packet again. This retry mechanism is supported for all packet types except data packets. This restriction is due to the very large data packet size (up to 8KB) that would need to be buffered at the interface to support the retry mechanism. The large buffer size competes with the goal of keeping drive costs low, and therefore data packet retry is not supported.

Lower Voltages

As silicon manufacturing process sizes became smaller, they also require smaller voltages. Because the ATA signaling was done at 5 volts, it is not suitable for these smaller processes.

In contrast, SATA's signal voltages have been lowered to 0.7 volts. This permits much smaller drive interface chips, as well as reducing power.

Migration to Servers

ATA drives are compelling for server applications primarily due to the low price/megabyte of storage when compared with SCSI and Fibre drives. Many of the shortcomings associated with ATA drives can be mitigated with RAID implementations. RAID can provide additional flexibility, reliability and improve overall performance. The parallel interface limitations of two drives/cable, no hot docking support, and lack of support for other server-related features such as dual ported drives, make ATA server implementations problematic.

3 SATA Overview

Previous Chapter

The previous chapter introduced the motivation for implementing a serial version of ATA. Many recognized shortcomings of PATA were addressed by the SATA implementation. The chapter identified these shortcomings and described the solution provided by SATA.

This Chapter

This overview provides a conceptual understanding of the operation of the SATA interface. This includes the basics of how SATA communication is handled and the primary functions of each layer as defined by the specification. How SATA maintains software compatibility with parallel ATA is also covered in this chapter, along with the new features introduced by the SATA II extension.

The Next Chapter

Communications across the SATA interface consist primarily of transferring Frame Information Structures (FISes). A FIS may deliver shadow or ATA register contents, data, control information, etc. This chapter introduces the reader to the basics of the FIS transmission protocol. Subsequent chapters detail the protocol further and discuss variations due to flow control requirements, errors, and related issues.

The SATA Specification

The SATA specification is developed and maintained by the Serial ATA International Organization. Member companies can download the spec from www.sata-io.org. Non-member companies must pay a fee (currently $25) in order to download a copy.

SATA version 2.6 was the latest version of the specification at the time of printing. The generic terms SATA I and SATA II are used in this book to describe the

features defined by the first and second generations of SATA. It's important to note that while the SATA II features include the 3.0Gb/s transmission rates, drives based on the latest version of the specification can operate at either 1.5Gb/s or 3.0Gb/s. Furthermore, most of the features introduced by SATA II are supported at both transmission rates.

Summary of SATA Features

The last chapter introduced the major shortcomings of ATA and the improvements made by Serial ATA. Table 3-1 contains a more comprehensive list of features associated with the SATA implementations.

Table 3-1: Summary of Major SATA Features

Feature	Description
Serial Interface	The serial interface consists of two differential pairs for bi-directional signaling.
Scalable Performance	Transmission rates of 1.5Gb/s and 3.0Gb/s.
ATA Commands	Support for existing ATA commands.
Low Voltage Signaling	Signaling is performed at low differential voltages: 500mV peak-to-peak.
Improved Cable & Connectors	Implementation of thin flexible cable with maximum length of 1 meter and connectors that provide blind mating capability.
Error Detection and Retry	Each packet, or more properly, each Frame includes CRC generation and checking. CRC generation covers only the Frame Information Structure. CRC errors detected by the receiver are reported back to the transmitting node, perhaps resulting in an automatic re-transmission of the FIS (Frame Information Structure) via a hardware retry mechanism.
Link Power Management	The SATA serial interface can be placed into low power modes following lengthy periods of idle and can quickly recover to normal power and operation.

Table 3-1: Summary of Major SATA Features

Feature	Description
Hot Plug Support	SATA ports can optionally provide support for hot plugging drives. The SATA connector is designed so that 3 contact points can be implemented (1. ground pins, 2. current-limited power, 3. other power and signals) to support hot plugging drives.
Asynchronous Event Notification	Drives and port interfaces can signal the host of events that take place. As an example, a CD-ROM drive could report a CD ejection event to the host adapter.
Full Support for Native Command Queuing	Detailed protocols and new commands are defined to support Native Command Queuing (NCQ) on SATA II drives and hosts. NCQ permits drives to queue up to 32 commands and execute them in the order that will result in the best overall performance.
Port Multipliers	These devices expand the number of ports for attaching and accessing larger numbers of drives via a single HBA port.
Port Selectors	These devices permit two hosts to gain access to the same port/drive to allow fail-over implementations.
Enclosure Services	SATA defines the devices and structure for managing heat, power, error conditions, etc. related to large drive bays and other enclosure types.

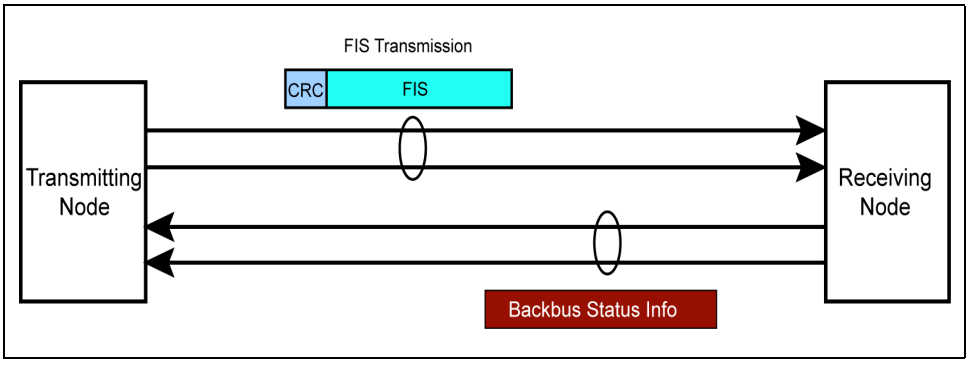
The Serial Interconnect

Serial ATA defines a high-speed serial connection between the Host Adapter and each Drive. The SATA I version of the specification defined the transmission rate of 1.5Gb/s (Generation 1) and the SATA II specification increased the transmission rate to 3Gb/s (providing support for both speeds). The maximum bandwidth is easily calculated because each byte transmitted is a 10-bit value as a result of 8-bit to 10-bit encoding.

Gen 1 speed = 150MB/s
Gen 2 speed = 300MB/s

The SATA interface consists of 2 differential pairs implemented at the HBA port and at the SATA device. However, the SATA implementation is very different from high-speed serial bus implementations that operate in the dual simplex mode where data transfers can be performed in both directions simultaneously. The SATA implementation is based on a half duplex scheme where transfers can be performed in only one direction at a time. SATA implements one differential pair for transmission of data and the other pair for feedback from the receiving device. As illustrated in Figure 3-1 on page 46, the transmitting node (either Host or Device) obtains current transmission status from the receiving node (via the back bus) as the packet (also called a FIS) is delivered.

Figure 3-1: SATA Serial Bus Implementation



SATA Compatibility with Parallel ATA

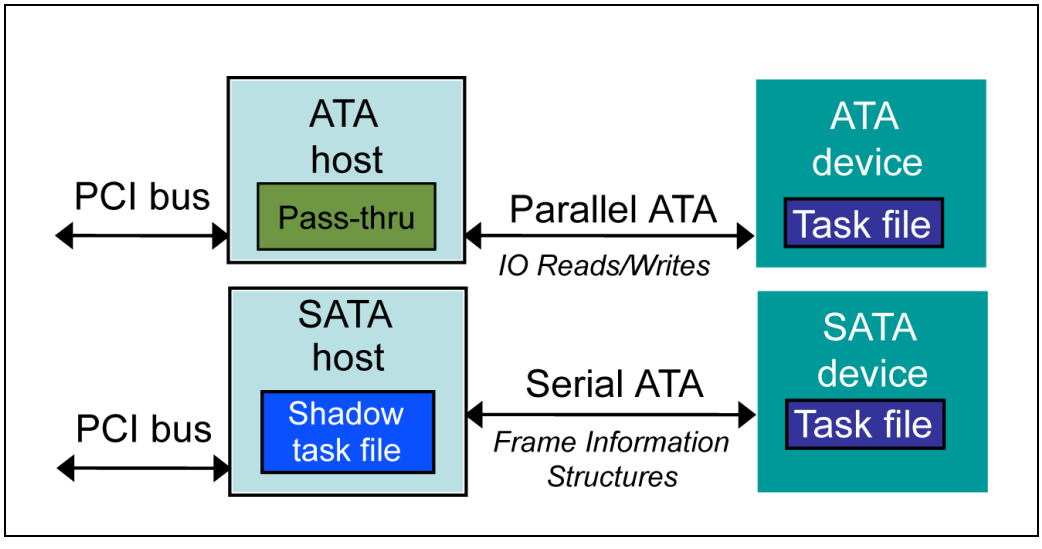
SATA's software compatibility allows an improved hardware implementation without requiring a huge amount of new firmware and software. This goal places the burden on the hardware design to ensure compatible operation with PATA software. The following sections discuss several areas of software compatiblility.

The Legacy Programming Interface

The primary element of software compatibility is of course the ATA programming interface. Figure 3-2 on page 47 conceptualizes the approach taken by SATA to provide legacy programming support versus that of parallel ATA. Recall that ATA host adapters merely forward IO reads and writes, which origi-

nate from the CPU, across the ATA cable to the drive's registers (task file). In SATA implementations, a duplicate copy of the Task File, called "shadow registers," is located in the host adapter. Once software has loaded the shadow register, the HBA sends the register contents to the drive via a packet called a "Frame Information Structure" or FIS. All information exchanged between the SATA HBA and SATA drive is done serially via these Frame Information Structures.

Figure 3-2: Task File Implementations ATA vs. SATA



Refer to Figure 3-3 on page 48 for a more detailed view of the HBA and SATA drive implementation. Note that legacy software is unaware that its IO accesses do not directly target the task file within the drive. Consider the normal process of software initializing the shadow registers and issuing an ATA command to the drive for execution. When software issues the command by writing to the command register, this triggers the SATA HBA to perform the following sequence of events:

1. The SATA HBA sends a packet containing the entire contents of the shadow register (called a Register FIS) to the drive via the high-speed serial link.
2. The drive receives the Register FIS and updates its registers with the FIS contents.
3. Next, the drive parses and processes the command.

As the command is processed, the drive and HBA exchange a particular sequence of FISes needed to fulfill the command. This may include transferring

4 Introduction to FIS Transfers

Previous Chapter

The previous chapter provided a conceptual understanding of the operation of the SATA interface and transmission protocol. This included the basics of how SATA communication is handled and the primary functions of each layer defined by the specification. How SATA maintains software compatibility with parallel ATA was also covered in this chapter, along with the new features introduced by the SATA 2.0 extension.

This Chapter

Communications across the SATA interface consists primarily of transferring Frame Information Structures (FISes). A FIS may deliver shadow or ATA register contents, data, control information, etc. This chapter introduces the reader to the basics of the FIS transmission protocol. Subsequent chapters detail the protocol further and discuss variations due to flow control requirements, errors, and related issues.

The Next Chapter

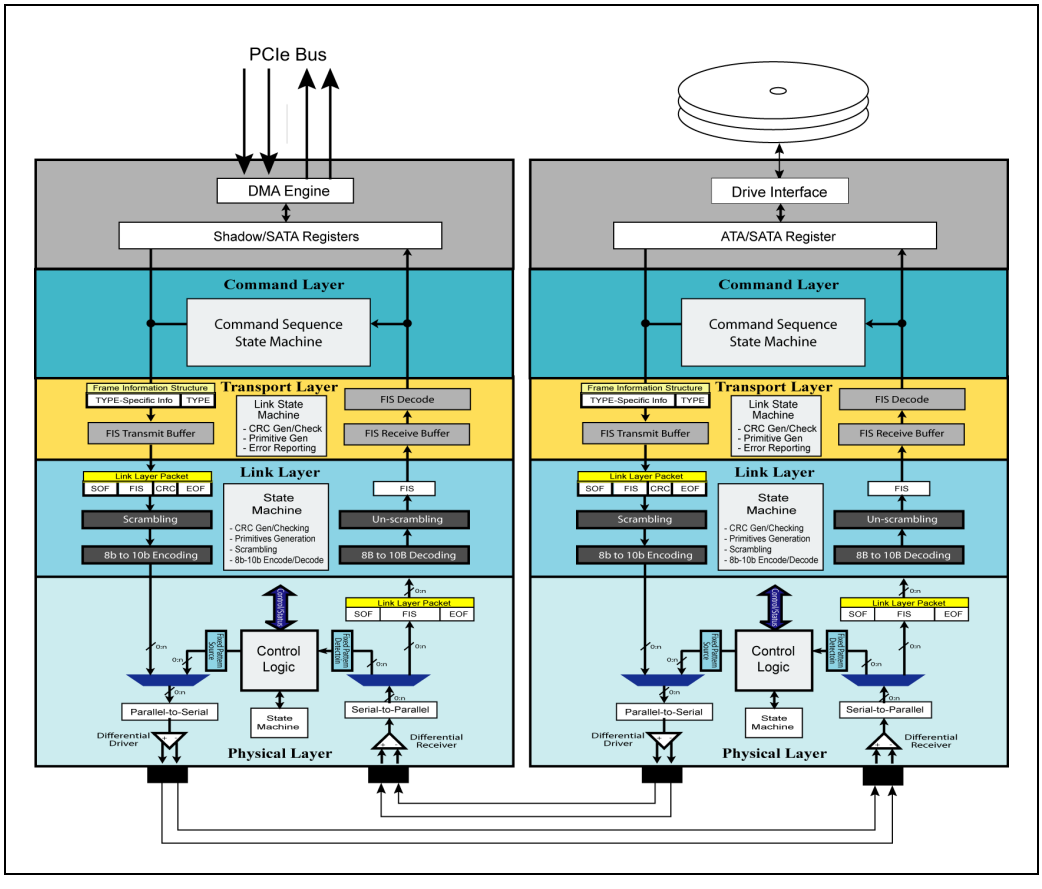
The next chapter is provided primarily for reference purposes. It includes details associated with each of the Frame Information Structures, including definition of all fields within each FIS.

General

Whether a FIS originates at the HBA or a SATA device, the transmission protocol is the same. Figure 4-1 on page 72 illustrates the SATA interface layers in greater detail, showing both the transmit and receive sides. The following discussion describes the actions taken by each layer in the process of sending, receiving and reporting FIS transmission status. The following example

describes a FIS being sent by the HBA as a result of software having issued a command using legacy methods (i.e., multiple IO writes to the HBA shadow registers).

Figure 4-1: FIS Transmission and SATA Layers



FIS Transfers

This section focuses on the transmission of a FIS between the HBA and a SATA drive, and the role that each layer plays in the FIS transfer protocol. The example used for explaining these actions results from host software having issued a Read DMA command. Note that the functions performed by the SATA layers are essentially the same irrespective of whether the HBA or Drive is transmitting the FIS.

Application Layer (HBA)

Host software issues commands to the HBA by performing a series of IO writes required to setup and issue a command. In a SATA HBA these writes target the shadow registers. The actual DMA Read command is issued when host software writes the DMA Read command code to the command register. The SATA specification requires the same behaviors as the Parallel ATA implementations as follows:

- When software writes to the shadow Command register, the HBA must set the BSY status bit within 400ns of the Command register write.
- When BSY (or DRQ) is set software is not permitted to write other shadow registers, including; Features, Sector Count, LBA Low, LBA Middle, LBA High, or Device registers.

The Command register write triggers the delivery of the entire shadow register contents to the Transport layer.

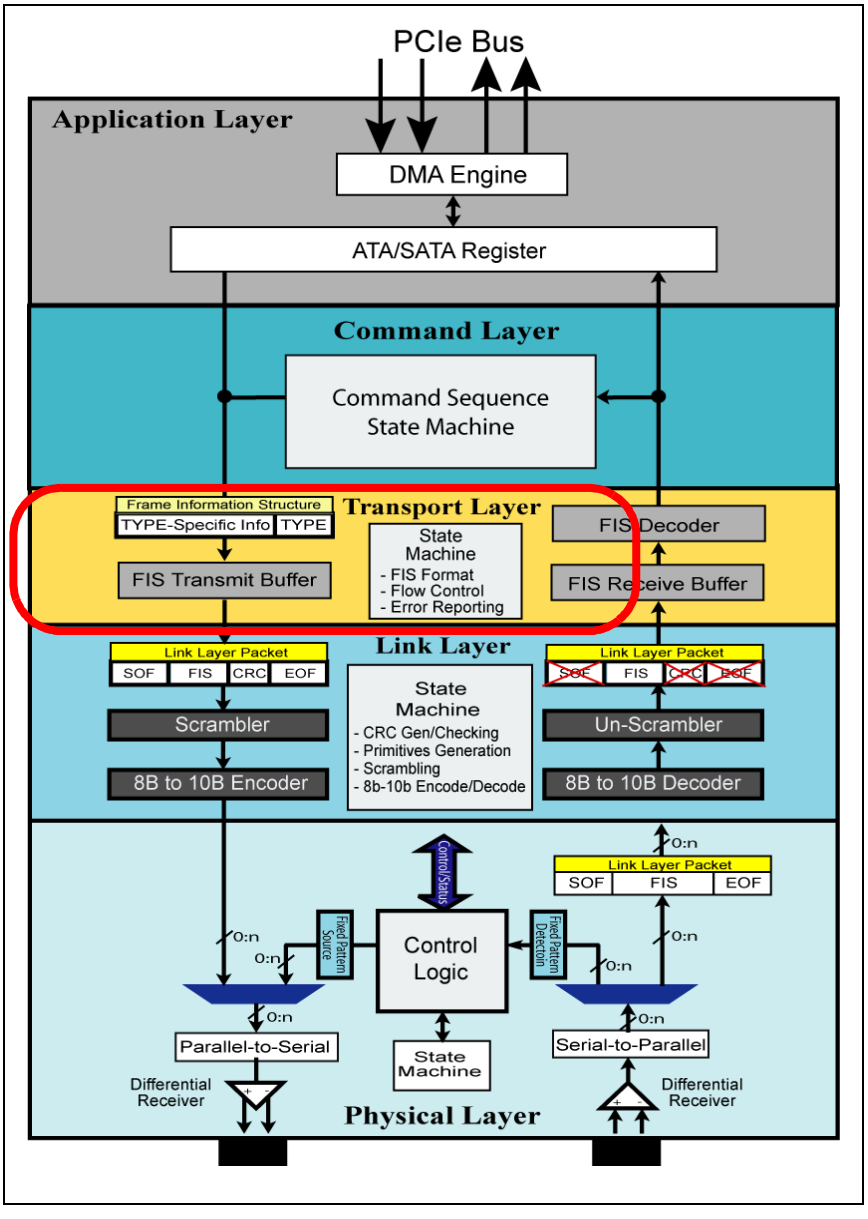
Transport Layer (HBA)

Once software has issued the command (a DMA read in this example), the Transport layer obtains the contents of the shadow register. See Figure 4-2 on page 74. The transport layer has several responsibilities associated with delivering the Register FIS as follows:

- Create a compliant FIS
- Notify Link layer of FIS pending delivery
- Notify link of flow control requirements during transmission

Figure 4-2 on page 74 also shows the presence of a transmit buffer that is intentionally small to reduce the cost of the SATA interface.

Figure 4-2: Transport Layer



Frame Information Structures

SATA uses 12 types of Frame Information Structures to support the sizable number of commands defined by ATA and extended by SATA. Each FIS contains an ID value that identifies the particular FIS being transferred. Table 4-1 lists all of the SATA FISs defined through the 2.0 version of the specification.

Table 4-1: FIS Types

FIS Name	FIS ID
Host to Device Register	27h
Device to Host Register	34h
Set Device Bits	A1h
PIO Setup	5Fh
DMA Activate	39h
First Party DMA Setup	41h
Data	46h
BIST Activate	58h

Note that several of the FIS types are used for more than one purpose. For example, the Set Device Bits FIS is used by SATA drives to modify certain bits within the HBA's Shadow registers, and it's also used for event notification. FIS content and format is described in detail in Chapter 5, entitled "FIS Types and Formats," on page 85.

FIS Ready for Transfer

Upon receiving the FIS transfer request, the Link layer creates a "Transfer Ready" (X_RDY) primitive and forwards it to the physical layer for transmission to the drive. The drive detects the X_RDY primitive and if it's ready to receive a FIS, it returns a "Receive Ready" (R_RDY) primitive to the HBA. Note that these primitives are signaled continuously until it is time to send a different primitive.

5 FIS Types and Formats

Previous Chapter

Communications across the SATA interface consists primarily of transferring Frame Information Structures (FISs). A FIS may deliver shadow or ATA register contents, data, control information, etc. The previous chapter introduces the reader to the basics of the FIS transmission protocol. Subsequent chapters detail the protocol further and discuss variations due to flow control requirements, errors, and related issues.

This Chapter

This chapter is provided primarily for reference purposes. It includes details associated with each of the frame information structures.

The Next Chapter

The link transfer protocol is implemented primarily within the link layer of both the transmitter and receiver. This chapter discusses the individual steps taken by the transport and link layers in transferring a FIS.

General

A variety of characteristics can be associated with each FIS these include:

- Each FIS includes an 8-bit ID value that identifies its type
- FIS size is always multiples of 4 bytes
- Direction of a given FIS may be HBA to Drive, Drive to HBA, or both
- Some FISs have alternate formats that are variations of their standard definition, which are determined by bit fields within the FIS
- Minimum FIS size is 4 bytes
- Maximum FIS size is 8196 bytes

Table 5-1 on page 86 summarizes all the FIS types, including their IDs, size, direction, and generation in which they were introduced.

Table 5-1: FIS Types and Characteristics

FIS Type	ID	Direction	Size	Generation
Register FIS	27h	HBA to Device	5 DWs	Gen 1
Register FIS	34h	Device to HBA	5 DWs	Gen 1
Set Device Bits with Active field with Event Notification field	A1h	Device to HBA	2 DWs	Gen 1 Gen 2 Gen 2
PIO Setup	5Fh	Device to HBA	5 DWs	Gen 1
DMA Activate	39h	Device to HBA	1 DWs	Gen 1
First Party DMA Setup with Auto Activation	41h	Bidirectional	7 DWs	Gen 1 Gen 2
Data	46h	Bidirectional	2049 DWs	Gen 1
BIST Activate	58h	Bidirectional	3 DWs	Gen 1

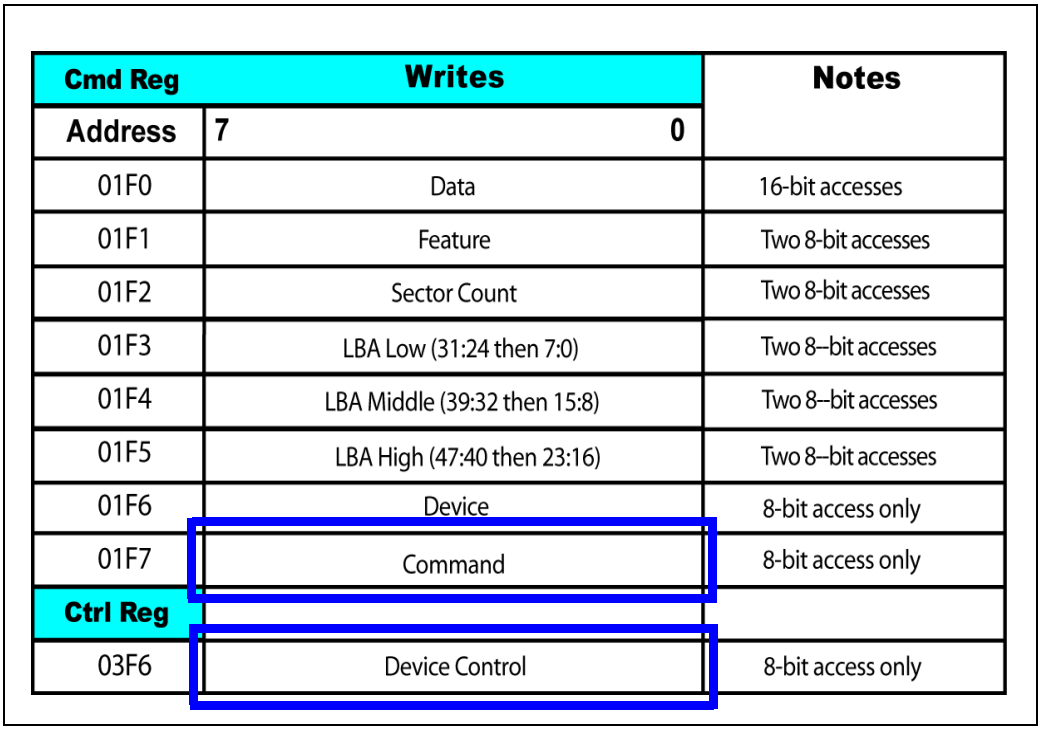
Register FIS - Host to Device

A Register FIS sent to a SATA device always contains the contents of the HBA's shadow registers. Two specific events trigger the HBA to send a Register FIS:

1. A write to the Shadow Command Register
2. A write to the Shadow Control Register

Figure 5-1 on page 87 depicts the shadow register set and highlights the Command and Control registers. The Register FIS contents are a reflection of the values previously written to the shadow register by host software.

Figure 5-1: Shadow Register Definition on Writes (Primary Interface)



Cmd Reg	Writes	Notes
Address	7 0	
01F0	Data	16-bit accesses
01F1	Feature	Two 8-bit accesses
01F2	Sector Count	Two 8-bit accesses
01F3	LBA Low (31:24 then 7:0)	Two 8--bit accesses
01F4	LBA Middle (39:32 then 15:8)	Two 8--bit accesses
01F5	LBA High (47:40 then 23:16)	Two 8--bit accesses
01F6	Device	8-bit access only
01F7	Command	8-bit access only
Ctrl Reg		
03F6	Device Control	8-bit access only

Figure 5-2 on page 88 depicts the contents and format of the Register FIS resulting from a write to the shadow command or control register. Note that the specification defines the fields within the shadow registers using the cylinder, head, sector terminology. Table 5-2 on page 88 describes the contents of each field and specifies the corresponding LBA addresses, where applicable.

Figure 5-2: Register FIS - Host to Device

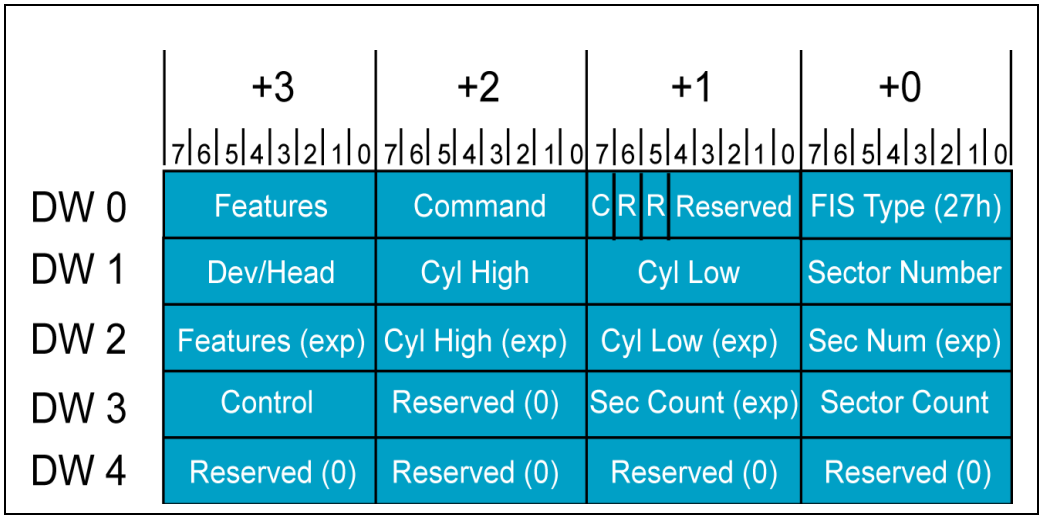


Table 5-2: Register FIS - Host to Device Fields and Descriptions

Field Name	Description
FIS Type	This field contains the ID (27h) of the Register - Host to Device FIS.
C	The “C” bit field (DW 0, offset +1) specifies which event caused the Register FIS delivery. 0 = write to Control register 1 = write to Command register
Command	Contains contents of the shadow Command register that when written causes a Register FIS to be sent to the SATA Device, and the “C” bit to be set.
Features	Contents of the shadow Feature register
Features (expanded)	Contents of the upper 8 bits of the shadow Feature register when using 48-bit addressing.

Table 5-2: Register FIS - Host to Device Fields and Descriptions

Field Name	Description
Sector Number	Contents of the shadow Sector number, or LBA 7:0
Sector Number (expanded)	Contents of the upper 8-bits of the expanded Sector Number value (LBA 15:8) when using 48-bit addressing.
Cylinder Low	Contents of the least significant 8 bits of the Cylinder number register, or LBA 23:16.
Cylinder Low (expanded)	Contents of the most significant 8 bits of the Cylinder number, or LBA 31:24 when using 48-bit addressing.
Cylinder High	Contents of the least significant 8 bits of the Cylinder number, or LBA 39:32
Cylinder High (expanded)	Contents of the most significant 8 bits of the Cylinder High number, or LBA 47:40 when using 48-bit addressing.
Device/Head	This field contains two values: - Bit 4 = the Device Number used to select drive 0 or drive 1 - Bits 3:0 = Head number when software uses the legacy addressing (Cylinder, Head, Sector).
Sector Count	This byte field specifies the number of contiguous sectors to be accessed from the start address on disc using bits 7:0
Sector Count Expanded	This byte field contains the upper bit 15:8 of the sector count when using 48-bit addressing.
Control	Contents of the Control register that when written cause a Register FIS to be sent to the SATA Device and result in the "C" bit to be cleared.
Reserved	The reserved fields all contain zeros.

Register FIS - Device to Host

In SATA, software does not have direct access to the ATA registers located within the drive. When the drive executes a command and updates status information within the ATA registers, the status information is not visible to host software until the shadow registers are updated. Therefore it is necessary for the drive to read it's ATA registers and deliver their contents to the HBA via a Reg-

6 Transport and Link Protocols

Previous Chapter

The previous chapter is primarily for reference purposes. It includes details associated with each of the frame information structures, including definition of all fields within each FIS.

This Chapter

The link transfer protocol is implemented primarily within the link layer of both the transmitter and receiver. This chapter discusses the individual steps taken by the transport and link layers in transferring a FIS.

The Next Chapter

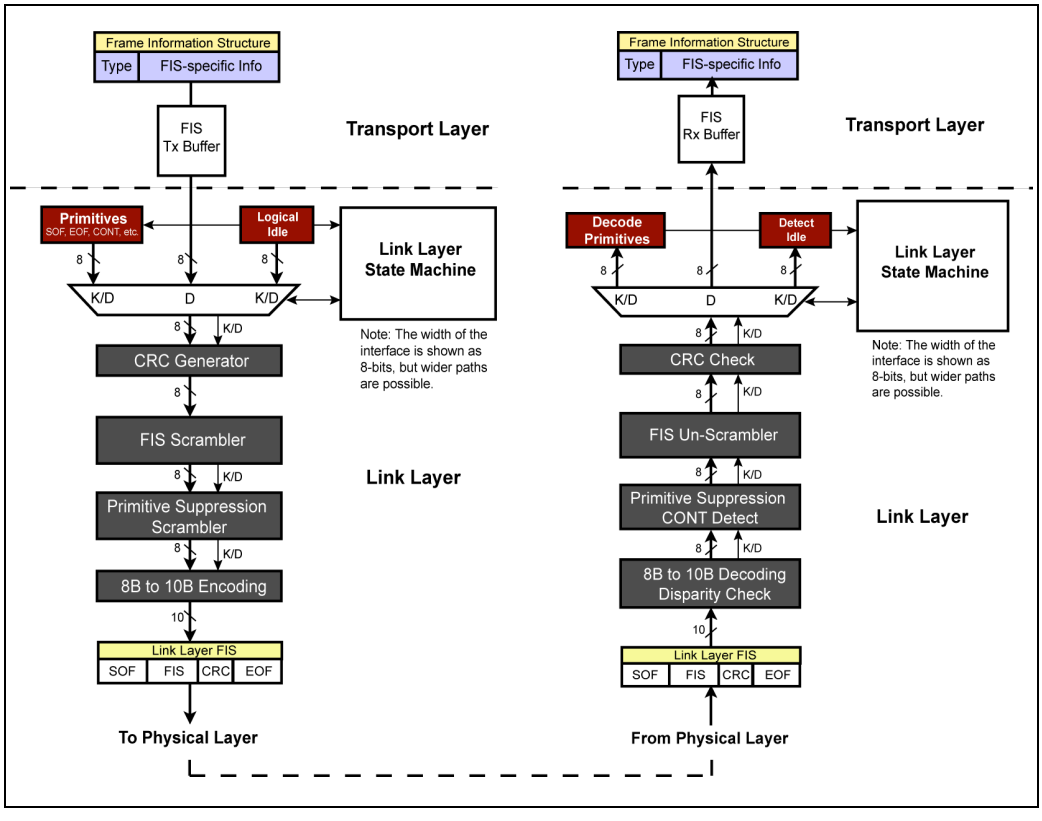
When a FIS is transferred across the SATA link transmission errors are typically detected and reported back to the transmitting device. Upon detection of the transmission error, the Transport layer is notified to the failure. Having retained a copy of the failed FIS, the Transport layer re-transmits the FIS. The next chapter discusses the mechanisms and conditions under which a FIS retry is permitted.

Overview

FIS delivery is most often triggered by software initiating a particular command. Each command consists of a sequence of frame information structures that are exchanged between the HBA and the drive. In some cases the frame information structure originates at the Application layer, while in other cases the frame information structure is determined at the command layer. As has been stated previously, regardless of the origin of a frame information structure the transfer protocol used for the delivery of each FIS is essentially the same.

The transport and link layers work together to create and control the delivery of each FIS. The Transport layer creates and stores each FIS in a transmit buffer and notifies the link that FIS delivery is pending. The Link layer manages most of the transmission protocol through delivery and reception of primitives. Figure 6-1 details the steps performed by the transport and link layers associated with the transmission and reception of each FIS.

Figure 6-1: Transport- and Link-Layer Elements Involved in FIS Transfers



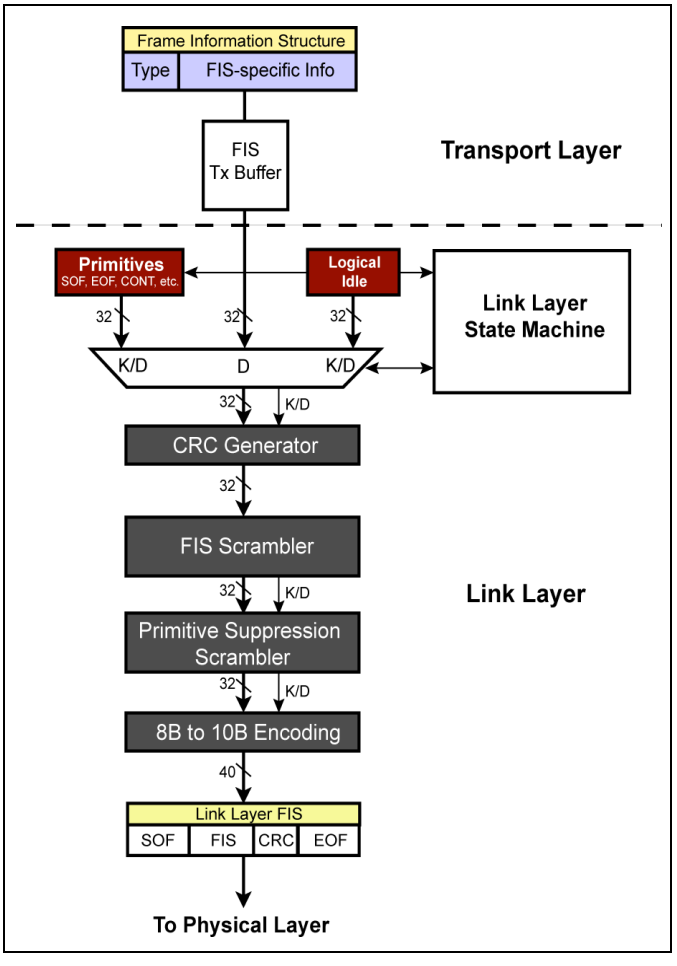
The descriptions in the following sections presume that the FIS transfer occurs successfully. Subsequent chapters detail protocol variations such as FIS transmission retries and flow control.

FIS Transmission

Figure 6-2 on page 106 illustrates the primary elements associated with preparing the FIS for transmission and managing the protocol. The link layer manages event sequences associated with the delivery of each frame information structure. These event sequences involve the generation and reception of primitives, along with preparing each FIS for delivery across the link. This process involves several steps:

- Link Transfer Request/Arbitration
- FIS Transfer (Transport layer to Link layer)
- CRC generation
- Framing each FIS with start and end of frame primitives
- FIS scrambling
- Primitive suppression scrambling
- 8- to 10-bit encoding

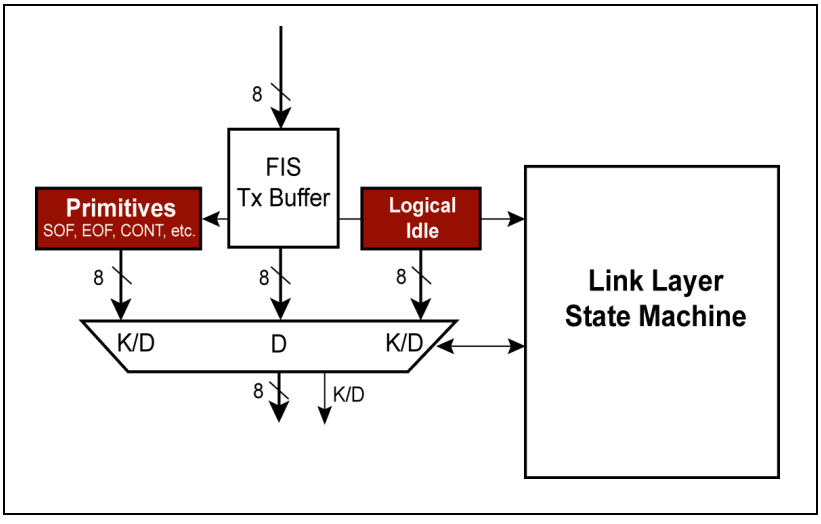
Figure 6-2: Major Transport- and Link-Layer Elements Associated with FIS transmission



Primitive Generation - FIS Transmitter

Primitives originate in the link layer and provide control and status information during FIS transmission and for functions such as link power management. Figure 6-3 on page 107 is a general block diagram that depicts the source of primitives. The primitives like all of the information delivered across the serial link is encoded via the 8-to-10-bit encoder. (See “8b/10b Encoding” on page 116.)

Figure 6-3: Primitive Generation within the Link Layer



Primitives comprise a 10-bit control value followed by 3 consecutive data values. Other architectures use the terminology "ordered set" for this type of sequence. In SATA, once a SATA link is initialized via OOB signaling, it constantly carries primitives and occasionally frame information structures. Figure 6-4 shows the 10-bit encoded values and format of the Sync primitive. The Sync primitive, which is sent repeatedly, indicates an idle condition on the SATA interface and in most implementations is the most prevalent primitive.

Figure 6-4: Content and Format of Sync Primitive

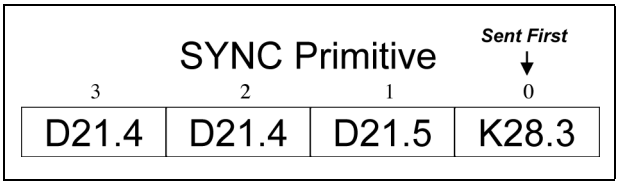


Table 6 - 1 on page 108 lists the primitives that the FIS transmitter sends and describes the purpose of each. Table 6 - 2 on page 119 describes the primitives that are transmitted by a device that is receiving a FIS. Each primitive is discussed in context to FIS transmission in the following sections. For a complete list of primitives along with the values please see Table 23-6 on page 403.

7 FIS Retry

Previous Chapter

The link transfer protocol is implemented primarily within the link layer of both the transmitter and receiver. The previous chapter discussed the individual steps taken by the transport and link layers in transferring a FIS.

This Chapter

When a FIS is transferred across the SATA link, transmission errors are typically detected and reported back to the transmitting device. Upon detection of the transmission error, the Transport layer is notified of the failure. Having retained a copy of the failed FIS, the Transport layer re-transmits the FIS. This chapter discusses the mechanisms and conditions under which a FIS retry is permitted.

The Next Chapter

When performing large data transfers the transmit and receive buffers within the transport layers may either overflow or underflow without some flow control mechanism. The next chapter describes the flow control mechanisms and protocols.

General

In many instances a FIS transmission error can be recovered without notification of software by simply re-sending (e.g., retrying) the FIS. This retry mechanism is managed by the Transport layer and requires the use of buffers that can retain a copy of the FIS after having sent it. Note that retry can be performed by either the HBA or drive.

Which FISes Can be Retried?

A FIS is eligible for retry only if it can be stored in its entirety within the replay buffer. Table 7-1 on page 126 lists each FIS type and its size. The Data FIS is by far the largest with a maximum size of 2049 DWs (8196 bytes). Note that the total size includes only the header and its payload because CRC, SOF and EOF are added by the link layer. The other FIS types have sizes no larger than 7DWs (28 bytes). Due to cost considerations the Transport layer's transmission buffer is not required to store an entire Data FIS, thereby eliminating the Data FIS from the retry protocol. This permits a 28 byte transmission buffer to support FIS retry for all other FIS types, except the BIST type, which is typically used during testing and diagnostics operations.

Without a buffer sufficient to store the maximum-sized data FIS, retry cannot be supported. Transmission of a data FIS results in a change in the host bus adapter's internal state, either through the DMA controller changing its state or through a change in the remaining PIO repetition count, data transmission FIS's shall not be retried.

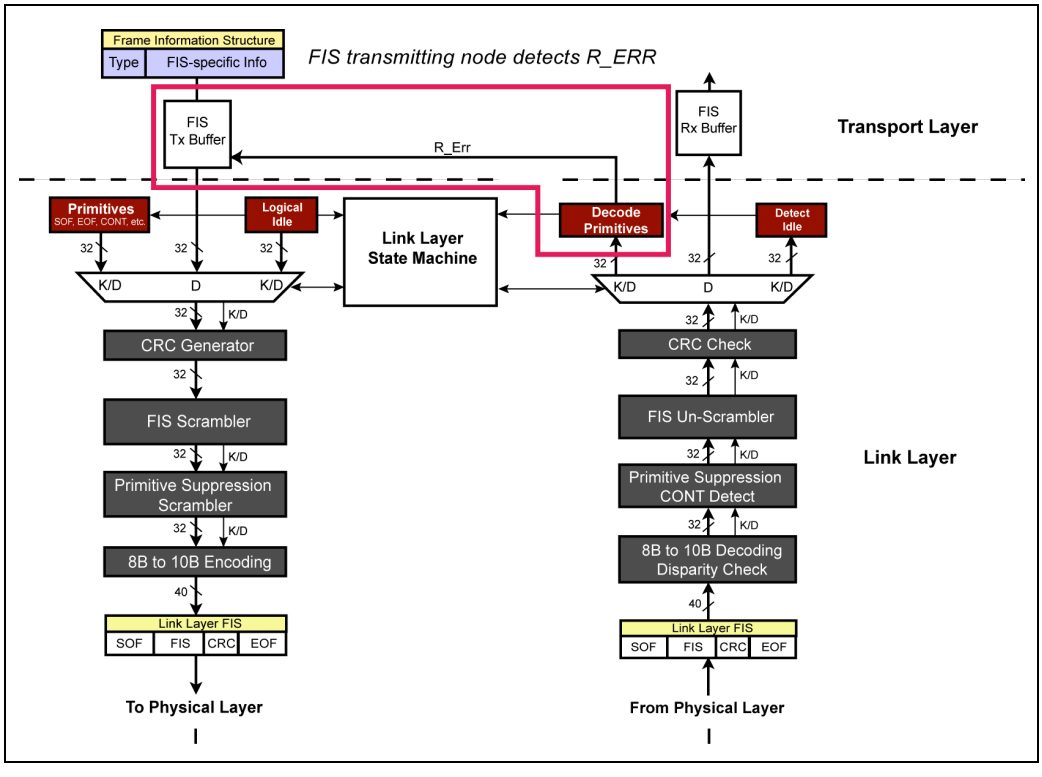
Table 7-1: FIS Type, Size and Retry Eligibility

FIS Type	Size	Retry?
Data	2049 DWs	No
First Party DMA Setup	7 DWs	Yes
Register FIS	5 DWs	Yes
PIO Setup	5 DWs	Yes
BIST Activate	3 DWs	No
Set Device Bits	2 DWs	Yes
DMA Activate	1 DWs	Yes

Retry Protocol Overview

The retry protocol if implemented is triggered by an R_ERR handshake primitive being returned from the receiving device. Figure 7-1 on page 127 illustrates either an HBA or drive receiving an R_ERR primitive indicating that the FIS was received with some type of error condition. In this example the contents of the FIS remains in the Transport layer's transmit buffer so it can be retried. The specification does not limit the number of retries that can be attempted. If the FIS re-transmission fails repeatedly, host software will ultimately time-out, which results typically in the link interface being reset.

Figure 7-1: FIS Retry on Transmission Error



What Errors Result in Retry Attempts?

SATA hardware may optionally be designed to automatically re-transmit a FIS that fails transmission across the link or that has some other problem. In general, errors that are transient in nature are those that qualify for and are likely to be recovered via the retry protocol. For example, errors may be due to transmission problems such as noise or due to power glitches, etc. Such errors are typically detected within the link layer and forwarded to the Transport layer for handling. Other FIS errors may be detected within the Transport layer itself.

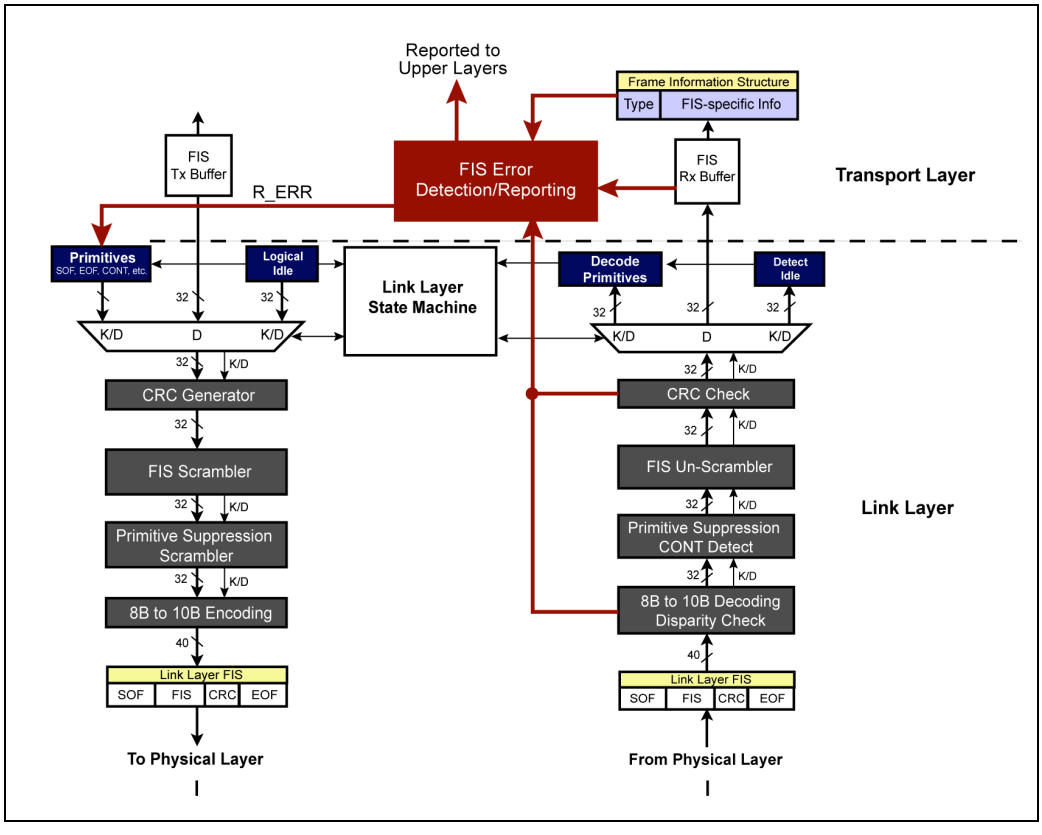
Whether the HBA or the drive transmits a frame, transmission errors are detected by the receiving node and are typically reported back to the transmitting node via the R_ERR primitive. In every case, these link transmission errors are reported in the SATA Error register that resides within the HBA.

Figure 7-2 on page 129 illustrates the Transport and Link layers and the FIS error detection and reporting features. Several error detection mechanisms are used and two error reporting mechanisms are employed:

1. Delivery of R_ERR back to the transmitting node (Both HBA and drive)
2. Notifying upper layers to set the appropriate bits in the SATA Error Register (when the HBA receives an R_ERR primitive from the drive or when the HBA detects an error when receiving a Frame)

In a majority of the frame transmission error cases, an R_ERR primitive is sent to report the error. However, some types of error are not allowed to attempt retry. The various cases are discussed in the following sections.

Figure 7-2: FIS Receiver Error Detection and Reporting



Transmission Errors — Detected by Link Layer

The link layer detects two error types both of which are typically due to link transmission errors:

- CRC Errors
- Disparity Errors

These errors are reported to the Transport layer, which is required to return an R_ERR primitive to the FIS transmitting node. In addition the HBA's Transport layer causes the appropriate bit(s) in the SATA Error register to be updated as indicated in Figure 7-2 (whether acting as transmitting or receiving node). In

8 Data Flow Control

Previous Chapter

The previous chapter discussed the mechanisms and conditions under which a FIS retry is permitted. When a FIS is transferred across the SATA link, transmission errors are typically detected and reported back to the transmitting device. Upon detection of the transmission error, the Transport layer is notified of the failure. Having retained a copy of the failed FIS, the Transport layer re-transmits the FIS.

This Chapter

When performing large data transfers the transmit and receive buffers within the transport layers may either overflow or underflow without some flow control mechanism. This chapter details the mechanism defined for managing flow control for two circumstances. The first condition is when the node transmitting a data FIS is starved for data to send. This buffer dry condition triggers the flow control protocol for the FIS transmitter. The second condition occurs when the receiving node is unable to empty its buffer as quickly as its being filled. The resulting buffer full condition trigger the flow control mechanism from the receiving side.

The Next Chapter

This portion of the book focuses on FIS protocol and correspondingly this discussion of physical layer focuses on the functions related to FIS transmission and reception. Part 5 of this book details the physical layer's role in reset, initialization, Hot Plug, and electrical details.

Overview

One of the major design goals of the SATA implementation is to keep drive design costs low. To this end, the buffers used in the SATA interface are intentionally kept small. As discussed in the previous chapter, transmit and receive buffers within the SATA Transport layer are sized just large enough to support transaction retries of the largest non-data FIS (28 bytes), which is much smaller than the maximum data FIS size (8196 bytes). Consequently, flow control is needed only for data FISs. Figure 8-1 on page 135 illustrates the Transport layer transmit and receive buffers, along with essential link layer elements involved in flow control. Figure 8-1 A illustrates a the buffer nearly dry condition that can occur at the transmit buffer and Figure 8-1 B depicts a receive buffer nearly full condition.

The additional buffers shown below the Transport layer buffers are located in the Link layer. These buffers are typically only one or two DWs in size and are used for handling link-specific delays such as the insertion of primitives into a FIS.

The SATA flow control mechanism prevents Transport layer buffer underruns and overruns. The mechanism uses two primitives to support the protocol:

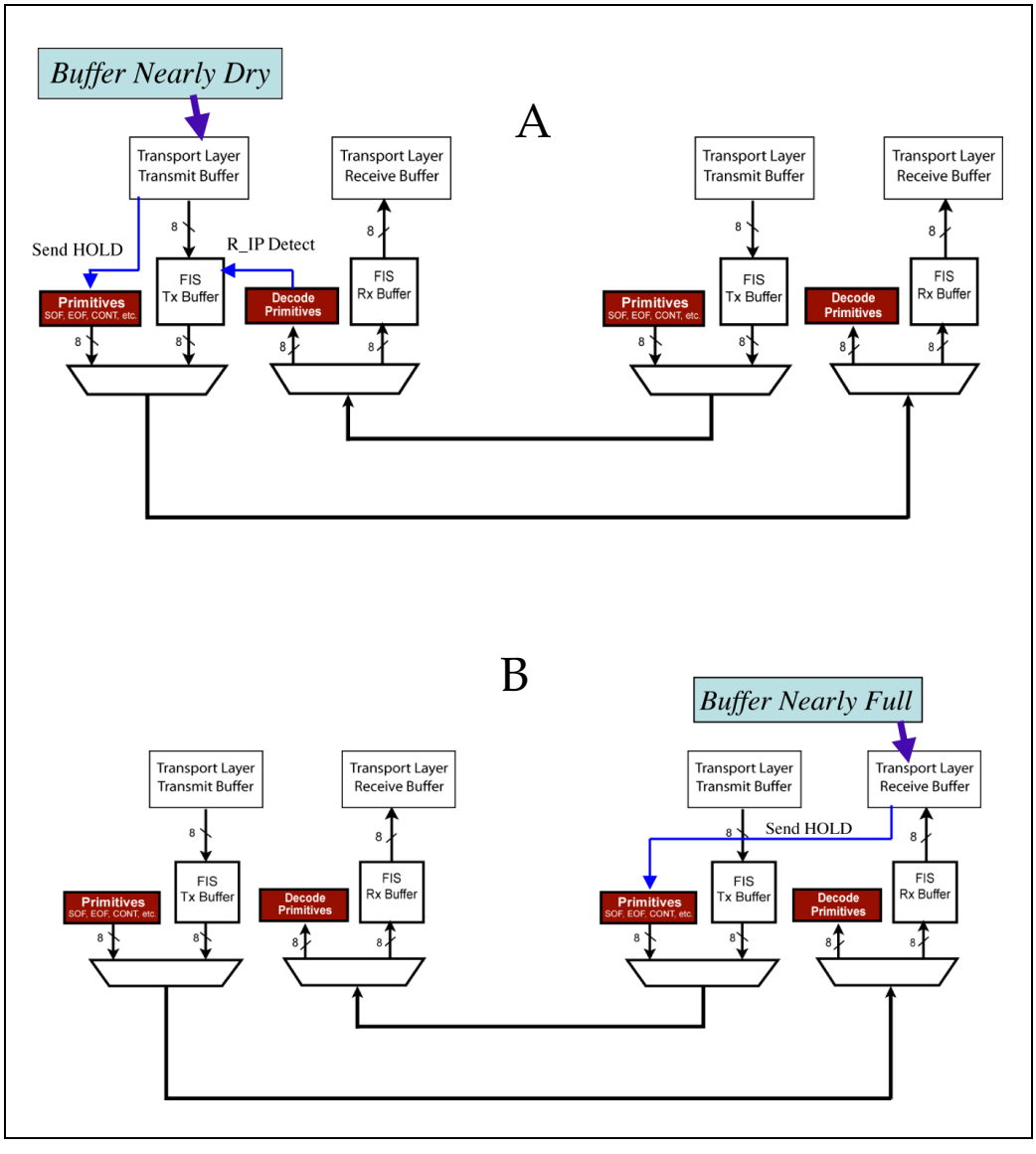
- HOLD — indicates the need to pause FIS delivery
- HOLDA — hold acknowledge verifies the FIS pause

Flow control can be implemented by either the FIS transmitter or receiver. Both cases are discussed in the following sections.

Flow Control by Transmitter

When a node transmits a Data FIS the data may be supplied to the Transport layer at a slow rate than data being delivered from the Transport layer to the node on the opposite end of the link. This condition can cause a potential buffer underrun. For example, during DMA write operations one or more Data FISs are used to deliver data from the HBA to the drive. The data must be fetched from memory by the HBA's DMA engine, which may be required to gather the data from multiple 4KB pages in memory. When the DMA engine delivers the initial 4KB data block the HBA may start transmission of a Data FIS with a payload size of 8KBs. Delays in gaining access to memory can occur as the DMA engine attempts to fetch the next page. This could result in a buffer underrun condition, if the flow control mechanism were not present.

Figure 8-1: Essential Flow Control Elements



Flow Control Protocol (Transmitter Initiated)

This example discusses the protocol used to avoid an underrun at the transmit buffer. The example also presume that primitive suppression scrambling is being used by the transmitting node.

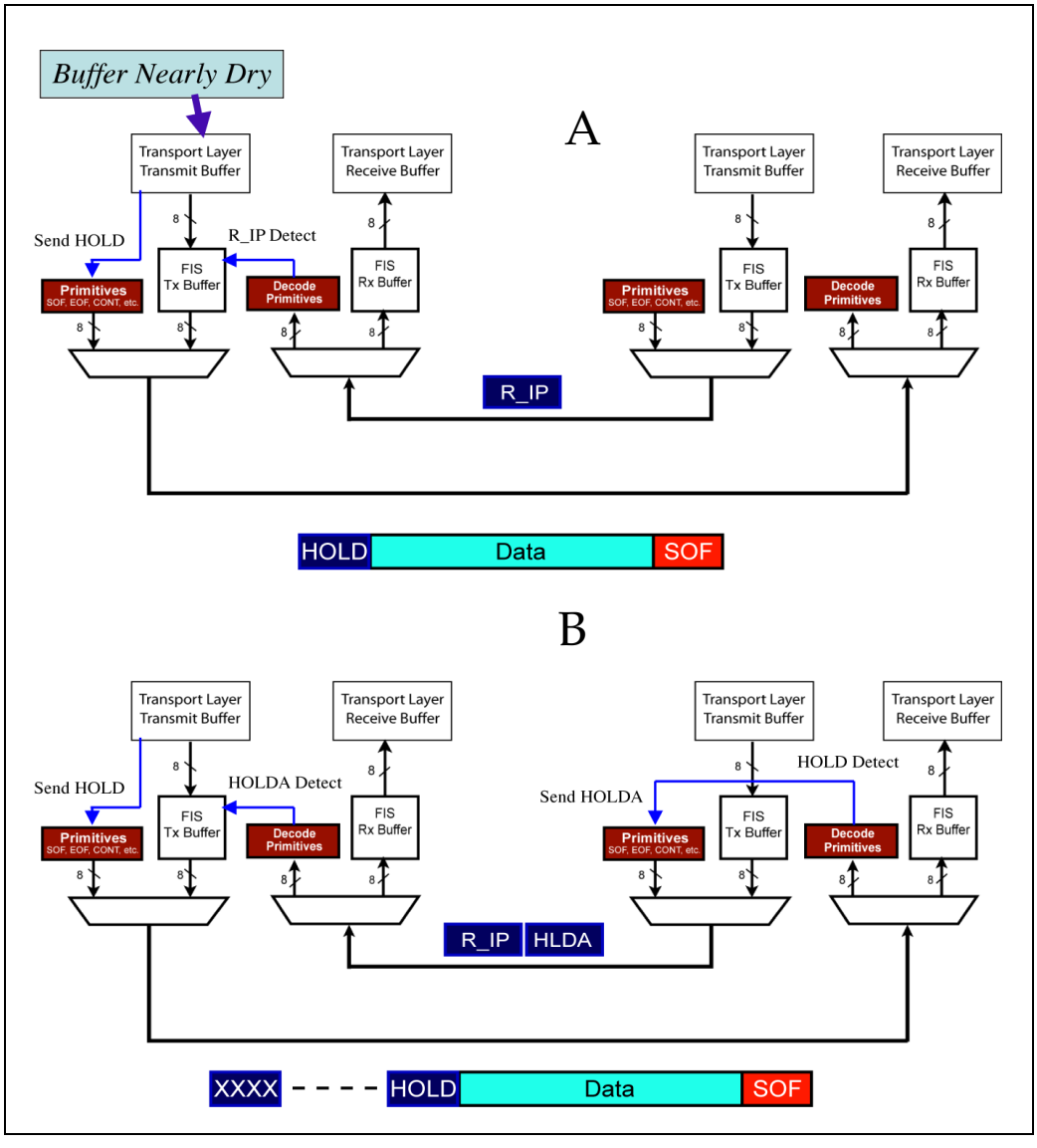
Dry Condition Results in HOLD

Figure 8-2 A on page 137 illustrates a condition in which the Transport layer transmit buffer has reached a nearly dry condition. The Data FIS transfer begins normally with the receiving node detecting the FIS and returning the receive in progress (R_IP) primitive. When the transmitting node reaches a nearly dry buffer condition is sends a HOLD primitive is to notify the receive that data is being temporarily stopped. The HOLD primitive is delivered continuously until it's time to resume the data transfer. Although not shown in Figure 8-2 A, the example presumes that the HOLD primitive is delivered twice followed by a continue (CONT) primitive to avoid repeated primitives that would contribute to EMI.

Receiver Acknowledges HOLD

The receiver continues to receive the data until it detects the HOLD primitive. The receiver responds to HOLD by returning a Hold Acknowledgement (HOLDA) primitive to the transmitting node as presented in Figure 8-2 B on page 137. Also notice that the information being sent across the link following HOLD is shown as "XXXX" to represent the pseudo-random data associated with primitive suppression scrambling. The receiver may also implement primitive suppression when delivering continuous HOLDA primitives.

Figure 8-2: Data FIS Transmission Temporarily on Hold and Acknowledged



9 Physical Layer Functions

Previous Chapter

When performing large data transfers the transmit and receive buffers within the transport layers may either overflow or underflow without some flow control mechanism. The previous chapter detailed the mechanism and protocols defined for managing the flow control.

This Chapter

This portion of the book focuses on FIS protocol and correspondingly this discussion of physical layer focuses on the functions related to FIS transmission and reception. Part 5 of this book details the physical layer's role in reset, initialization, Hot Plug, and electrical details.

The Next Chapter

The discussions thus far have focused on the FIS transfer protocols and for the most part has presumed that the transfers occur without error. The next chapter discusses the error detection mechanisms, describes the sources of these errors, and specifies the methods used to handle and report them.

Introduction

Functions associated with the physical layer during FIS transmission and reception are illustrated in Figure 9-1 on page 152 and include the following activities:

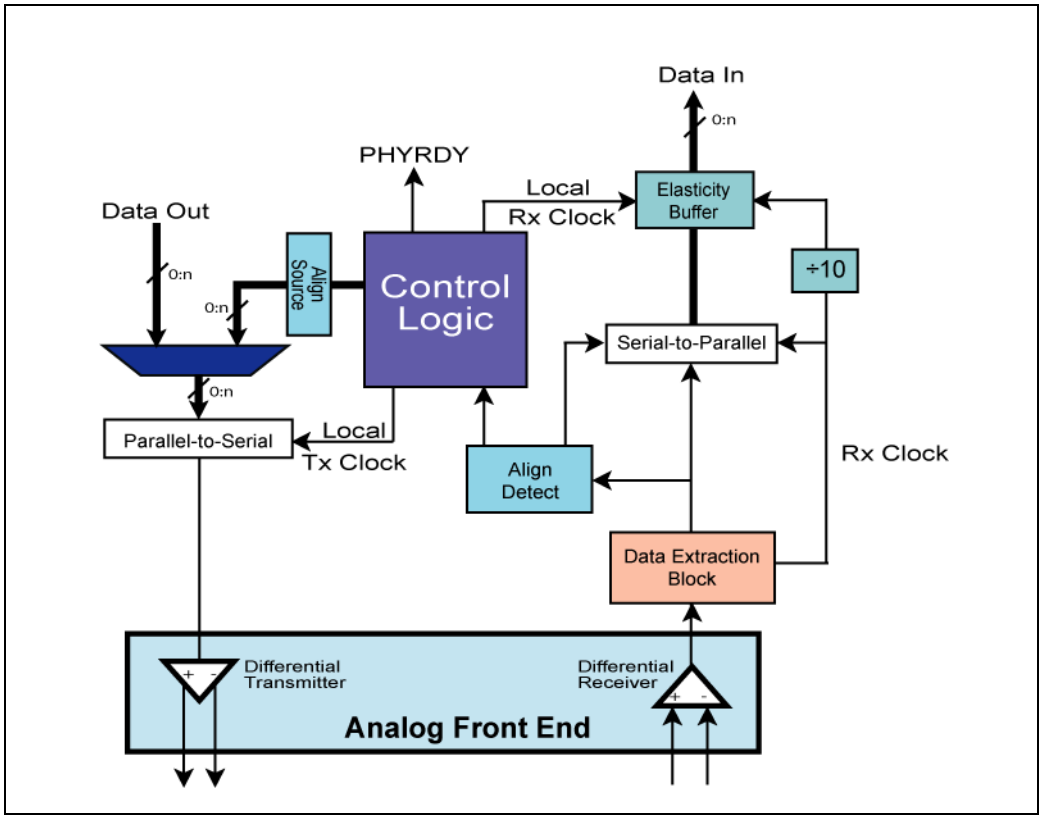
FIS Transmission

- Parallel to serial conversion
- Differential transmission at 1.5 or 3.0Gb/s
- Align primitive insertion for clock compensation
- Spread-spectrum clocking optional

FIS Reception

- Differential reception
- Data Extraction
- Serial to parallel conversion
- Align primitive decoding
- Elasticity buffer

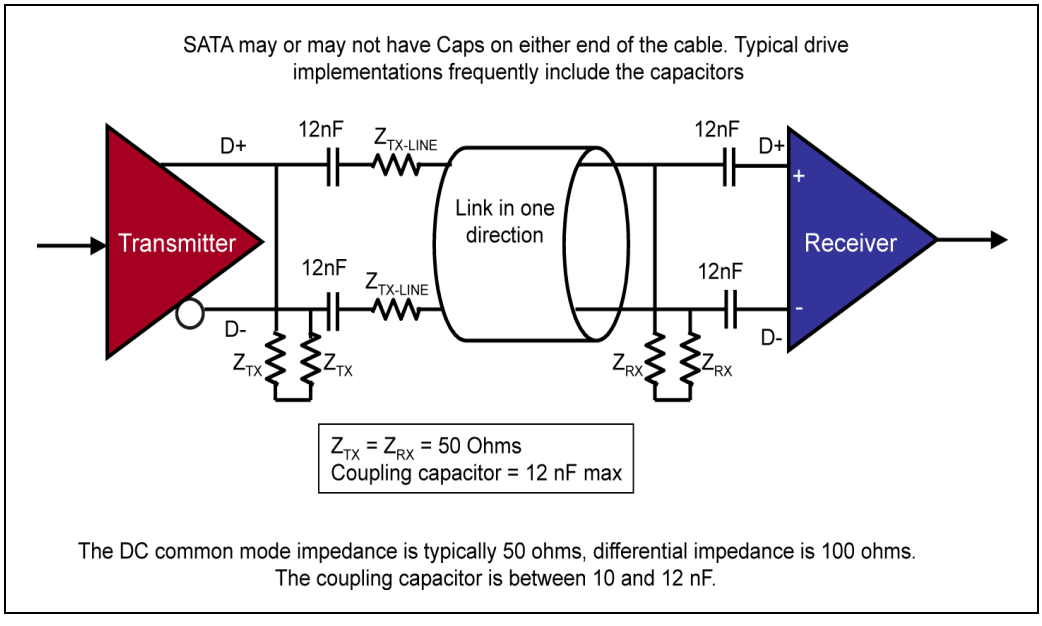
Figure 9-1: Functional Block Diagram — Physical Layer



Differential Transmitter/Receiver

Figure 9-2 provides a schematic view of the transmitter, receiver and link characteristics. Notice that the optional AC-coupling capacitors are implemented on many drives to allow different common mode voltages at the drive and HBA. The differential impedance of the link is nominally 100 ohms and the terminations provide impedance matching to reduce reflections.

Figure 9-2: Link Electrical Interface



Transmitter Characteristics

The peak-to-peak transmit voltage is specified as 400-600mV for Gen 1 drives and hosts and 400-700mV for Gen 2 drives and hosts. However, the transmission voltage required depends on the signaling environment, specified as:

- Internal connections (i) — internal or inside-the-box connections (e.g., internal to PC)
- Medium length (m) — longer cables and short backplane environments
- External Cabled (x) — very long external cables and long backplane applications

The minimum and maximum transmit voltages for the m and x applications change due to increased signal attenuation. Figure 9-1 lists the specified m and x voltages. These voltage specifications are required by the host side and may require additional interface circuitry between the host and drive, but no change is required for the drive itself. Otherwise, different drives would be required for different signaling applications. Similarly, receiver sensitivity is modified to account for these signaling applications (Table 9-2). See Chapter 19, entitled "Physical Layer," on page 325 for addition electrical information.

Table 9-1: Minimum/Maximum Transmit Voltages

Application	Gen 1	Gen2
Internal (i)	400 - 600mV	400-700mV
Medium (m)	500-600mV	500-700mV
External (x)	800-1600mV	800-1600mV

Receiver Characteristics

The differential receivers have an input sensitivity specified inTable 9-2 based on the different signaling applications described previously.

Table 9-2: Minimum/Maximum Receive Voltages

Application	Gen 1	Gen2
Internal (i)	325 - 600mV	275 - 750mV
Medium (m)	240 - 600mV	240 - 750mV
External (x)	275 - 1600mV	275 - 1600mV

Clock Management

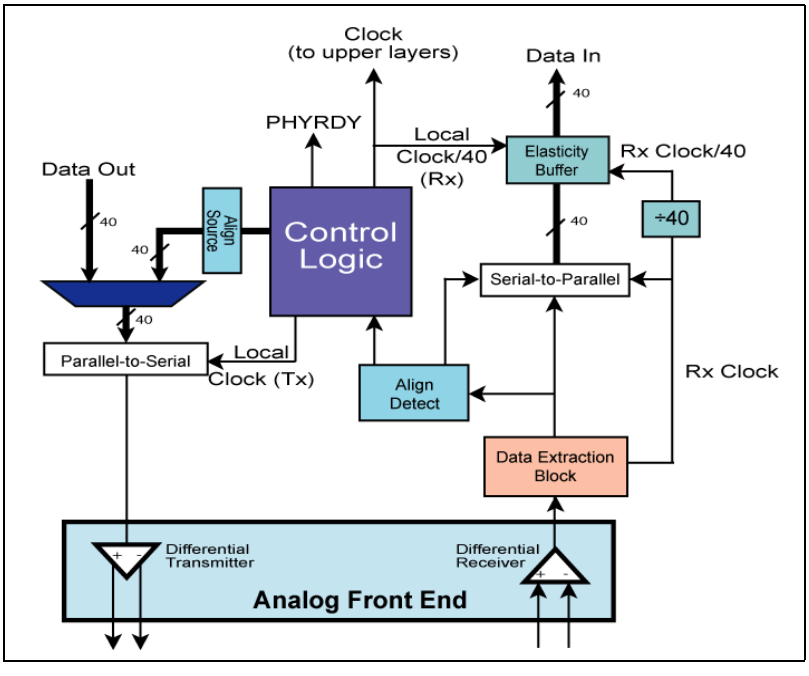
The physical layer is responsible for a variety of clock-related functions:

- Local clock generation
- Local clock division and distribution
- Receive clock recovery for data extraction
- Clock compensation management
- Spread-Spectrum Clocking (SSC) - optional

Local Clock Frequency

The local clock in the host and target is generated within the physical layer. This clock is used as the reference clock for transmitting data (at the Gen 1 or Gen 2 rate) and for clocking receive data out of the elasticity buffer. Figure 9-3 illustrates a clock distribution implementation. In this example, the parallel data path between the upper layers and the physical layer is 40-bits wide; thereby reducing the clock frequency by a factor of 40.

Figure 9-3: Example Clock Implementation



10 Error Detection and Handling

Previous Chapter

Because this portion of the book focuses on FIS protocol the previous chapter focussed on the physical layer functions related to FIS transmission and reception. Part 5 of this book details the physical layer's role in reset, initialization, Hot Plug, and electrical details.

This Chapter

The discussions thus far have focused on the FIS transfer protocols and for the most part has presumed that the transfers occur without error. The next chapter discusses the error detection mechanisms, describes the sources of these errors, and specifies the methods used to handle and report them.

The Next Chapter

Much of the discussion prior to the next chapter has focused on the delivery of individual Frame Information Structures. Next, the discussion turns to the various categories of commands that each require the exchange of a particular sequence of Frame Information Structures. It is the responsibility of the command layers to manage this sequencing.

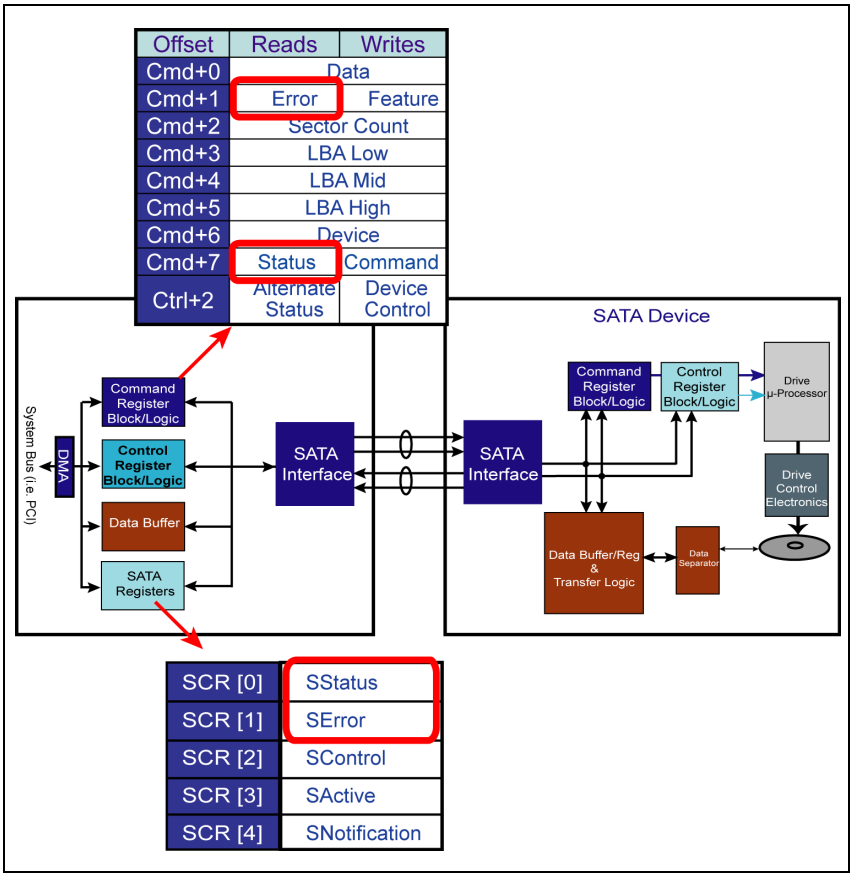
Scope of SATA Error Checking

The SATA environment supports ATA error checking as well as SATA-specific error checking. This capability can be segmented into the following categories:

- Commands failing to complete properly — reported in the ATA completion status and error registers
- FIS transfer Protocol Errors — reported in the SATA-specific error register
- SATA Link Transfer Errors — reported in the SATA-specific error register
- HBA Errors — reported in the SATA-specific error register and/or in IO bus-specific registers (e.g., in PCI configuration status registers)

Figure 10-1 on page 160 illustrates the location of the status and error registers that system software accesses to detect errors. The specific registers that contain error-related information include the ATA status and error registers located within the Shadow registers, along with the SStatus and SError registers. Note that in the AHCI implementation, the contents of the ATA registers are located in main memory instead of the HBA.

Figure 10-1: Location of Shadow (ATA) and SATA-Specific Register Blocks



Error Reporting — HBA Versus SATA Drives

The error reporting capabilities of the SATA drives and HBA are quite different. All of the status and error registers are located within the HBA, giving software the ability to detect that an error has occurred and in some cases determine the type of error, along with its severity. Drives can also detect errors but must forward notification of the error to the HBA for reporting to software. Two primary mechanisms are available for drives to communicate errors to the host:

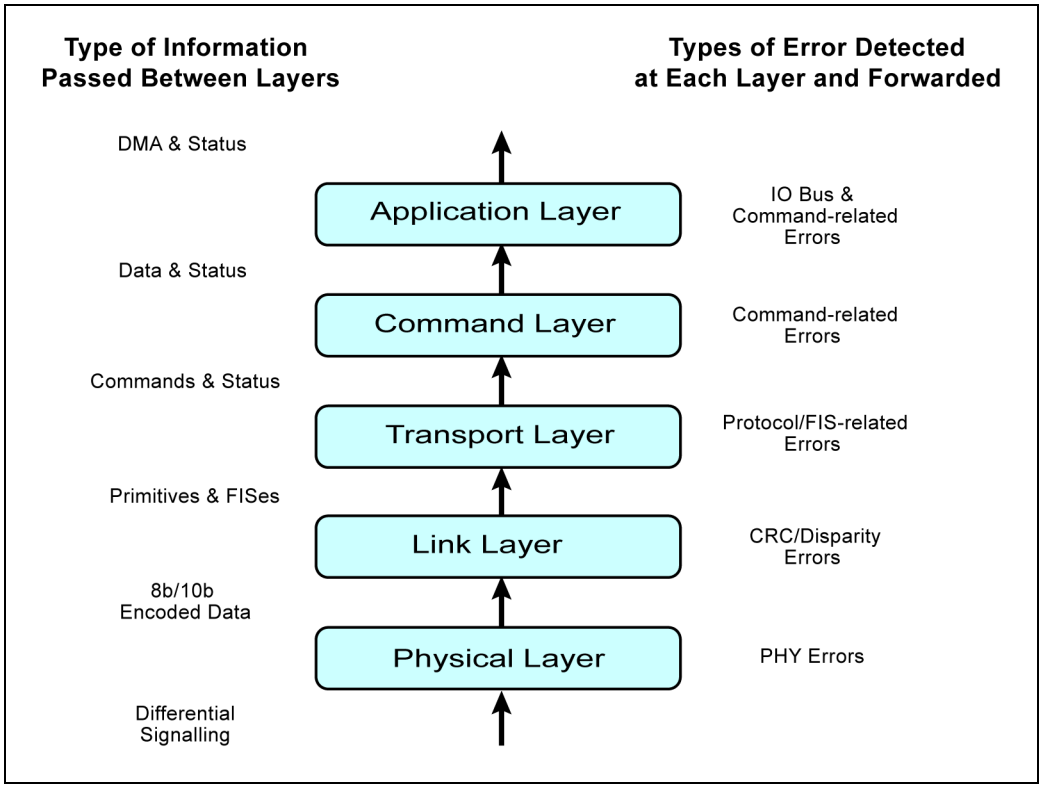
1. Drives update the ATA Status and Error registers during command execution and forward the results to the HBA via the Register FIS - Device to Host.
2. Drives report FIS transmission errors via the FIS transfer protocol handshake (R_ERR). In this case, the HBA has no visibility regarding the nature of the error.

Many SATA errors can only be reflected in the HBA's SStatus and SError registers. These same errors may be detectable within the drives but no mechanism is available to report the exact nature of these errors as is done by the HBA.

Error Reporting and Handling Mechanisms

SATA errors are generally detected and forwarded to upper layers where status and error registers may be updated and where the method of error handling is determined. Figure 10-2 on page 162 illustrates the hierarchical error detection and handling approach defined by the specification. On the left side of the diagram the type of information passed between the layers is highlighted, and the right side lists the types of error checks made at each layer.

Figure 10-2: SATA Error Detection and Reporting Methodology



The actions taken at each layer depend, in part, on the type and severity of the error. The specification identifies four actions or responses that can be taken:

- Freeze — indicates that no action is taken because of a severe error condition that is unrecoverable. In these circumstances, software will time-out because the command being performed did not complete. A reset is typically required to clear the error.
- Abort — indicates that the error conditions is presumed to be persistent and the associated command has failed. Persistent errors are typically forwarded to the Application layer to inform host software of the failed command.

- Retry — indicates that the error condition is expected to be transient, such as a CRC or disparity error. Such errors must have no adverse affect on the state of the SATA system; therefore, the failed transfer is generally retried. Errors detected at the Transport layer or below may be retried under hardware control (non-data frames only); whereas, errors detected above the Transport layer must rely on software to initiate the retry.
- Track/ignore — indicates a recoverable error conditions that is not critical and therefore can be ignored or tracked by software. Such errors include those that have been successfully retried by the Transport layer hardware. Even though these errors have been corrected by hardware, they typically affect performance adversely and may also indicate a component that is pending failure.

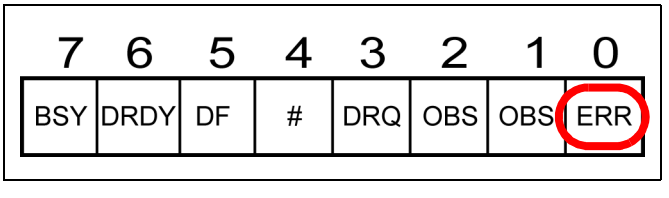
ATA Compatible Registers

When a drive executes a command it reports completion status within the ATA Status register. If a command fails to complete successfully, the Status register reflects the failure and the Error register identifies the nature of the error as described below.

ATA Status Register

Figure 10-3 depicts the contents of the ATA Status register. Note that for error reporting purposes, only the Error bit (ERR) is relevant. When set, ERR indicates the command failed due to some type of error condition and that the Error register contains additional information regarding the failure. Note that if the Status register Error bit is not set, then the contents of the Error register are invalid.

Figure 10-3: ATA Status Register Indicates if an Error Has Occurred



11 The Command Protocol

Previous Chapter

The discussions to this point have focused on the FIS transfer protocols and for the most part have presumed that the transfers occur without error. The previous chapter discussed the error detection mechanisms, described the sources of these errors, and specified the methods used to handle and report them.

This Chapter

Much of the discussion prior to the last chapter focused on the delivery of individual Frame Information Structures. The discussion now turns to the various categories of commands, each of which require the exchange of a particular sequence of Frame Information Structures. It is the responsibility of the command layer to manage this sequencing.

The Next Chapter

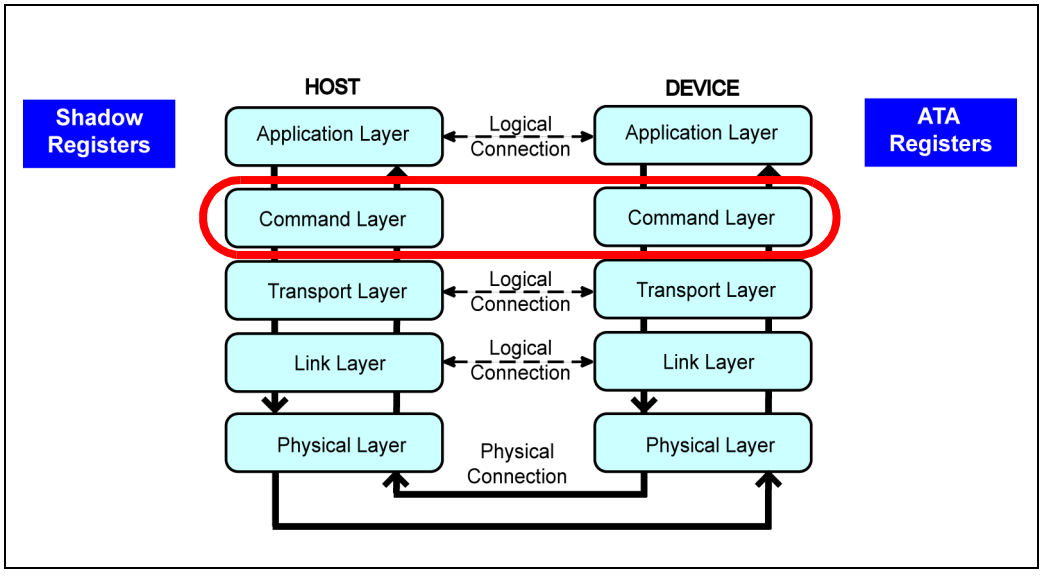
The next chapter discusses the functions associated with the ATA Control register. Writing to the Device Control register forces the HBA to send a Register FIS to the device. The particular bits written force the drive to take the specified action. Each of the Control register functions is discussed in detail.

Overview

This chapter focuses on the protocols associated with executing commands in the SATA environment. Most of the commands implemented are commands supported by the Parallel ATA implementations. However, new commands have been added to support new features available only in the SATA environment.

Figure 11-1 illustrates the Command layer within the hierarchy of SATA interface layers. When host software issues the command to the HBA it forwards the command to the drive where it is decoded and passed to the Application layer within the drive. The command layer state machine knows the sequence of FISes that must be exchanged to complete the command successfully. Similarly, the Command layer within the HBA also knows the FISes to expect and return for a given command.

Figure 11-1: Command Layer in Hierarchy



Command Types

Twelve different categories of command are defined by the specification. Commands are grouped by category because all commands within a given category have behaviors in common that allow them to use the same command protocol. That is, the sequence of FISes exchanged and the actions taken in the process of executing and completing the command is the same. Table 11-1 on page 179 lists the 12 command categories and identifies the number of commands supported by each.

Table 11-1: Command Protocol Types

Command Category	Number of Commands
Command Not Implemented	NA
Non-Data	34
PIO Data-In	13
PIO Data-Out	14
DMA-In	2
DMA-Out	2
DMA-In Queued	2
DMA-Out Queued	2
Packet (ATAPI)	1
Service	1
Device Reset	1
Execute Device Diagnostics	1

Command Delivery

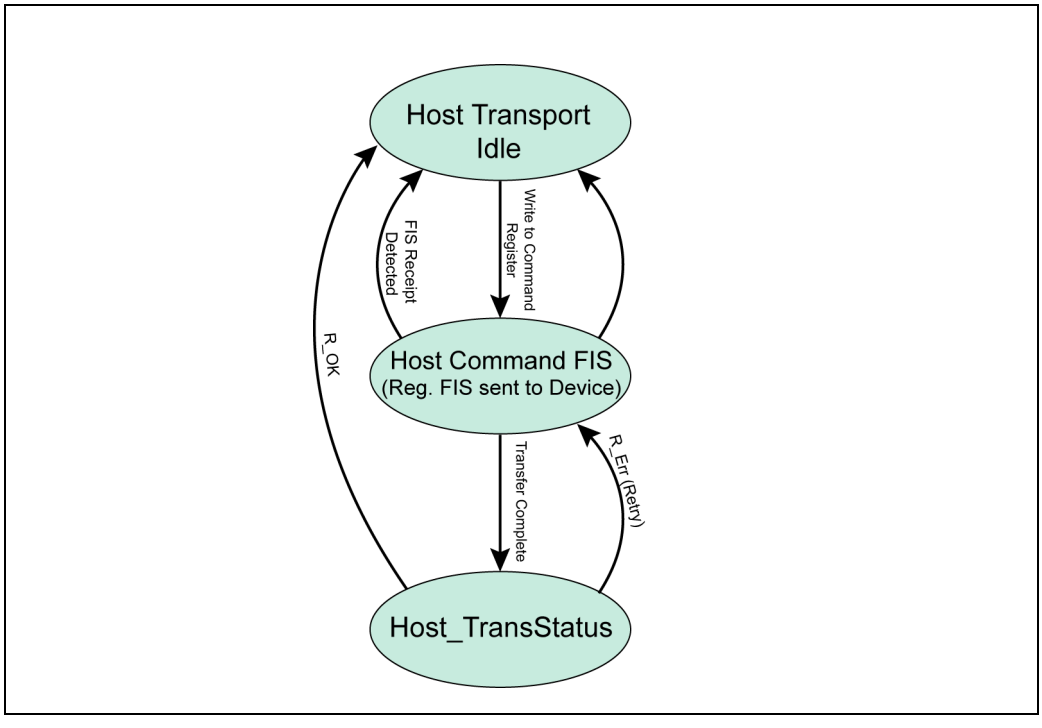
Every command sequence begins with a register FIS used to delivered the command to the drive. Once a command is received and recognized, the command protocol is established within the drive. The steps involved in this process are detailed below.

Command Transport to Disc Drive

The state diagram in Figure 11-2 on page 180 shows the delivery of a Register Host to Device FIS to the drive. Initially the host Transport layer is idled because is has not received a FIS transfer request from the Application layer. When a command value is written to the shadow register the HBA transitions to the "Host Command FIS" state, triggering FIS delivery to the drive. During

transmission the FIS transfer protocol is followed and the FIS is generally received without error and R_OK is returned to the HBA. Note that violations of the transfer protocol that are detected by the HBA can force the link back to the idle state as illustrated in Figure 11-2. Once the FIS is received, the drive performs various checks (disparity, CRC, etc.) to verify that the FIS is valid. Errors are reported via R_ERR and the FIS is retried.

Figure 11-2: State Diagram for Register FIS delivery to the Disc Drive



Command Reception

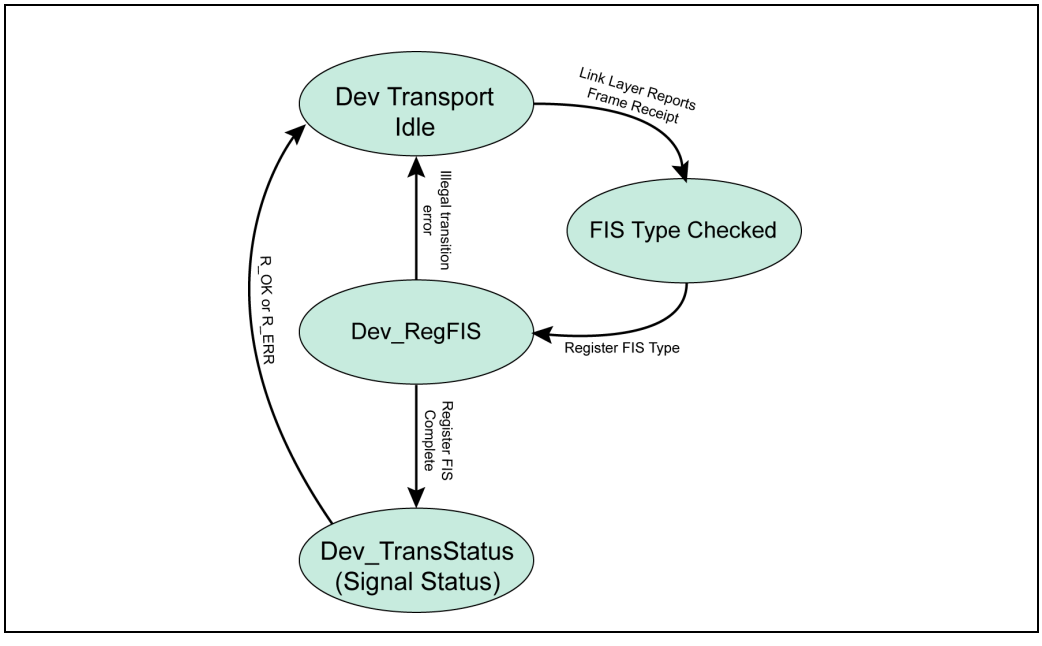
The state diagram in Figure 11-3 on page 181 illustrates the required actions associated with Register FIS reception by a disc drive. When the incoming FIS is detected by the Link layer it is forwarded to the Transport layer. The first byte received by the Transport layer is the FIS type field, which is checked and deter-

mined to be a Register FIS and the drive transitions to the "Dev_RegFIS" (Device Register FIS) state. Two transitions are possible while the drive is in this state:

- If during reception an illegal transition error occurs, the device sends SYNC primitives to the HBA; thereby transitioning its transmit side to logical idle.
- Once the FIS has been completely received with no illegal transitions, the drive transitions to the "Device Transfer Status" state.

In the Dev_TransStatus state, the Transport layer, having received and checked for errors returns either R_OK or R_ERR. Presuming the status is OK and that its type is a Register FIS, the Transport layer knows that the contents of the FIS must be forwarded to the Application layer, where the ATA registers are loaded with the new command and its associated parameters.

Figure 11-3: Receive States for Register Host-to-Device FIS Reception



12 Control Protocol

Previous Chapter

Earlier portions of the book focussed on the delivery of individual Frame Information Structures. The last chapter discussed the various categories of command, each of which require the exchange of a particular sequence of Frame Information Structures. It is the responsibility of the command layer to manage this sequencing.

This Chapter

This chapter discusses the functions associated with the Device Control register. Writing to the Device Control register forces the HBA to send a Register FIS to the device. The particular bits written force the drive to take the specified action. Each of the Control register functions is discussed in detail.

The Next Chapter

The next chapter introduces the primary features added by the SATA II specification.

Write Control Protocol

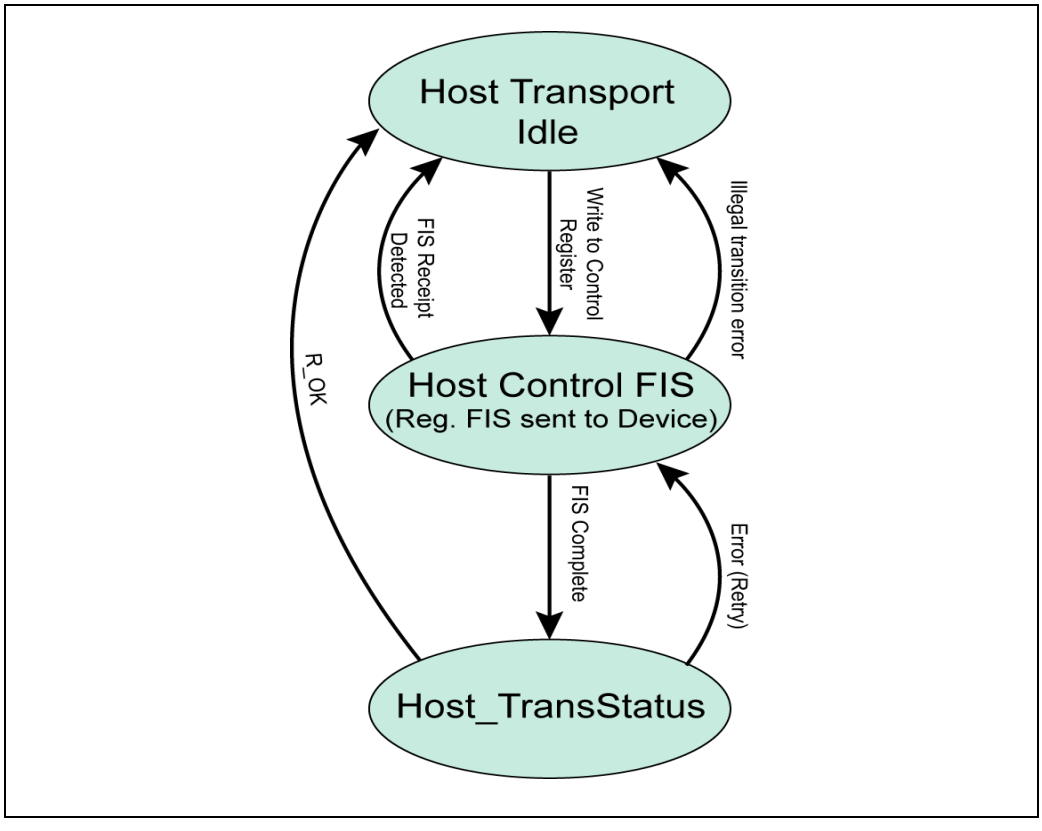
When software writes to the shadow Control register, the Control register contents are copied to the Control field of a Register FIS and the FIS's "C" bit cleared. The contents last written to the Command Block registers are also copied to the FIS and the Register FIS is delivered to the drive. Figure 12-1 on page 222 illustrates the HBA protocol associated with a write to the shadow Control register. Prior to the Control register write operation, the HBA Transport layer is in the logical idle state. When the shadow control register is written by host software the HBA transitions to the Host Control FIS state. During this state the Register FIS is delivered to the device. Three possible conditions can occur:

1. If the Register FIS is transmitted without error, the HBA transitions to the Host_TransStatus state, where the HBA waits for the drive to report results of the transmission.

2. If an illegal transition is detected during transmission of the FIS, the HBA terminates the FIS transfer and returns to logical idle.
3. If the HBA detects reception of a FIS from the drive, it must also return to idle and respond to the incoming FIS.

Once FIS transmission is complete the HBA transitions to the Host_TransStatus state where it awaits the result of the transmission from the drive. In the event a transmission error is detected, the HBA will be notified via an R_Err primitive and the Register FIS will be retried. Upon successful receipt of the Register FIS, the drive updates its Control register with the new Control register values.

Figure 12-1: Host Control Protocol

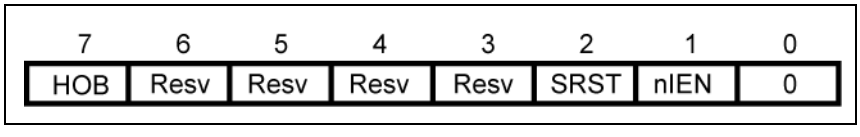


Device Control Register Functions

The Control register supports several functions defined below. Figure 12-2 on page 223 illustrates the Control register bit field designations.

- nIEN (Interrupt Enable, negative logic) when nIEN is cleared (0), interrupt request generation is enabled within the device, while setting the bit disables interrupt generation.
- SRST (Soft Reset) — Software sets this bit to reset the drive. Reset is asserted when the SRST bit is set (1) and is cleared when the SRST bit is cleared (0).
- HOB (High Order Byte) — Some register address support back-to-back byte-sized writes to the same address location to load additional values to support features such as 48-bit Logical Block Addressing. Setting the HOB bit enables software to read the previous written byte from the selected 8-bit register.

Figure 12-2: Control Register Contents



Interrupt Enable (nIEN) Control Protocol

The nIEN bit is implemented to provide legacy software support. Following reset the nIEN bit is cleared, resulting in interrupts being enabled. Software can disable interrupts by setting the nIEN bit. When software either sets or clears the nIEN bit, a Register FIS is delivered to the SATA drive; thereby, setting or clearing the nIEN bit within the drive's control register. Note, that any write to the shadow control register clears an interrupt pending condition within the HBA.

Also, unlike the PATA environment, the nIEN bit has no affect on a SATA drive's behavior. In the SATA environment the HBA actually signals interrupt requests and also handles the nIEN function.

Software Reset (SRST)

Software resets a SATA drive in legacy fashion by writing a one to the SRST bit within the shadow Control register. This of course causes the HBA to send a Register FIS (with "C" bit cleared) to the drive. Software must write a zero to the SRST bit to clear the reset.

An SRST affects only the SATA device and has no effect on the link (unlike a COMRESET). Also during a soft reset the SATA device performs internal diagnostics and once the SRST bit is cleared (followed be a Host-to-Device Register FIS) the drive returns a Register FIS to the HBA to report results of the drive's diagnostics. This SATA drive protocol is illustrated in Figure 12-4 and described below:

1. The HBA has delivered a Register FIS to the drive to notify it that software has set the SRST bit in the shadow Command register. The Device is currently sending logical idle when the drive receives the Register FIS.
2. The drive checks the FIS type and detects the C bit cleared and the SRST bit set; thereby, indicating that Software has issued a Reset.
3. During the Reset_Assert state, the drive re-initializes to its default state, starts internal diagnostics, and waits for software to clear the reset condition.
4. The drive receives the second Register FIS with the C bit clear and SRST bit cleared, causing transition to the Execute Diagnostic state. The drive waits for the diagnostics to complete (if not already done) and updates its ATA registers to report either good or bad status.
5. The drive sends a Register FIS to the HBA to report the results.

Figure 12-3 illustrates the contents of the ATA registers when reporting both good and bad status. When bad status is returned, the Error register contains a drive-specific error code (any value except 01h).

Figure 12-3: Soft Reset Status Values

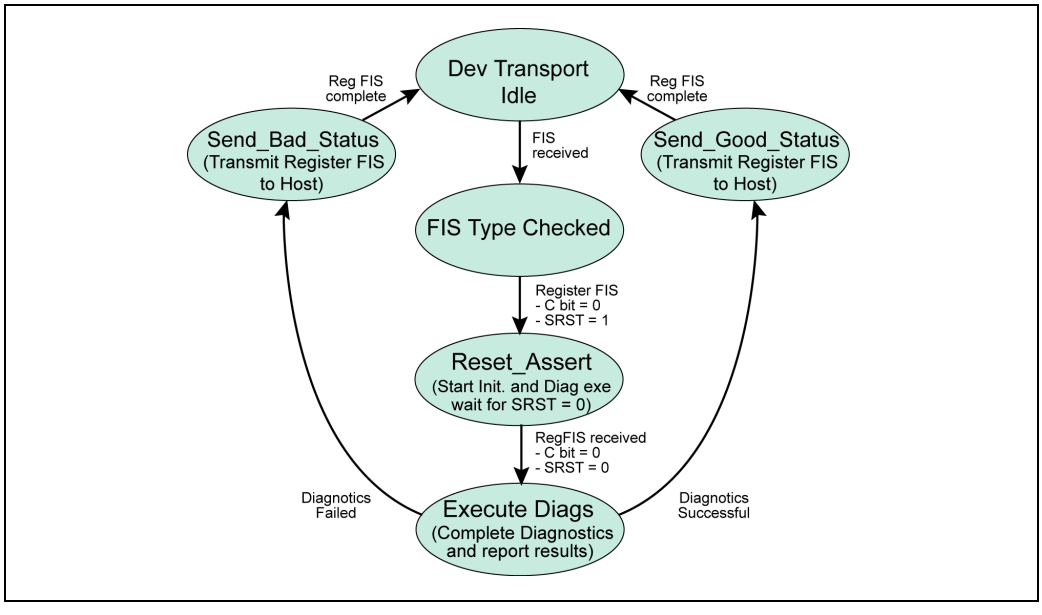
Good Status

Sector Count	01h
Sector Number	01h
Cylinder Low	00h
Cylinder High	00h
Device/Head	00h
Error	01h
Status	00h-70h*
* Bits 4-6 are device specific	

Bad Status

Sector Count	01h
Sector Number	01h
Cylinder Low	00h
Cylinder High	00h
Device/Head	00h
Error	00h, 02h-27h
Status	00h-70h*
* Bits 4-6 are device specific	

Figure 12-4: SATA Drive Reset Protocol



13 SATA II Features

Previous Chapter

The previous chapter discussed the functions associated with the Device Control register. Writing to the Device Control register forces the HBA to send a Register FIS to the device. The particular bits written force the drive to take the specified action. Each of the Control register functions is discussed in detail.

This Chapter

This chapter introduces the primary features added by the SATA II specification and describes the motivation for adding them.

The Next Chapter

The next chapter describes the concepts and mechanisms associated with Native Command Queuing (NCQ). This mechanism provides major performance improvements over mainstream ATA and SATA drives that use standard DMA Read and Write operations.

Overview

The SATA II implementation was introduced as an extension to the initial SATA specification. SATA II introduced many new features most of which are optional. The new features fall into several categories:

- Performance and Reliability Features — Higher transmission rates and support for Native Command Queuing are among the most important improvements added by SATA II.
- Server-specific Support — Many new SATA II features focused on making SATA more appealing for server applications. Most of the server-related enhancements focus more on the SATA infrastructure and less on the drives themselves.
- Fixes and Enhancements to SATA — A variety of fixes and miscellaneous features were added to SATA II; including, asynchronous event notification.

Another potentially important issue is that Serial Attached SCSI (SAS) connectors permit attachment of SATA drives, and supports the SATA protocol.

This chapter introduces the major features of the SATA II specification.

Performance and Reliability

Three notable features introduced with the SATA II specification improve performance and/or reliability over the previous versions:

- SATA Generation 2 transmission rates (3Gb/s)
- Native Command Queuing
- Asynchronous Signal Recovery

Generation 2 Transmission Rates

The generation 2 transmission rates of 3 Gb/s increased the maximum throughput of the SATA interface to 300 MB/s. The increased transmission speed is particularly important when multiple SATA drives share the same HBA link, which is characteristic of the SATA II Port Multiplier implementation. Chapter 19, on page 325 for details regarding the Gen II clock and signaling.

Native Command Queuing

Native Command Queuing (NCQ) can improve overall storage subsystem performance, and can also improve drive reliability and durability. NCQ enhances drive performance by executing queued commands out of order; thereby, reducing overall seek time. This process also results is less mechanical movement when compared with sequential execution of commands, thus reducing wear. See Chapter 14, entitled "Native Command Queuing," on page 235 for more information.

Asynchronous Signal Recovery

The Phy Layer state machine defined prior to SATA II did not include the ability to automatically attempt recovery when communication is lost between the Host and SATA device (i.e., no signals are received). This condition can occur for a variety of reasons including the signal connector being unplugged during normal operation and then replaced while power is still present.

SATA II Phys that support Asynchronous Signal Recovery will, upon detecting that link communications are down, automatically attempt to re-establish communication. These Phys use a variable named the RetryInterval that determines the rate at which signal recovery polling is attempted. See "Asynchronous Signal Recovery" on page 316.

Enhanced Support for Server Applications

A number of server-related features were introduced with the SATA II specification:

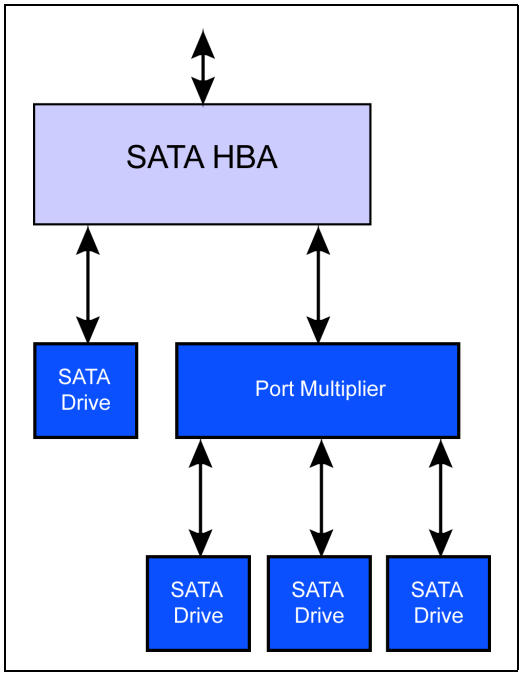
1. Port Multipliers that allow a single HBA port to support up to 15 drives
2. Port Selectors that provide dual-port functionality at the drives
3. Hot Docking support
4. Multilane Cables
5. Support for Enclosure Services and Management

Port Multipliers

Rather than having SATA drives connect directly to individual HBA ports, the port multiplier provides up to 15 additional ports to which SATA drives can be connected. This provides port fan-out capability needed in large rack mount implementations.

Figure 13-1 on page 232 illustrates the typical SATA drive connection to an HBA port, along with the Port Multiplier approach. Obviously, the HBA can only communicate with one drive at a time, thus overall performance may be affected with the Port Multiplier approach, but provides access to huge amounts of storage at much lower costs when compared to SCSI and Fibre Drive implementations. Additional information regarding the Port Multiplier implementation can be found in Chapter 15, on page 257.

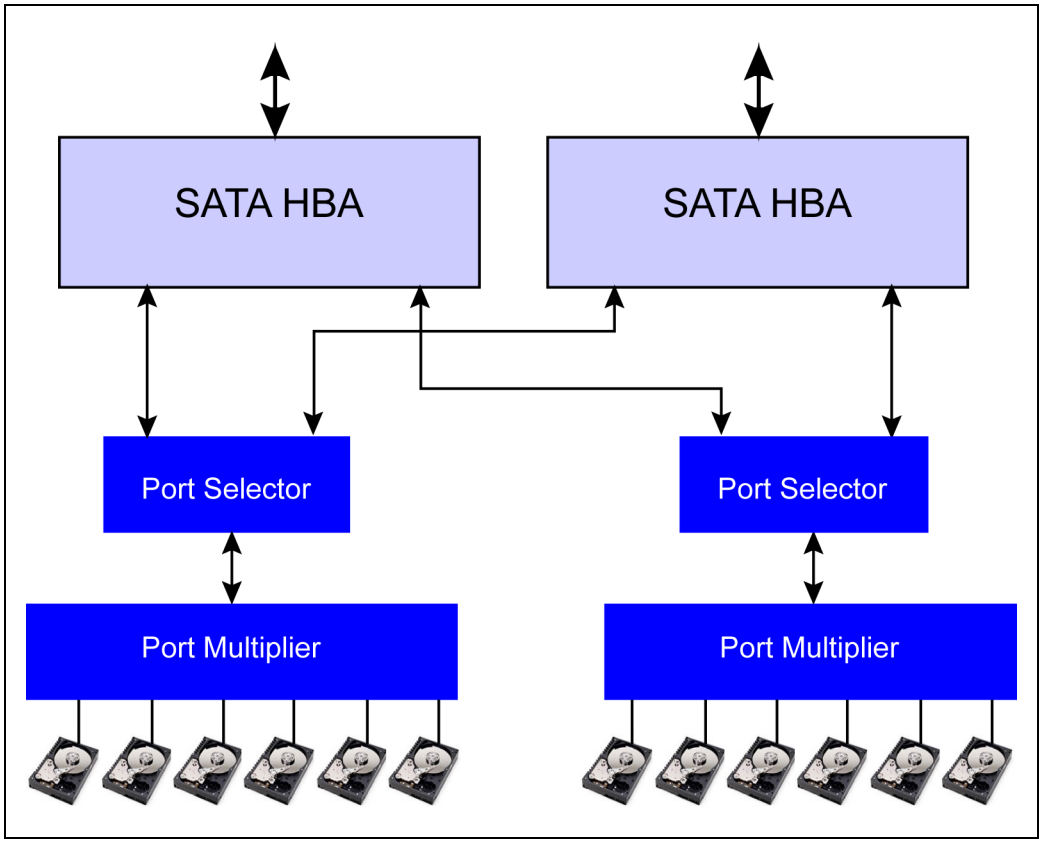
Figure 13-1: Port Multiplier Example



Port Selectors

The port selector feature provides a way for two HBAs to access a SATA drive. The goal of the port selector capability is to provide a fail-over mechanism to assist in the goal of "non-stop operation" for server implementations. Figure 13-2 on page 233 illustrates an example port selector implementation that includes the use of port multipliers.

Figure 13-2: Example Port Selector Implementation



Multilane Cables

The SATA specification defines a variety of Multilane cables for both internal and external implementations. These cables are discussed in Chapter 20, on page 361.

14 Native Command Queuing

Previous Chapter

The previous chapter introduced the primary features added by the SATA II specification.

This Chapter

This chapter describes the concepts and mechanisms associated with Native Command Queuing (NCQ). This mechanism provides major performance improvements over mainstream ATA drives that use standard DMA Read and Write operations.

The Next Chapter

Port Multipliers provide a fan-out capability, whereby up to 15 ports can be accessed via a single HBA port. This capability is useful in large rack mount implementations. The next chapter details the hardware and software required to support the Port Multiplier solution.

Overview

This section takes an evolutionary approach to describing the performance limitations associated with typical Parallel ATA drives and associated software. Next, the concept of command queuing is reviewed as a method of improving performance, and its limitations are explored. Finally, today's drives supporting Native Command Queuing can provide significant improvements in performance over the earlier approaches.

The Problem - Limited Performance

In the mainstream PC environment, software and most ATA drives were designed to execute one command at a time. That meant that once a drive completed a command, the drive remained idle for relatively long period of time before the next command was issued. Even if a drive had the ability to deliver data across the ATA bus at a rate of 66MB/s, that speed advantage was less significant due to the requirement that a given command must have completed before the next could be sent. This type of command processing is sometimes termed synchronous IO.

Queuing Helps

Once software supports asynchronous IO, it can issue (or queue) one or more subsequent commands prior to the current command completing. This ability reduces the time that a drive remains idle between commands. This capability is often referred to as software queuing. Note however, that software is still restricted to sending the next command to a standard ATA drive only after the previous command completes.

Even better performance is possible if the drive also supports Command Queuing. That is, some drives are designed to queue commands internally. This reduces even further the delays between completing one command and beginning the next. Command Queuing was first included in the ATA/ATAPI-4 specification, but mainstream software at the time did not support asynchronous IO. Consequently, few drives ever supported Command Queueing.

Native Command Queuing is Better

Today's drives based on SATA II typically support Native Command Queuing, or NCQ. This queuing technique is similar to that described in the previous section, except the commands enqueued within an NCQ drive can be completed out-of-order. Algorithms used by NCQ drives schedule the next command to perform based on the shortest time required to reach the associated target sector, rather than scheduling commands based on the order in which they were received. This can result in average seek times being reduced significantly; thereby, providing dramatic increases in a drive's performance. Some testing has reported that the performance of a 7200 rpm drive with NCQ is roughly equivalent to that of a standard 10k rpm drive.

Figure 14-1 on page 238 illustrates two disc layouts that depict the same sequence of four commands pending and the associated sectors to be read. Side A in Figure 14-1 illustrates the commands being performed sequentially, while side B illustrates the same commands being performed using the NCQ approach. In the examples, sequential accesses of the sectors start a point 1. Side A accesses might proceed as follows:

- Sector 1 sequence to start of Sector 2 sequence (approx. 1.34 revolutions)
- Sector 2 sequence to start of Sector 3 sequence (approx. 1.26 revolutions)
- Sector 3 sequence to start of Sector 4 sequence (approx. 1.22 revolutions)
- Sector 4 sequence to end (approx 0.45 revolutions)
- Total Revolutions = approx. 4.27

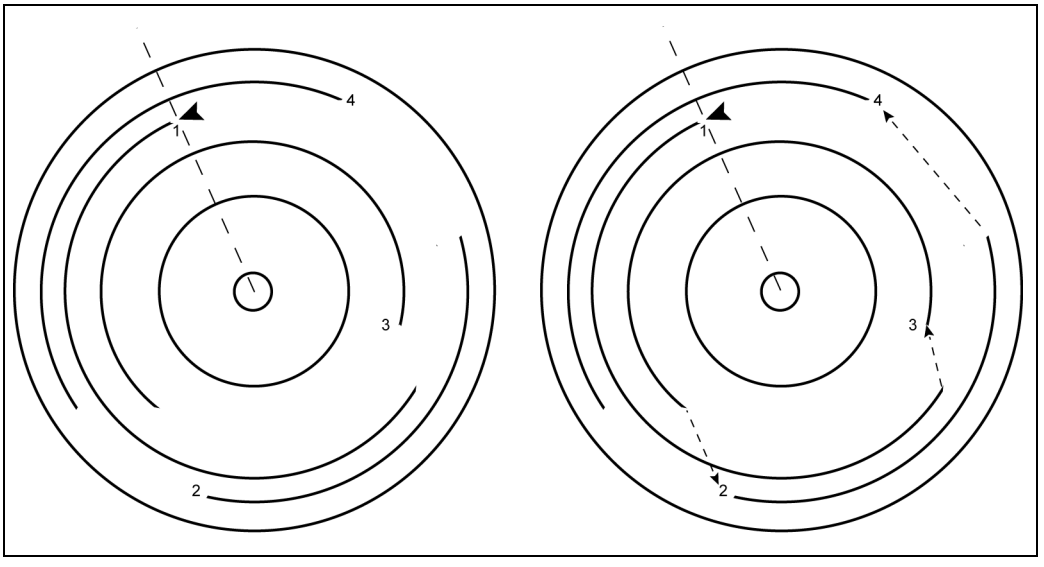
In contrast with NCQ the accesses might proceed are shown in Side B in Figure 14-1.

- Sector 1 sequence to start of Sector 3 sequence (approx. 0.60 revolutions)
- Sector 3 sequence to start of Sector 2 sequence (approx. 0.76 revolutions)
- Sector 2 sequence to start of Sector 4 sequence (approx. 0.46 revolutions)
- Sector 4 sequence to end (approx 0.45 revolutions)
- Total Revolutions = approx. 2.27

This simple example shows how the overall time required to perform the four commands can be significantly reduced when using NCQ. Another factor associated with this approach is that the amount of mechanical movement is also reduced, in terms not only of the number of rotations but also of the number of steps taken by the head as it moves between tracks (cylinders). This should also lead to better drive reliability.

The drive, the HBA, and software all must be designed to support NCQ. The following sections define the specific elements used in the SATA environment to support NCQ.

Figure 14-1: Sequential versus NCQ Example



System Support Requirements

Three primary elements are included in the SATA protocols that support NCQ:

- First Party DMA — this mechanism provides an NCQ drive with the ability to select and initiate a DMA operation for any of the enqueued commands without involving host software. Because an NCQ drive re-orders command execution, when data is ready for transfer only the drive knows which command is being performed and therefore which locations in memory should be accessed. This is accomplished via the The First Party DMA Read and Write commands that include a tag value that associates each command with the scatter/gather list that software has prepared for the command.
- Race-Free Status Return — the First Party DMA protocol allows completion status to be sent to the host for whichever command happens to have completed. This mechanism does not require that a separate Register FIS be transmitted to the host in order for status to be communicated. Instead, a Set Device Bits FIS is used to communicate which command has completed, and in some cases, multiple completions can be reported with a single FIS.
- Interrupt Aggregation — The drive generally sends an interrupt each time it completes a command. This adds to overhead and overall system perfor-

mance suffers. However, the average number of interrupts per command can be fewer than one when using NCQ. This can happen when two queued commands complete is a short span of time, resulting in two interrupts being reported to the HBA. When the interrupt handler checks status it will detect that two interrupts have occurred and will handle both; thereby, reducing overhead.

A variety of new features are required to support Native Command Queuing (NCQ). This support includes:

- NCQ Drive Features
- New First Party DMA Read and Write Commands
- Queue Management Capability
- Command Tracking

NCQ Drive Support

As suggested earlier in this chapter, one of the major impediments to improving drive read and write performance is the latency experienced when seeking the next starting sector. The two primary components of this latency are:

- Seek Latency (Head Positioning) — the time it takes the actuator to move the head to the target track
- Rotational Latency — once the target track is reached, the disc must rotate to the starting sector position in the selected track.

NCQ drives implement complex algorithms to re-order command execution to determine the shortest overall access time. An NCQ drive attempts to reduce this latency by reordering the execution of a group of up to 32 pending commands. This process attempts not only minimizing seek time, and rotational latencies, but also considers the age of the pending commands. That is, it tracts the amount of time a command has spent in the queue. Note also that when new commands are placed on the queue, they are dynamically included in the ordering schedule.

Some drives may also support accessing portions of a contiguous sector block to reduce overall latency. This means that offsets associated with a given command block transfer must also be managed by the drive. For example, consider the following scenario where multiple FPDMA Read command are pending completion:

15 Port Multipliers

Previous Chapter

The previous chapter described the concepts and mechanisms associated with Native Command Queuing (NCQ). This mechanism provides major performance improvements over mainstream ATA drives that use standard DMA Read and Write operations.

This Chapter

Port Multipliers provide a fan-out capability, whereby up to 15 ports can be accessed via a single HBA port. This capability is useful in large rack mount implementations. This chapter details the hardware and software required to support the Port Multiplier solution.

The Next Chapter

A common goal of server implementations is non-stop operation. This typically includes some type of redundant capability that allows for failures to be managed without shutting down the entire system. Port Selectors provide access to SATA drives from two separate sources, thereby permitting a fail-over solution. The next chapter discusses the Port Selector implementation and associated protocols.

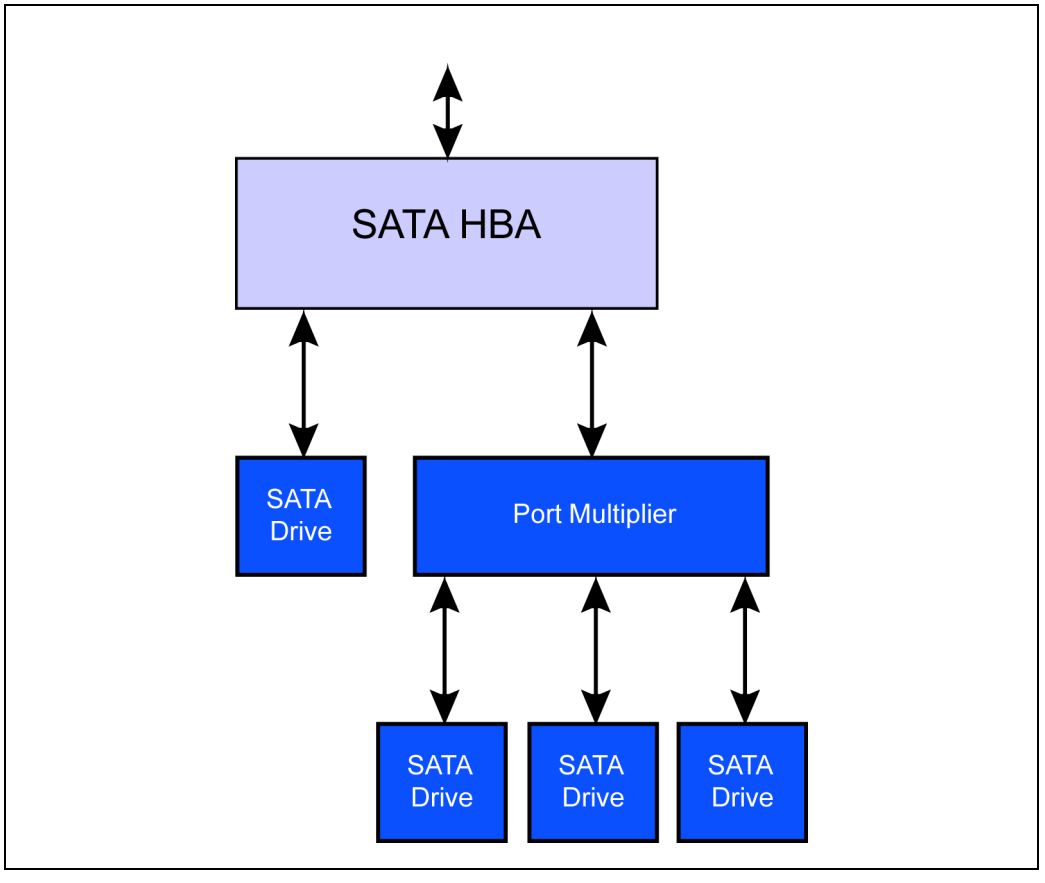
Overview

Because SATA drives are normally implemented with a point-to-point connection to a single HBA port, the cost and ease of implementing large drive installations is problematic. To mitigate this problem SATA II defined Port Multipliers that provide up to 15 additional ports to which SATA drives can be connected, called Device Ports. This provides the port fan-out capability needed in large rack mount implementations.

Figure 15-1 on page 258 illustrates the typical SATA drive connection to an HBA port, along with the Port Multiplier approach. Obviously, the HBA can only communicate with one drive at a time, thus overall performance may be

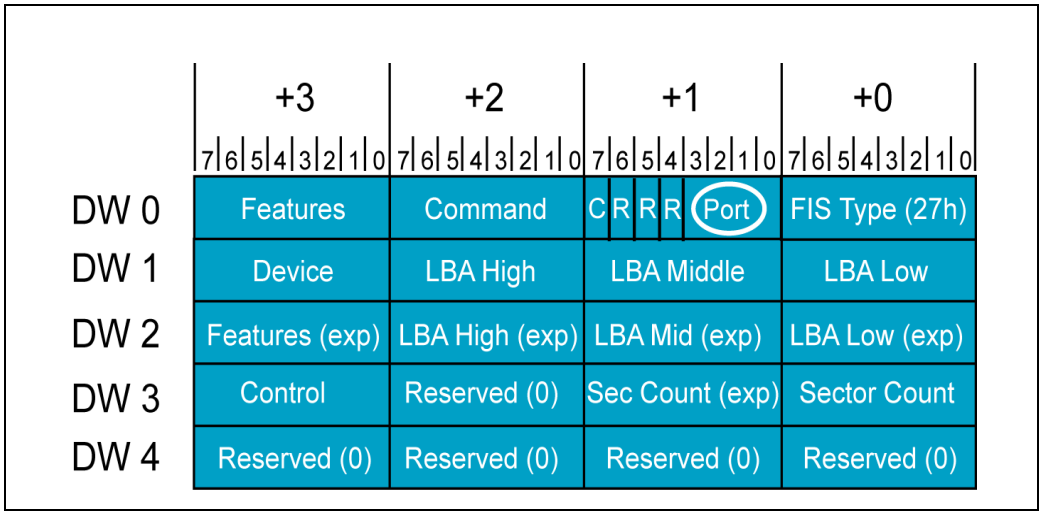
affected with the Port Multiplier approach, but it provides access to huge amounts of storage at much lower costs when compared to SCSI and Fibre Drive implementations. Note also that Port Multipliers are not allowed to be cascaded.

Figure 15-1: The Port Multiplier Concept



The Port Multiplier implementation does not require any modifications to the drives. However, host software must be able to assign port numbers to each FIS that is delivered so that it can be routed to the intended recipient. The FIS definition includes a Port Number field that was previously reserved. For example, Figure 15-2 on page 259 depicts a Register FIS, Host-to-Device with the Port Number field highlighted.

Figure 15-2: Port Number Added to Support Port Multiplier Routing

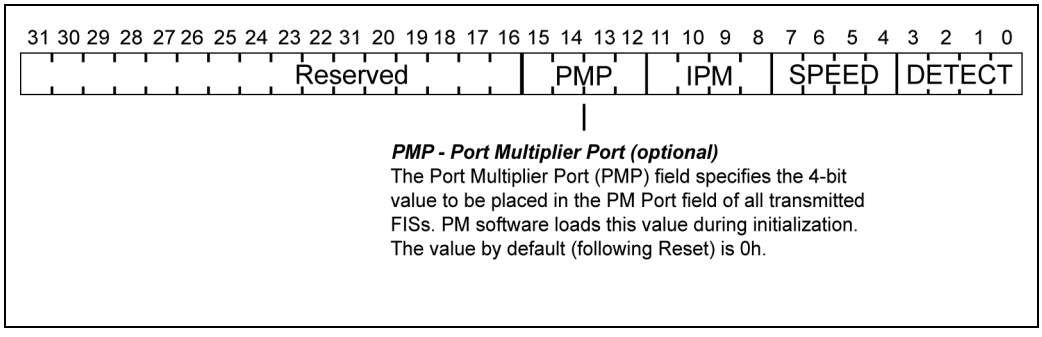


Port Multiplier Port Addresses

Some method was needed to direct frames sent from the HBA to the appropriate target drive. SATA uses a port numbering scheme to accomplish this routing. Software normally targets a specific drive by specifying an HBA port. When port multipliers are used drives must be targeted based on both an HBA port and Port Multiplier port. Two mechanisms can be used to assign port numbers that determine which drive is being targeted:

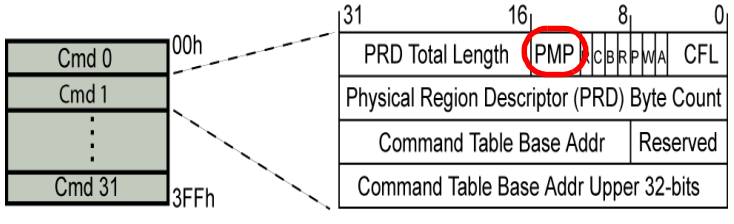
1. **Command-Based Switching** — This method of port number assignment is used when an HBA supports command queueing to one drive at a time. In this case, software assigns the PM port number via the Port Multiplier Port (PMP) field within the SControl register before writing the command. (See Figure 15-3 on page 260.) The HBA uses that PMP value to populate the Port field within the FIS. If a subsequent command targets a different drive, the PMP field must be updated with the new port number prior to issuing the next command. This approach is limiting in that when a Port Multiplier is used, only one drive at a time can be efficiently performing queued commands.

Figure 15-3: PMP Field within the SControl Register



2. **FIS-Based Switching** — This method provides dynamic context switching between ports. That is, the HBA supports the ability to handle consecutive commands, each of which can have different port numbers. In the case of AHCI, a port number field is associated with each entry of the command queue. When the queue is read by the AHCI controller it saves the Port number and than populates the Port field within the corresponding FIS.

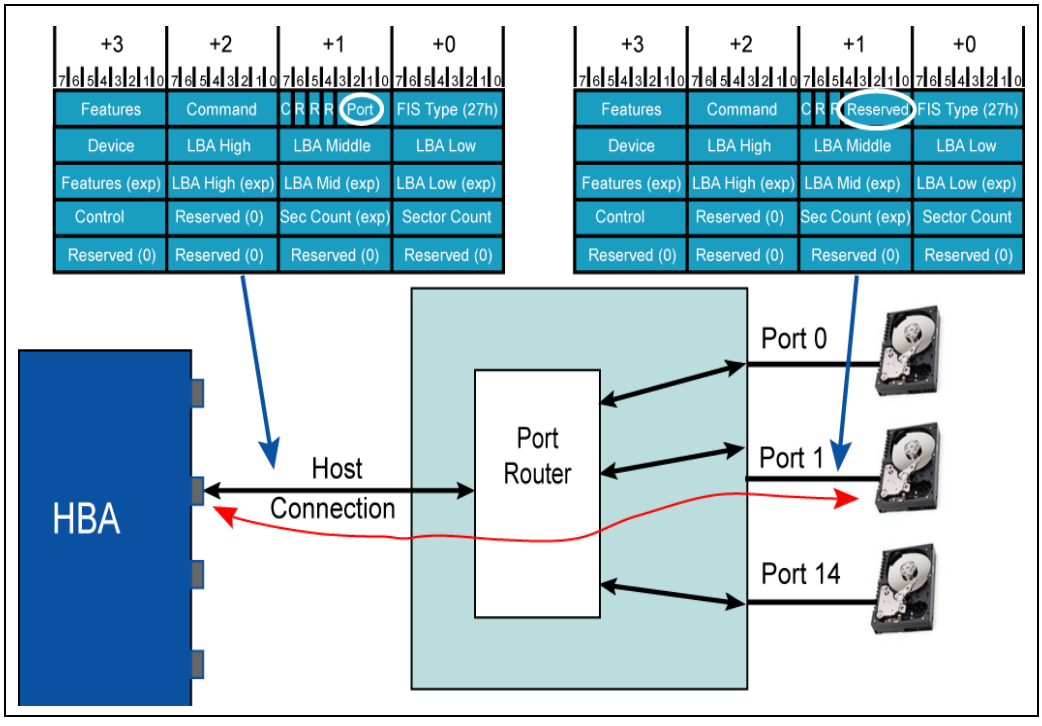
Figure 15-4: PM Port Assignment within Command Queue in AHCI Implementations



Frame Routing

Each frame that targets a PM port is routed as illustrated in Figure 15-5 on page 261. Note that a frame being sent from the HBA to a drive will have the target Port number specified. The PM routes the frame to the specified port but will remove the port number before forwarding the frame to the drive. When the drive returns a frame the port number field will initially be reserved, and the PM will assign a port number to identify which drive sent the frame.

Figure 15-5: Routing Frames to the Target Drive



Device Port Numbers

A port multiplier may implement up to 15 ports to which SATA devices may be attached. These device ports must be numbered starting with device port zero and be assigned sequentially up to the maximum number of ports implemented. Because SATA is a point-to-point implementation, each device port interface performs many of the same functions associated with an HBA port interface. For example, each Port Multiplier port has the SATA-specific registers that are used to specify capabilities, control the link, and gather and report status and errors.

16 Port Selectors

Previous Chapter

Port Multipliers provide a fan-out capability; whereby, up to 15 ports can be accessed via a single HBA port. This capability is useful in large rack mount implementations. The previous chapter detailed the hardware and software required to support the Port Multiplier solution.

This Chapter

A common goal of server implementations is non-stop operation. This typically includes some type of redundant capability that allows for failures to be managed without shutting down the entire system. Port Selectors provide access to SATA drives from two separate sources, thereby permitting a fail-over solution. This chapter discusses the Port Selector implementation and associated protocols.

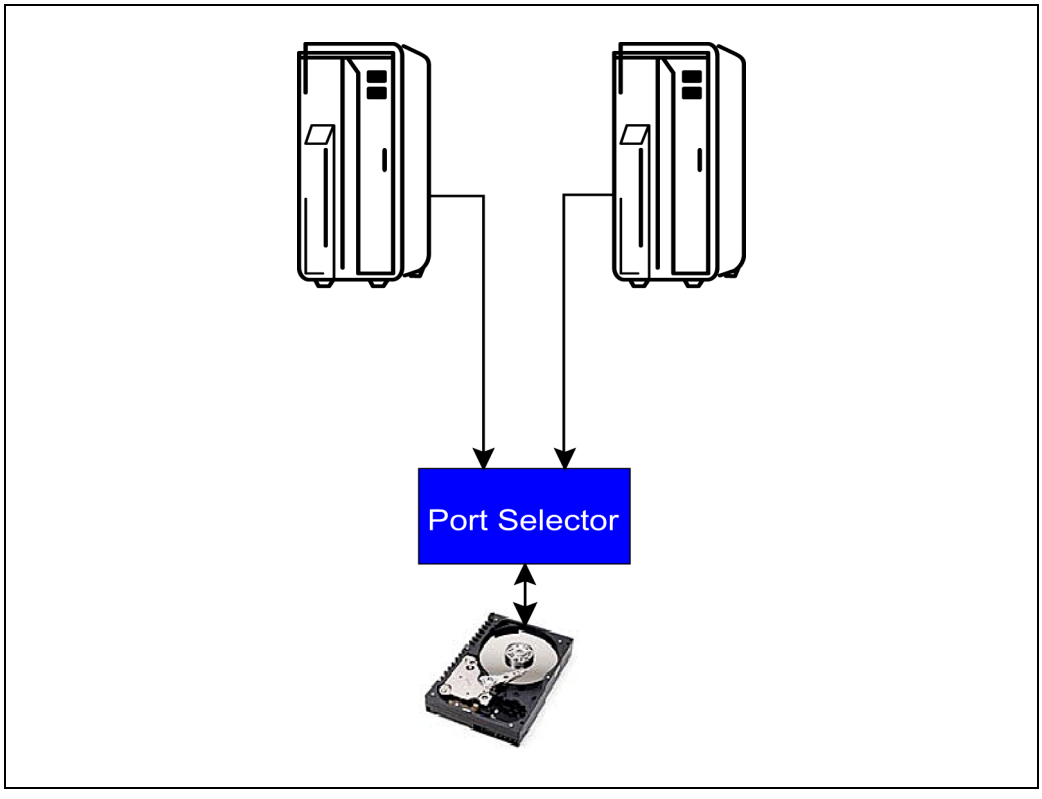
The Next Chapter

Large arrays of drives are typically housed in cabinets called drive enclosures. Managing the environment within an enclosure such as temperature, fan speed, power, etc., is critical to the health of the drive array. SCSI systems have long incorporated hardware and software to help manage drive enclosures. Two SCSI standards are supported by SATA II. This chapter highlights the support that has been added by SATA to ease the implementation of enclosure services.

Overview

Port Selectors provide a method for connecting drives to one of two different host ports. The goal of this mechanism is to provide a way for system designers to create redundant implementations. The typical implementation would likely employ a RAID solution in combination with the Port Selector capability. Figure 16-1 on page 288 illustrates the Port Selector implementation; whereby, a standard SATA drive can be accessed by separate HBA ports or Systems.

Figure 16-1: Port Selector



The overall design of the Port Selector was based on key design requirements including:

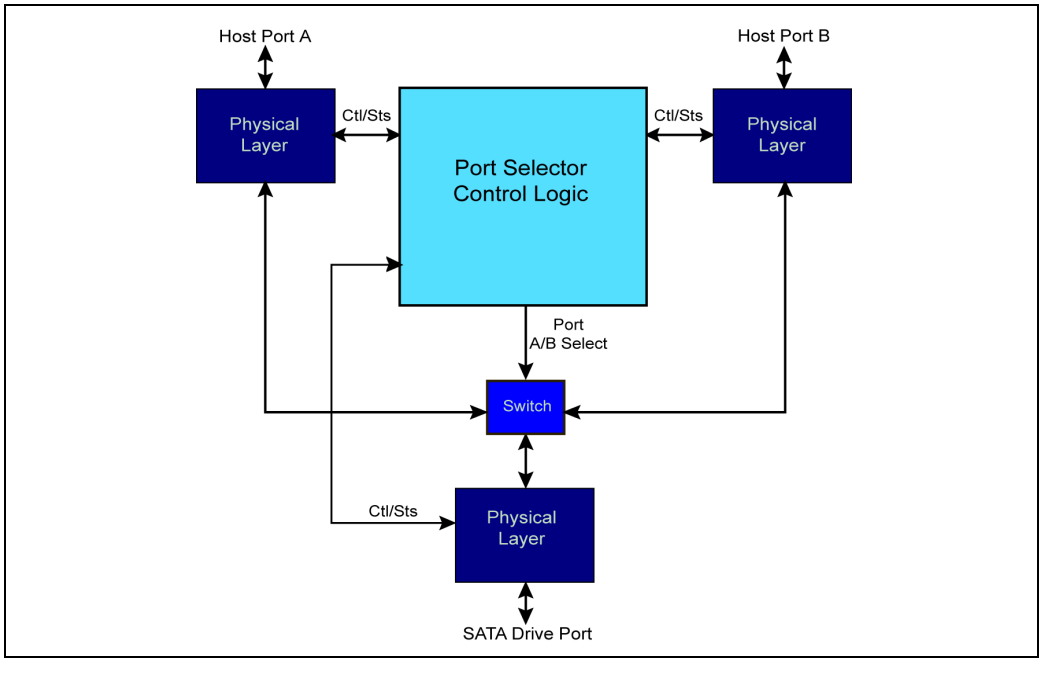
- No modifications to Serial ATA 1.0a compliant devices required
- No hardware modification required for Host Adapters (Some optional features do require modification)
- No new primitives needed
- No new FIS types needed
- Full-function link and transport layers not needed
- Support for only two host connections
- Only one port can be active at a time
- Port selectors cannot be cascaded
- New software required to manage fail-over

Port Selector Functions

Port selectors are switches that select which host port will be permitted access to the drive, and they also repeat frames between the HBA and drive. The block diagram in Figure 16-2 illustrates the primary element of a Port Selector. The following list describes the primary functions associated with a Port Selector:

- Port Selectors participate in the OOB (Out-Of-Band) sequence to establish link communications with both the active HBA and the attached drive.
- The HBA and Port Selector may optionally support a mechanism to detect the presence of a Port Selector based on OOB signaling.
- A Port Selector must support a mechanism that permits software to change the active port. The specification defines a protocol-based method of port selection and also permits side-band solutions (defined as implementation-specific).
- Port Selectors repeat frame and OOB traffic between the active host port and the drive, and in doing so must satisfy all of the protocol and timing requirements.

Figure 16-2: Simplified Block Diagram of a Port Selector



Detecting Port Selector Presence

Detecting whether a Port Selector is present is an optional capability and was defined for Port Selectors that support protocol-based port selection. Port Selectors may identify themselves via a variation of the standard OOB signaling sequence during link initialization. Normally the OOB sequence proceeds as illustrated in Figure 16-3; however, to identify itself as a Port Selector, a COMWAKE is inserted in front of the normal COMINIT after detecting COMRESET from the host (Figure 16-4). HBAs that are designed to detect the presence of Port Selectors will recognize this variation and update the SError register (Diag field) to indicate that a Port Selector is present. (See Figure 16-5 on page 291.)

Figure 16-3: Normal OOB Sequence

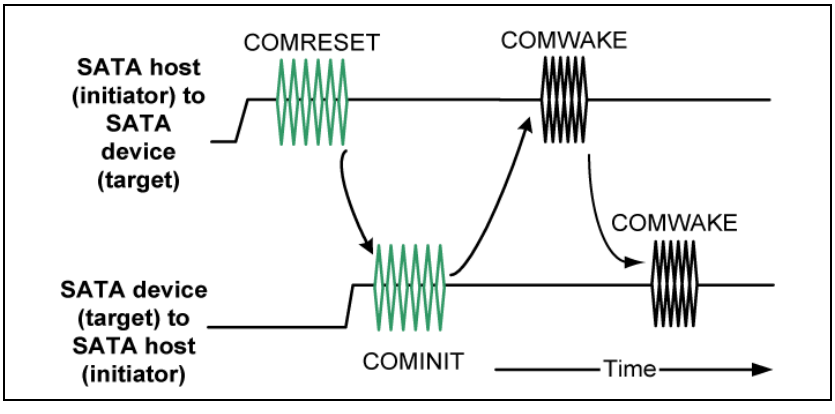


Figure 16-4: OOB Sequence Used for Port Selector Presence Detection

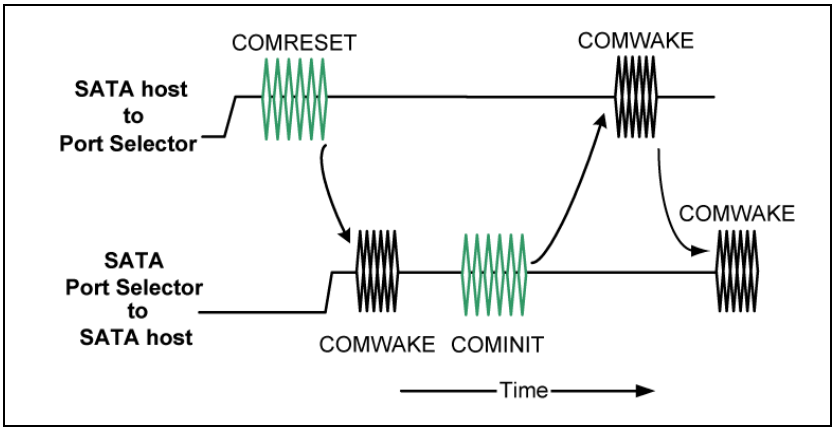
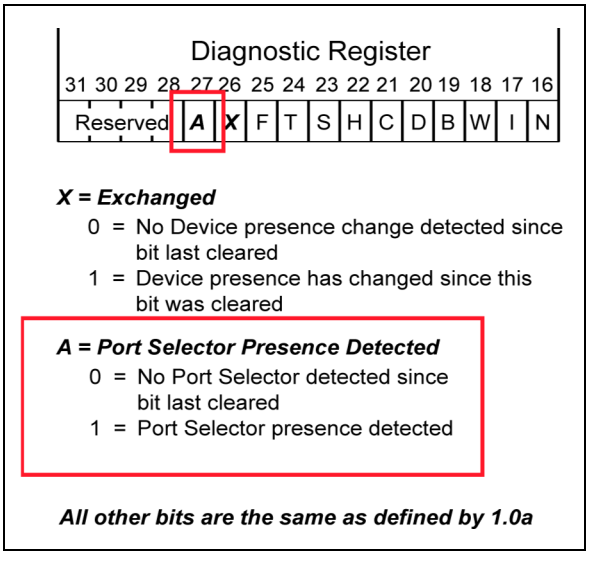


Figure 16-5: Port Selector Presence Bit



Port Selection

Two methods of port selection are defined for software to dynamically switch host ports. These methods include:

- Sideband signaling — this method requires that the Port Selector implement a pin that can be used to signal the Port Selector that the active port is to be switched. The specification considers this method to be implementation specific and provides a suggested implementation.
- Protocol-based switching — the specification defines a specific OOB (Out-OF-Band) signaling protocol that is sent to the inactive host port to notify the Port Multiplier to switch the active port.

One of these two methods must be employed but neither one is specifically required to be implemented.

17 Enclosure Services

Previous Chapter

A common goal of server implementations is non-stop operation. This typically includes some type of redundant capability that allows for failures to be managed without shutting down the entire system. Port Selectors provide access to SATA drives from two separate sources, thereby permitting a fail-over solution. The previous chapter discussed the Port Selector implementation and associated protocols.

This Chapter

Large arrays of drives are typically housed in cabinets called drive enclosures. Managing the environment within an enclosure such as temperature, fan speed, power, etc., is critical to the health of the drive array. SCSI systems have long incorporated hardware and software to help manage drive enclosures. The SCSI standards are supported by SATA II. This chapter highlights the support that has been added to ease the implementation of enclosure services.

The Next Chapter

The next chapter describes the hardware link initialization process. To begin communicating, link neighbors must go through a sequence of steps after reset to recognize that a proper device type is attached and is working at a supported speed. The process is relatively simple compared to other serial protocols because the devices have defined roles: one acts as the host/initiator while the other behaves as the target device. New device types added with the SATA II definition have necessitated some new initialization protocols as well.

Overview

This chapter provides an overview of the primary features associated with the enclosure management features included in the SATA II specification. This chapter does not detail the protocols and additional information defined by the two industry standards that the SATA II specification supports. These two stan-

dards (listed below) provide support for managing drive enclosures:

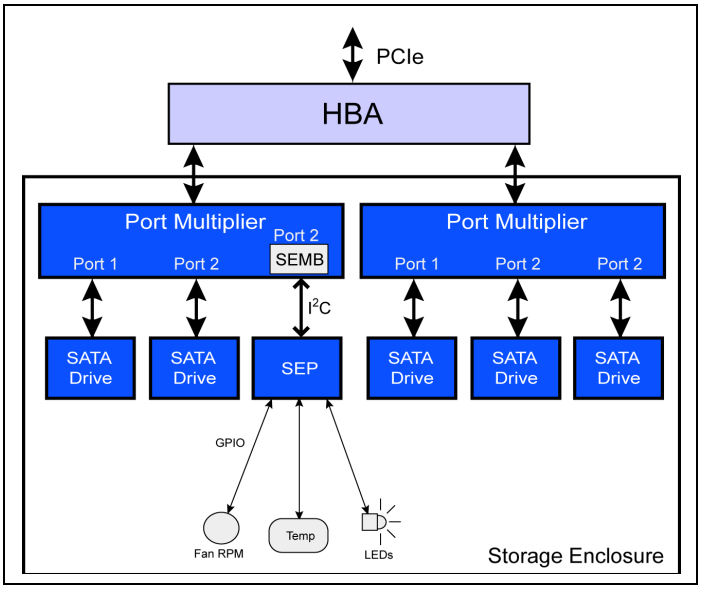
- SAF-TE (SCSI Accessed Fault-Tolerant Enclosures)
- SES (SCSI Enclosure Services)

These SCSI specifications define the services for managing storage subsystems, and can also be applied to SATA Storage subsystem. The elements involved in Storage enclosure management are illustrated in Figure 17-1. The specific implementation illustrated shows a portion of the Storage Enclosure function embedded within a Port Multiplier and accessible by the HBA via port 2. The Storage Enclosure elements include:

SEP (Storage Enclosure Processor) — This device interfaces with various sensors and indicators within a storage enclosure subsystem and responds to enclosure service commands. The SEP may support both the SAF_TE and SES protocols. The required interface to the SEP is the I2C bus, over which the commands and data are passed.

SEMB (Storage Enclosure Management Bridge) — This device is the interface (bridge) between the SATA host and the Storage Enclosure Processor (SEP). SEP commands are delivered via the standard SATA programming interface and sent to the SEMB. The SEMB converts the SATA commands into I^2C packets that are then delivered to the SEP.

Figure 17-1: Enclosure Management Elements



Enclosure Services

This section briefly describes the two industry standards that provide service for managing storage enclosures, and discusses the relationship between them.

SAF-TE (SCSI Attached Fault-Tolerant Enclosure)

The SAF-TE (SCSI Attached Fault-Tolerant Enclosure) was implemented as an inexpensive SCSI target that provided enclosure control and monitoring. The target is called an SEP (SCSI Enclosure Processor) and only acts in a target role. It does not include asynchronous notification of events and requires periodic polling to detect changes. It allows for access to enclosure information such as number and status of fans, power supplies, sensors, etc.

The problem with the SAF-TE specification was that it was not owned by any existing standards body and consequently was not extensible. In the mid 1990s the SAF-TE working group agreed to merge the architecture with the SCSI ESI (Enclosure Services Interface) definition to create a single standard called SES.

SES (SCSI Enclosure Services)

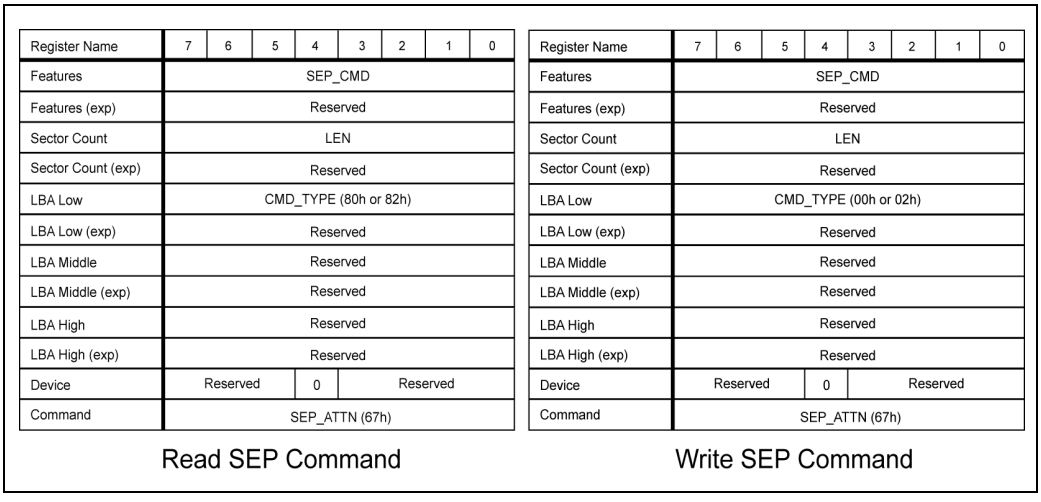
The SES (SCSI Enclosure Services) command set is used to communicate with a separate enclosure service processor intended to be implemented in high-end enclosures for sensing and managing resources that are relevant to an enclosure, such as fans, temperature sensors, and LEDs. The SES command set defines a series of Diagnostic Pages to allow devices to communicate with the enclosure processor over SCSI. The SCSI commands for this purpose are SEND DIAGNOSTIC and RECEIVE DIAGNOSTIC RESULTS.

SATA Storage Enclosure Services Commands

SATA defines a standard interface for software to pass enclosure commands to SEMB devices via the Shadow Registers. The SEMB passes these commands to the SEP via the I^2C interface. Software may use either the SAF-TE or SES command protocol to communicate with the SEP. Because these protocols are in widespread use, implementers can leverage existing software and hardware.

SATA II defines the new Read SEP/Write SEP commands to originate the enclosure management commands. Figure 17-2 on page 298 illustrates the Shadow register field definitions when performing both the Read and Write SEP commands.

Figure 17-2: Format and Field Definitions for SEP Commands



Register Name	7	6	5	4	3	2	1	0
Features	SEP_CMD							
Features (exp)	Reserved							
Sector Count	LEN							
Sector Count (exp)	Reserved							
LBA Low	CMD_TYPE (80h or 82h)							
LBA Low (exp)	Reserved							
LBA Middle	Reserved							
LBA Middle (exp)	Reserved							
LBA High	Reserved							
LBA High (exp)	Reserved							
Device	Reserved			0	Reserved			
Command	SEP_ATTN (67h)							

Read SEP Command

Register Name	7	6	5	4	3	2	1	0
Features	SEP_CMD							
Features (exp)	Reserved							
Sector Count	LEN							
Sector Count (exp)	Reserved							
LBA Low	CMD_TYPE (00h or 02h)							
LBA Low (exp)	Reserved							
LBA Middle	Reserved							
LBA Middle (exp)	Reserved							
LBA High	Reserved							
LBA High (exp)	Reserved							
Device	Reserved			0	Reserved			
Command	SEP_ATTN (67h)							

Write SEP Command

Table 17-1 lists and defines the fields associated with the SEP commands.

Table 17-1: Register Content and Descriptions for SEP Commands (Reserved Regs not shown)

Shadow Reg	Field Name	Description
Features	SEP_CMD	SEP Command — Defines parameters related to the SES and SAF_TE command protocol.
Sector Count	LEN	Length — defines the length of the data payload
LBA Low	CMD_TYPE	Command Type — encoding as follows: Upper nibble bits (7:4): 0 = SEP to Host Data Transfer 8 = Host to SEP Data Transfer Lower nibble bits (3:0): 0 = SAF-TE protocol used 2 = SES protocol used
Device	NA	Value specified is listed in Figure 17-2, above.
Command	SEP_ATTN	SEP Attention Command (67h) — indicates that the SEMB that a SEP command is being issued

Topologies

The only topology presented in this chapter is illustrated in Figure 17-1 on page 296, where the SEMB resides within a Port Multiplier and the SEP is attached to an I^2C port provided by the Port Multiplier. The specification provides examples of several other topologies that simply illustrate that the location of the SEMB and SEP can vary widely depending on the storage application.

18 SATA Initialization

The Previous Chapter

The last chapter highlighted the support that has been added to ease the implementation of enclosure services. Large arrays of drives are typically housed in cabinets called drive enclosures. Managing the environment within an enclosure such as temperature, fan speed, power, etc., is critical to the health of the drive array. SCSI systems have long incorporated hardware and software to help manage drive enclosures. The SCSI standards are supported by SATA II.

This Chapter

This chapter describes the hardware link initialization process. To begin communicating, link neighbors must go through a sequence of steps after reset to recognize that a proper device type is attached and is working at a supported speed. The process is relatively simple compared to other serial protocols because the devices have defined roles: one acts as the host/initiator while the other behaves as the target device. New device types added with the SATA II definition have necessitated some new initialization protocols as well.

The Next Chapter

The next chapter describes the physical layer logic and the electrical characteristics of the SATA interface. The chapter specifically discusses the electrical characteristics of the differential transmitter and receiver used to transmit the bits across the wire. It also covers the changes in the electrical signaling as different interface environments are used.

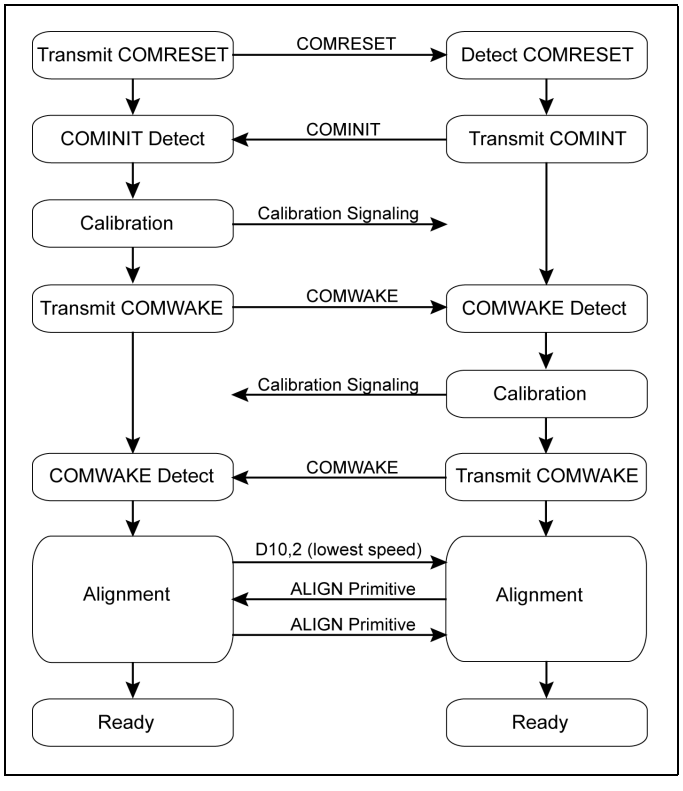
Introduction

After reset, the hardware in devices on both ends of a link automatically begins the process of detecting whether another device is present and at what speed the interface can run. This initialization takes place using OOB (out of band) sig-

naling even before the devices are able to recognize dwords as information from the transmitter. OOB uses the same signal path as every other bit from the transmitter, meaning they are not side-band signals, but occurs before the link is trained. Once the link is trained, application software reads the signature returned by the target device in the first Device-to-Host Register FIS and determines the device type. Software can then write configuration information into the device registers and set up the desired parameters.

The overall process associated with link initialization is illustrated in the diagram in Figure 18-1. OOB signaling involves bursts of signal activity followed by periods of idle time. The timing of these sequences vary and have given names that in some cases designate whether the HBA or the SATA Device is sending a particular sequence. During the sequence the HBA and SATA Device engage in the handshake sequence shown below. Details regarding the process are discussed later in this chapter.

Figure 18-1: Overview of Link Initialization Sequence



The definitions and protocols associated with initialization are defined by the Phy Initialization state machine. This state machine supports Port Multipliers, Hot Plug, and speed negotiation. Authors of the standard did not use the conventional bubble diagrams to describe the state machine because they are difficult to modify and update regularly. Instead, they chose a table format, which is precise and easy to update but unfortunately not very good at giving an overall understanding of the behavior. They also chose later not to convert the tables into bubble diagrams so as to avoid the risk of introducing errors. Rather than attempting to redraw the state diagrams here and take the same risk of introducing errors in the details, the author believes the reader will be better served by a discussion of the overall behavior of both host and target devices as they go through the initialization process.

Hardware Initialization

General

SATA host and target devices have a fixed role in hardware initialization because they only expect to be attached to each other. This makes the process simpler than it would be if there were other configurations possible. There are two basic steps of hardware initialization:

1. Detect the presence of an attached device.
2. Negotiate the highest transmission rate supported by both devices.

The first step is accomplished using OOB signaling, in which both nodes send a pattern and then wait for a response. If the proper response is seen, the second step is to negotiate the transfer speed on the link by sending a steady stream of patterns from both nodes. Modifying and observing this pattern allows devices to negotiate a common speed, recover the transmitter's clock, and achieve dword synchronization. At that point devices can begin to detect valid symbols and meaningful dwords in the data stream.

The Physical Plant

The SATA standard provides a suggested Phy implementation, referred to as the physical plant, and this is recreated in Figure 18-2 on page 307. This diagram is described more fully in the SATA 1.0a standard, but it's useful to discuss several of the definitions here in the context of initialization.

Physical Plant Signal Definitions

Align Source — refers to the ALIGN patterns sent for initialization during OOB, and for clock compensation after initialization. The receiver has squelch logic in the analog front end to detect the two OOB patterns. For more on these patterns, see "OOB (Out of Band) Signaling" on page 307.

PhyReset — causes the phy to initialize itself, after which a host will send COMRESET and a target device will send COMINIT.

PHYRDY — indicates that the phy has succeeded in establishing communications. This means that OOB and speed negotiation worked properly, and Dword synchronization has been achieved.

SPDSEL — a vendor-specific signal instructing the phy to negotiate the speed or to set a particular speed.

SPDMODE — a vendor-specific signal indicating the current speed setting.

COMMA — indicates that a K28.5 value was detected in the incoming data stream, identifying the beginning of both a 10-bit symbol and a 40-bit primitive.

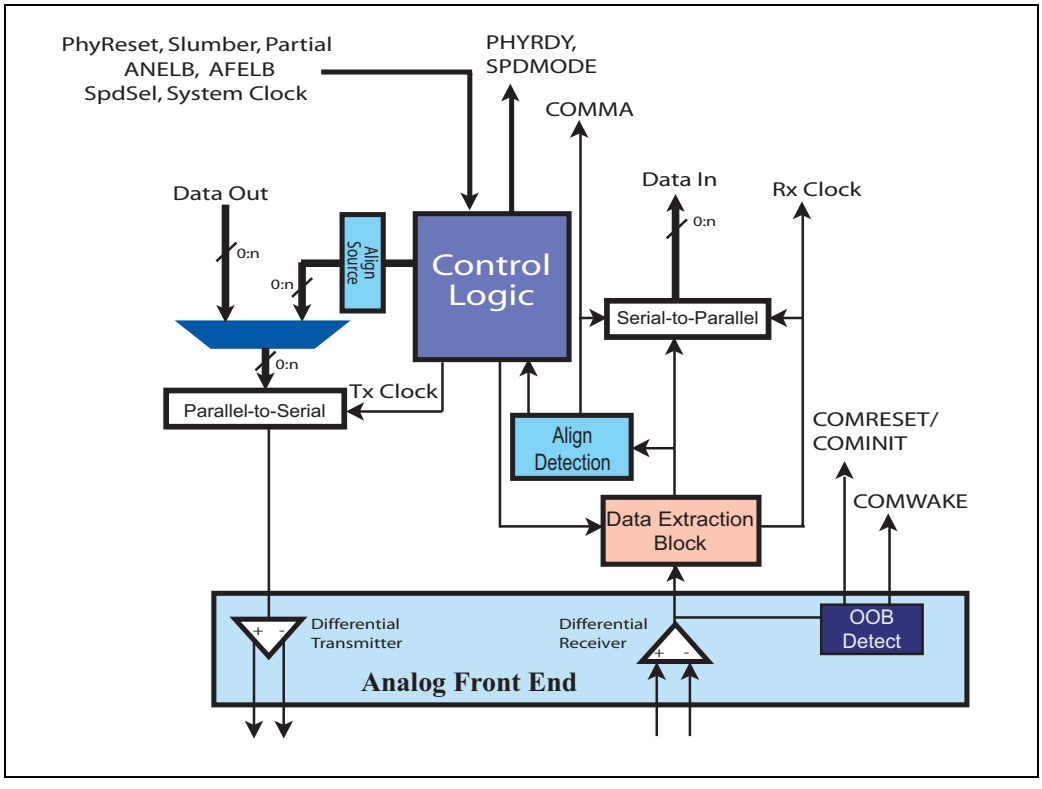
COMINIT/COMRESET and COMWAKE — outputs of the squelch logic in the analog front end that report the detection of these OOB patterns to the state machines and higher level logic.

Slumber/Partial — Link power management states that reflect the degree of power savings.

ANELB — Test feature that places the near end (local) transceiver into to a silicon loopback where the transmitter is looped back to the receiver.

AFELB — Test feature that places a far end (remote) transceiver into a silicon loop back mode where the receiver is looped back to the transmitter. Two version of this loopback exist: one that is retimed and one that is not.

Figure 18-2: Physical Layer Detailed Block Diagram



OOB (Out of Band) Signaling

Before the link has been trained devices cannot recognize bit patterns in the signal and are unable to communicate. Early in the SATA definition the designers considered a simple OOB that used steady-state signals: TX+ high meant one thing and low meant another, for example. However, when the AC-coupled solution was added later, that method no longer worked because the coupling capacitor blocked that DC signal, so a "chirp" method was developed instead. To accomplish this the transmitter alternates sending repeated ALIGN primitives (the chirp) followed by periods of no activity. Even though bit patterns can't be understood, a receiver can still detect that there is activity on the link by using a squelch-detect circuit. The timing of the pattern of activity and idle conveys the needed information.

19 Physical Layer

The Previous Chapter

The previous chapter described the hardware link initialization process. To begin communicating, link neighbors must go through a sequence of steps after reset to recognize that a proper device type is attached and is working at a supported speed. The process is relatively simple compared to other serial protocols because the devices have defined roles: one acts as the host/initiator while the other behaves as the target device. New device types added with the SATA II definition have necessitated some new initialization protocols as well.

This Chapter

This chapter describes the physical layer logic and the electrical characteristics of the SATA interface. The chapter specifically discusses the electrical characteristics of the differential transmitter and receiver used to transmit the bits across the wire. It also covers the changes in the electrical signaling as different interface environments are used. This includes internal cabling, external cabling, backplane environments.

The Next Chapter

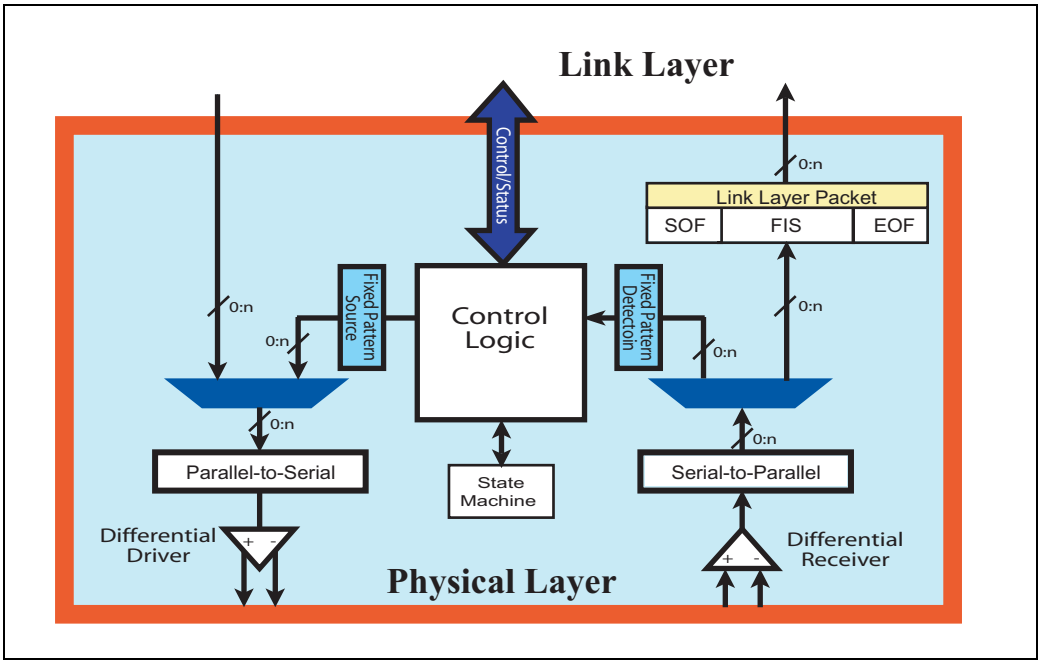
The next chapter covers the various form factors, cables and connectors that are described in the SATA specifications. The SATA II specification added several new cable and connector types in support of SATA-based server implementations.

Introduction

After all the work has been done to create a FIS for transmission, the Physical Layer, sometimes called just the Phy, is the final piece of logic that this packet will pass through. This layer resides hierarchically at the bottom of the SATA layered architecture (see Figure 19-1 on page 326) and can be described in two parts. The first part represents the logic needed to prepare the 10-bit encoded data for transmission or to recover it properly at the receiver. The second part

consists of the high-speed electrical interface and is made up of the differential transmitter, receiver, and transmission line.

Figure 19-1: Physical Layer



Logical Interface

It should be noted that the SATA standard does not require a particular implementation, specifying instead the behavior that must be carried out by each layer. The responsibilities of the logical part of the physical layer on the transmit side include:

- Serialize the data for transmission
- Insert primitives for clock management
- Initialize the link using OOB (described in the Initialization chapter)

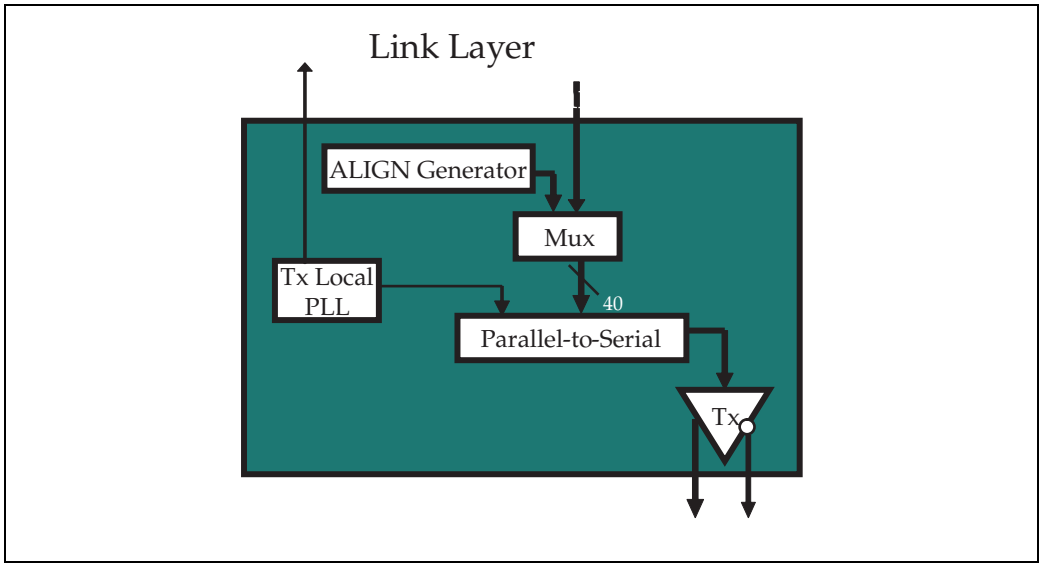
The responsibilities on the receive side are:

- Data and Clock recovery
- Manage the elasticity buffer for clock skew compensation
- Detect OOB signals and speed negotiation primitives (described in the Initialization chapter)
- Deserialize the bitstream
- Detect the K28.5 Comma value and align the parallel output with it
- Optionally support power management modes

Transmit Side

Each SATA interface consists of a transmit side and a receive side, as shown in Figure 19-2. For normal transmission, the Physical Layer takes the output from the Link layer and it's expected that many vendors will implement the transmit path as four bytes wide, as shown here, to simplify the design. This would also mean that the internal clock would only need to be 1/40th of the transmit clock, so if the serial data rate is 3.0 Gb/s, then the internal dword rate would be 75 MHz.

Figure 19-2: Physical Layer Transmit Block Diagram



Selecting Frames or OOB Primitives

A mux is shown on the transmit side because ALIGN primitives will be needed for initialization before the normal data path will be useful. The Physical Layer generates and injects these primitives for the OOB sequence and link initialization automatically after the device is reset. The output from the mux passes to the serializing buffer, which takes in a dword and shifts the bits out serially to the differential transmitter based on the transmit clock frequency that was selected during speed negotiation on the link.

Reducing EMI by Using SSC

The data is clocked out of the serializer using the internal transmit clock running at either 1.5 GHz or 3.0 GHz. The clocks must be accurate to within +/- 350 ppm, for a total maximum separation of 700 ppm. The use of SSC (Spread Spectrum Clocking) adds another 5000 ppm, resulting in a total difference between the clocks that could be as high as 5,700ppm. SSC varies the clock over a range of frequencies rather than keeping it fixed as a pure reference clock, for the purpose of reducing EMI (electro-magnetic interference). For that reason, it is commonly used in devices that operate in consumer environments where stricter EMI requirements must be met. While varying the operating frequency does not actually reduce the overall radiated energy of the device, it spreads the energy across a wider range of frequencies and therefore reduces the peak output that may be seen at any one given frequency. Since governmental emissions tests like those used in the U.S. test for energy thresholds at certain frequencies, the goal is to ensure that none of the frequencies has so much radiated energy that it would fail the threshold test. SATA is intended for use in consumer environments and so SSC is optional for SATA transmitters, while SATA receivers must tolerate it.

The SATA standard specifies that the clock period is never allowed to go below the minimum value specified, meaning the frequency shift can only be frequencies that are less than or equal to the nominal rate (referred to as "down-spreading"). The method of frequency modulation to be used is not specified, but an example is shown in the standard using a sawtooth pattern as reproduced here in Figure 19-3. In that example, the frequency ranges from its nominal value to 0.5% less than that and back again at a rate between 30 and 33 KHz (modulation period between 30 and 33ns).

Figure 19-3: Sawtooth Wave SSC Frequency Modulation

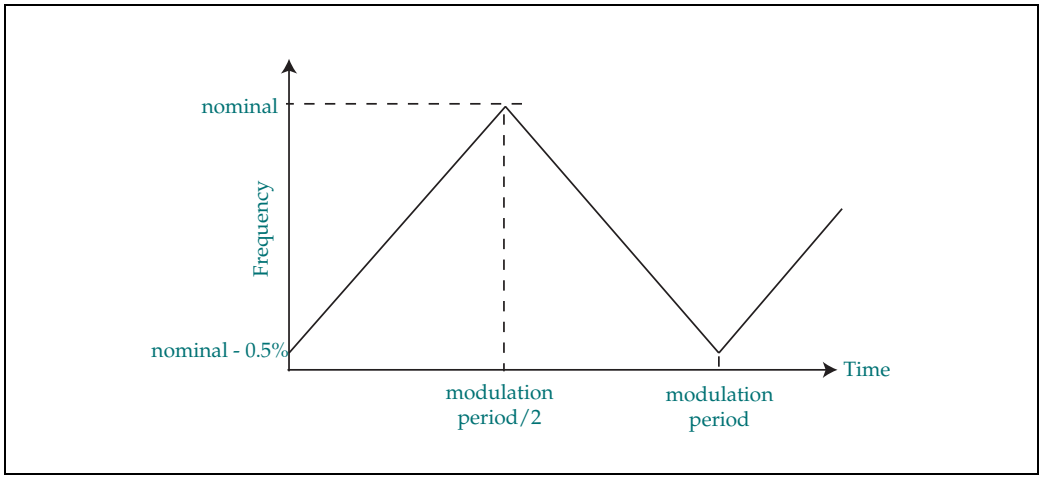
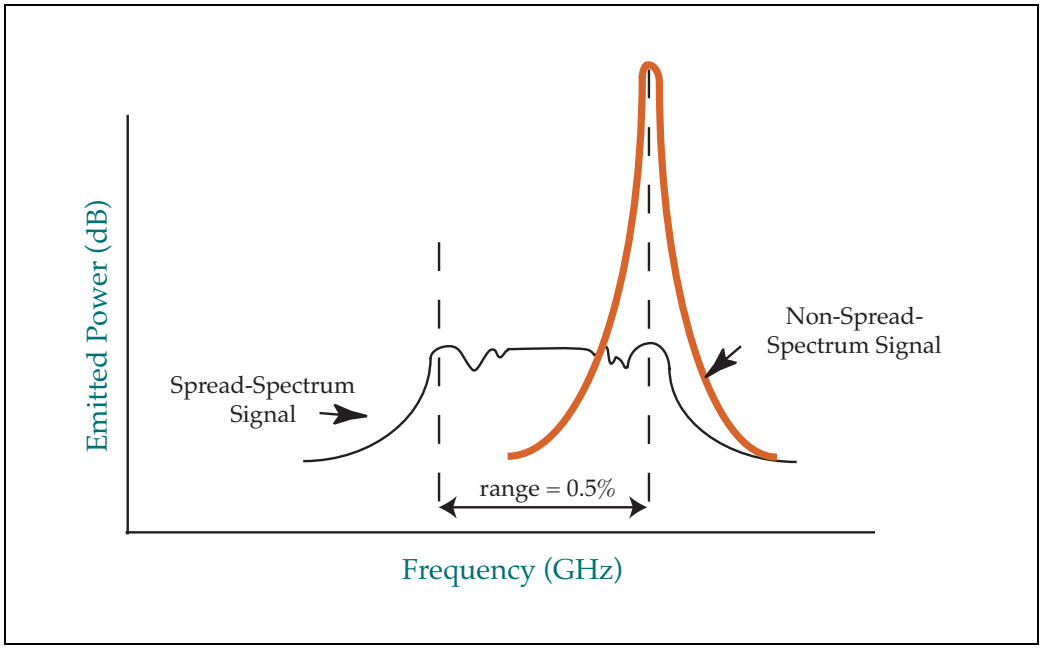


Figure 19-4: Results of SSC in Frequency Analysis



20 Cables & Connectors

The Previous Chapter

The previous chapter described the physical layer logic and the electrical characteristics of the SATA interface. The chapter specifically discusses the electrical characteristics of the differential transmitter and receiver used to transmit the bits across the wire. It also covers the changes in the electrical signaling as different interface environments are used. This includes internal cabling, external cabling, backplane environments.

This Chapter

This chapter covers the various form factors, cables and connectors that are described in the specification. The SATA II specification added several new cable and connector types in support of SATA-based server implementations. Also, the particular requirements associated with a given cable/connector type are discussed.

The Next Chapter

The SATA specifications add several key features intended to enable hot pluggable drive platforms. These features include a Hot Plug connector definition, detection of drive insertion and removal, asynchronous signal recovery, and more. The next chapter discusses the requirements necessary for SATA Hot Plug compliance.

Introduction

SATA's mechanical specifications define each connector (plug and receptacle) in detail, but the cable construction is purposely not specified. This allows manufacturers to make trade-offs between performance and cost; thereby giving the flexibility to design for different market segments.

Many of the original SATA cable and connector characteristics were based on the shortcoming and difficulties associated with the ATA implementation. The

SATA connections are superior to ATA in a variety of ways, including:

- smaller and thinner
- easier to manage
- smaller cables promotes better airflow through the system, potentially reducing overall system cost (e.g., fewer/smaller fans).
- blind connector mating capability
- use of edge contacts rather than pins that were prone to bending and breaking
- longer cables (up to one meter in length)
- hot pluggable connector at the drive

The specification states the following design goals for SATA connections:

- Support data rates of 1.5 Gbps and 3.0 Gbps. Clearly, supporting higher speeds would also be desirable since higher bit rates are likely in the future.
- Cost competitive with Ultra ATA. Cost is always an issue for the consumer market that was the first target for SATA, and the cost must be comparable with the cost of ATA.
- Facilitate smooth migration from ATA to SATA, meaning at least equal, and preferably better, cost and performance.
- Provide a common interface for both 2.5-inch and 3.5-inch devices

Usage Models and Form Factors

Cabled connections and direct connect solutions are based on a variety of usage models and related form factors. This section is a brief review of several of these models:

- Internal 1 meter Cabled Host to Device — This is the typical PC desktop application in which a drive is connected by cable to an HBA port. This application is not intended for hot plugging and uses the Gen1i or Gen2i electrical definitions.
- Internal 4-lane Cabled Disk Arrays — Rather than routing four separate cables, as defined in the previous example, a 4-lane cable can be implemented. The HBA connection is a single connector that support the four lanes and the drive end of the cable separates into four single drive connectors. The drive side of the cable may also be a single connection that attaches to an internal proprietary backplane that may have an IC that ensures that the proper Gen1i/Gen2i signal levels are met. Otherwise the characteristics are the same as the single cable solution.
- System to Device External Interconnect (long) – These types of intercon-

nects are anticipated for external drive enclosures that are relatively close to the PC (no longer than 2 meters). The single lane cable must operate at both Gen1 and Gen2 speeds and the HBA must support The Gen1x/Gen2x electrical requirements. Note that a single device may be attached to the cable or a Port Multiplier and multiple-drive enclosure.

- Short Backplane to Device — This application is intended for small drive arrays, in which drives are placed into drive canisters that plug directly into a backplane. Hot plug may or may not be supported and the Gen1m or Gen2m electrical specifications are used.
- Long Backplane to Device — similar to the short backplane implementation, but supporting larger disk arrays and requiring Hot Plug support. Because of the much greater signal attenuation the HBA must use the Gen1x or Gen2x electrical definitions. In addition, the signal levels may be too low to be used; consequently, an intermediate IC may need to be implemented (such as a Port Selector).
- System to System Interconnects (very long) – These connections may be used in applications such data centers where large drive arrays are required. These cabled connections are much longer than a normal external PC application and require higher quality and typically support multiple lanes (four and eight lane cables are defined). The HBA electrical interface requires Gen1x/Gen2x signaling.

It is apparent that many other applications and solutions can be implemented beyond those listed above.

Cables and Connectors — A Review

Detailed mechanical and electrical requirements are defined in the SATA 2.5 specification. This section describes the basic connection types and does not attempt to provide the detailed information that is the province of the specification.

The SATA Device Connector

Hard drives today use either a 3.5-inch or 2.5-inch format. While the SATA signaling connector is the same in either case, the power interface can vary depending on its size. For 3.5 inch drives there is still room for a legacy power connector, which many of the early SATA drives used. On the 2.5 inch drives that's no longer true and the newer power connector must be used.

The drive connector is actually implemented as a plug as illustrated in Figure 20-1 on page 364. Drives can be either plugged directly into a receptacle mounted in a backplane or drive bay or may be connected to the HBA via a cable receptacle. Several features can be seen in this figure, including the separate power and signal portions of the connector. The signal portion contains only 7 pins made up of 3 ground pins and 2 differential signal pairs, one pair for transmitting and the other pair for receiving, while the power portion contains 15 pins. Table 20-1 on page 365 lists each pin by number and name and describes its function where necessary. Note also that the drive contact includes two lengths to support Hot Plug solutions. Refer to Chapter 21, entitled "Hot Plug," on page 375 for more information.

Figure 20-1: Internal Backplane Connectors

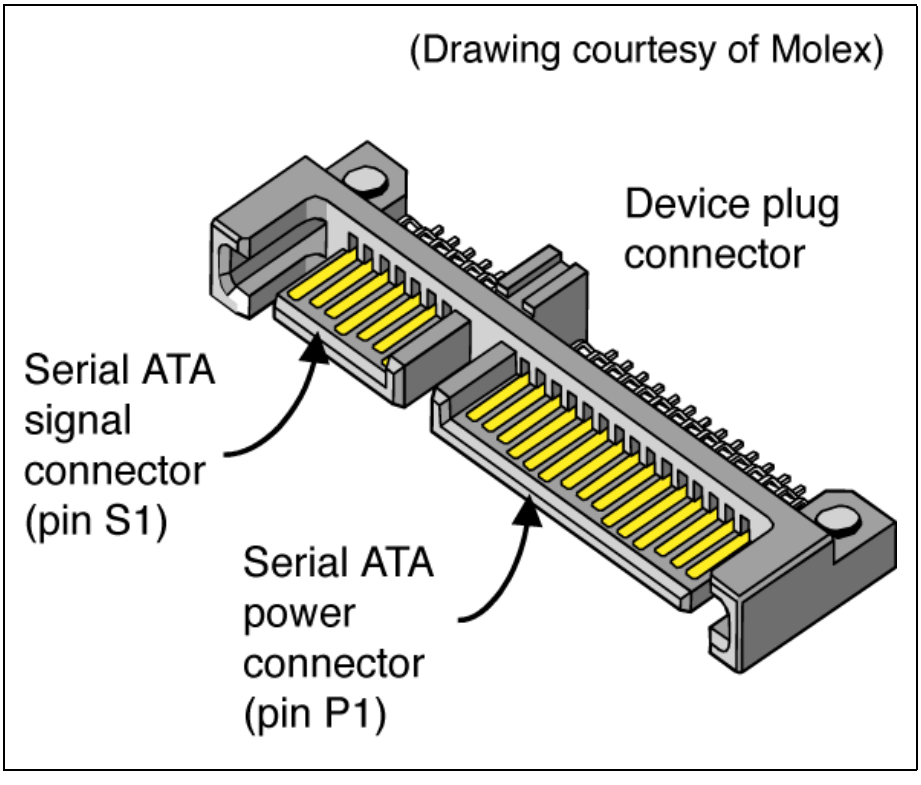


Table 20-1: SATA Drive Plug Contact Designation and Pinout

Contact	Name	Description
S1	GND	
S2	A+	Differential Signal Pair (A)
S3	A-	
S4	GND	Differential Signal Pair (B)
S5	B+	
S6	B-	
S7	GND	
Key (Slot between Signal and Power portion of the plug)		
P1	V_{33}	3.3V Power
P2	V_{33}	3.3V Power
P3	V_{33}	3.3V Power (Pre-charge)
P4	GND	
P5	GND	
P6	GND	
P7	V_5	5.0V Power (Pre-charge)
P8	V_5	5.0V Power
P9	V_5	5.0V Power
P10	GND	
P11	DAS/DSS	Device Activity Signal/Disable Staggered Spinup
P12	GND	
P13	V_{12}	12V Power (Pre-charge)
P14	V_{12}	12V Power
P15	V_{12}	12V Power

Internal cables and connectors

Platforms such as PCs may have entirely internal SATA drive connections. For this application the specification includes two primary connection types:

Direct-connects — a drive plugs directly into a receptacle within a drive bay or internal backplane.
Cable connections — a SATA cable has receptacles at both ends, thus providing the connection between drive plug and HBA plug.

21 Hot Plug

Previous Chapter

The previous chapter covers the various form factors, cables and connectors that are either described in the standard. The SATA II specification added several new cable and connector types in support of SATA-based server implementations.

This Chapter

The SATA specifications add several key features intended to enable hot pluggable drives. These features include a Hot Plug connector definition, detection of drive insertion and removal, and asynchronous signal recovery. This chapter discusses these and other features that support a Hot Plug solution.

The Next Chapter

SATA defines two levels of power conservation for the link, called Slumber and Partial. The next chapter describes the protocol associated with forcing the link into a low power state and recovering back to normal power and operation.

Overview

Whether Hot Plug is required or not, depends upon the particular platform. A wide variety of SATA Hot Plug solutions may be implemented. The specification defines three sets of electrical parameters based on the environment into which a SATA subsystem is implemented. The definition of these environments also determine whether the Host Controller must support the Hot Plug electrical requirements:

- Gen1i/Gen2i Implementations do not require Hot Plug support; however, if implemented in a short backplane implementation, Hot Plug capability is required.
- Gen1m/Gen2m Implementations require support for Hot Plug.
- Gen1x/Gen2x Implementations are required to support Hot Plug.

Platforms designed for Hot Plug-aware software may implement hardware that under software control can remove power from the connector prior to inserting or removing a drive. This chapter focuses on the features provided by SATA to assist in implementing Hot Plug platforms. The SATA specifications define the following key features that may be implemented in Hot Plug solutions:

- Asynchronous Signal Recovery (SATA II)
- Two Connector Contact Lengths to support Hot Plug Connectors
- In-rush Current Limiting Definition (SATA II)
- Drive Presence Detect (SATA II)
- Asynchronous Event Notification (SATA II)

Hot Plug Connector

Perhaps one of the most fundamental elements of a Hot Plug solution is a connector that reduces the likelihood that hardware damage will result from plugging or unplugging a device while the interface is powered and active.

Two types of Hot Plug connectors are defined:

- Cable-based connectors have two contact points
- Backplane-based connectors have three contact points

When a Hot Plug operation is performed two possibilities exist:

1. Power is present at the connector — insertion and removal is considered a surprise event because software has no prior notice of the Hot Plug event. When the surprise insertion or removal occurs, software either polls registers to detect the event or hardware may notify software directly.
2. Software has previously been notified by the user that a Hot Plug operation is requested. Under software control power is removed prior to the drive being removed and replaced.

The specification supports either of these types of Hot Plug operation.

Cable Hot Plug Connection

A SATA device connector has short and long contacts and the cable connector has all short contacts as illustrated in Figure 21-1 on page 377. This results in only two mating points. Table 21-1 on page 377 lists the mating order of the contacts. The specification does not consider the cable connection to be Hot Plug

compliant, in that it is not qualified to support insertion and removal while power is applied.

Figure 21-1: Pin Lengths and Mating for Cable Hot Plug Connection

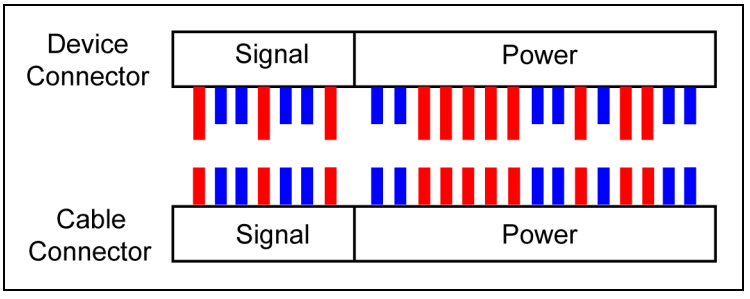


Table 21-1: Contact Mating When Drive is Plugged into Receptacle

Contact Combinations	Mating Points	Contact Assignments
Long to Short	First	3.3vdc Precharge Contact 5vdc Precharge Contact 12vdc Precharge Contact All ground Contacts
Short to Short	Second	Other 3.3vdc Contacts (2) Other 5vdc Contacts (2) Other 12vdc Contacts (2) Device Activity/Staggered Spinup disable Differential Signal Pairs (4 contacts)

Backplane Hot Plug Connector

Figure 21-2 illustrates the contact points of a backplane-based Hot Plug connector. Notice that the illustration shows the standard short and long contacts at the device connector and two long contacts at the host receptacle. The layout of contacts provide three mating points as a drive is plugged or unplugged from the backplane receptacle. Please also see Table 21-2.

Figure 21-2: Pin Lengths and Mating for Backplane Hot Plug Connections

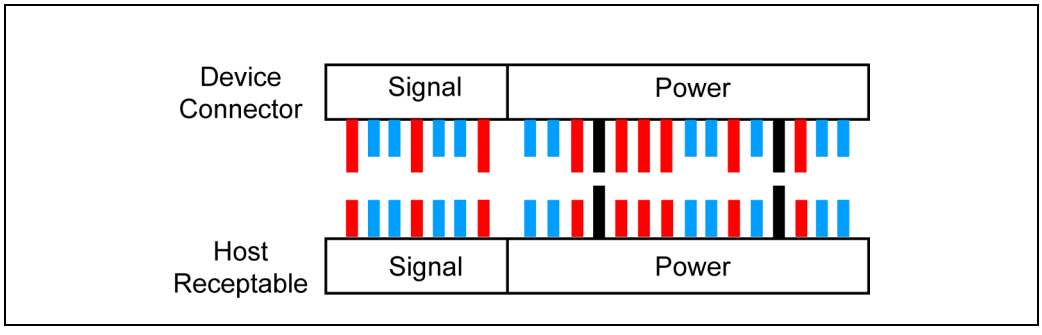


Table 21-2: Contact Mating When Drive is Plugged into Receptacle

Contact Combinations	Mating Points	Contact Assignments
Long to Long	First	Two Ground Contacts
Long to Short	Second	3.3vdc Precharge Contact 5vdc Precharge Contact 12vdc Precharge Contact All Other Ground Contacts (6)
Short to Short	Third	Other 3.3vdc Contacts (2) Other 5vdc Contacts (2) Other 12vdc Contacts (2) Device Activity/Staggered Spinup disable Differential Signal Pairs (4 contacts)

Drive Plug/Unplug Detection

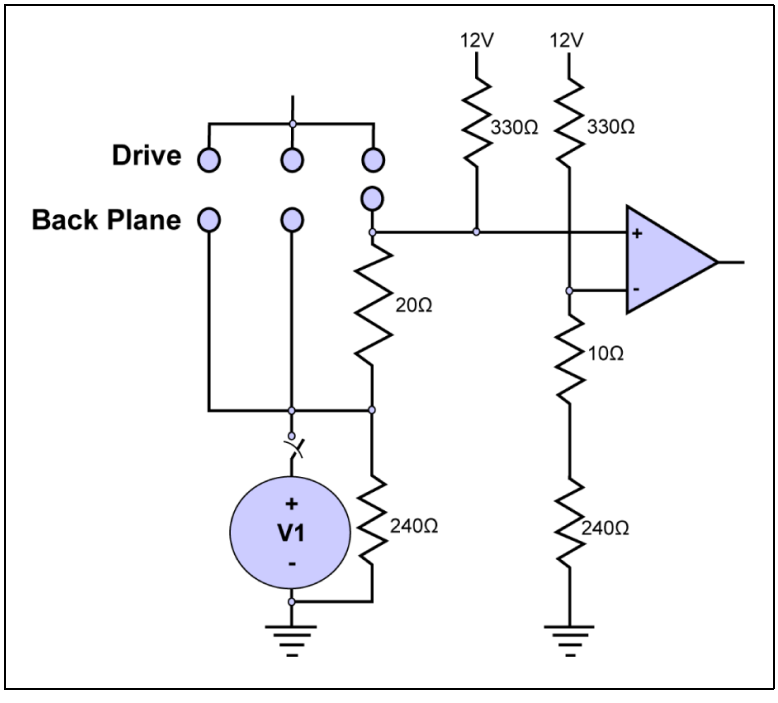
The ability to detect the insertion or removal of a drive can be implemented by a Hot Plug compatible host or PM device port interface. Figure 21-3 illustrates an example circuit for detecting drive attachment and removal, which is provided by the specification.

To support hot insertion and removal, the specification requires that SATA devices bus the power contacts together for each supply voltage as is illustrated in Figure 21-3. The contact assignments are:

- P1, P2, and P3 -- 3.3V power contacts
- P7, P8, and P9 -- 5V power contacts
- P13, P14, and P15 -- 12V power contacts

When device attachment or removal is detected, the host interface sets the "X" bit in the SError register's DIAG field (Figure 21-4).

Figure 21-3: Example Drive Attachment/Detachment Detection Mechanism



22 Link Power Management

Previous Chapter

The SATA specifications add several key features intended to enable hot pluggable drives. These features include a Hot Plug connector definition, detection of drive insertion and removal, and asynchronous signal recovery. The previous chapter discussed these and other features that support a Hot Plug solution.

This Chapter

SATA defines two levels of power conservation for the link, called Slumber and Partial. This chapter describes the protocol associated with transitioning the link into a low power state and recovering back to normal power and operation.

The Next Chapter

A variety of features built into the SATA subsystem assist in diagnosing problems, ensuring compliance, and validating proper operation of the SATA interface. The next chapter discusses these Built In Self Tests (BIST) some of which are required, while others are optional.

Overview

Like other serial interfaces based on 8b/10b encoding, SATA Phys are normally in the Phy Rdy state and are continuously sending and receiving information to keep the interface in the communicating state. Even during long periods of disk inactivity the SATA interface continues to send SYNC primitives (logical idle) to keep the interface ready to handle commands with very little latency when they are issued. Of course, the downside to this approach is that power continues to be consumed even when no work is being performed. The SATA specification defines an optional mechanism for placing the interface into the electrical idle state to conserve power and back into the normal operating state.

SATA defines two levels of link power conservation where the Phy is powered, but is dissipating less power than when in the PhyRdy state.

- Partial — Link power consumption is reduced by some vendor-specific level and the exit latency back to the PhyRdy state must be 10µs or less.
- Slumber — Link power consumption is reduced by some vendor-specific level less than that of the Partial state, and the exit latency from this state must be no longer than 10ms. As in the Partial state a zero differential voltage must be maintained unless a transmitter is AC coupled.

Even though the specification does not define a specific amount of power savings, the vendor may (the spec. says "should") publish the "not to exceed" power when in the Partial and Slumber states.

Configuring Link Power Management

Because link power management is an optional feature, the host implementation may or may support power management and the same is true of the drive. Whether a SATA host system supports link power management will likely depend on the operation environment; the two extremes being, battery-powered notebook computers where power saving is crucial and server-based platforms where the emphasis in on non-stop operation and low-latency disc access.

Detecting/Enabling Drive Link Power Management

Whether a drive supports the Partial or Slumber states is reported in the Identify Drive data. A drive may support:

- receipt of Host-initiated commands to enter Link Power Management
- initiation of commands that place the link into the partial or slumber states

Table 22-1 on page 385 lists the Identify Data information related to link power management, including its Link Power Management capabilities and whether these features are enabled.

Drives that report they can receive requests to enter the Partial or Slumber states are enabled to do so by default. However, drives that report the ability to initiate entry into a low-power link state are disabled to do so by default. Software must enable this feature via the SetConfiguration command.

Table 22-1: Identify Data Defines Drive's Support for Link Power Management

Word Offset	R/O	Description
76		**Serial ATA Capabilities** 15-11 Reserved 10 Phy event counters supported ***9 Receiving Link Power Management requests supported*** 8 Native Command Queuing supported 7-4 Reserved 3 Reserved for SATA 2 Supports Serial ATA Gen-2 signaling 1 Supports Serial ATA Gen-1 signaling 0 Reserved (0)
77		**Reserved for SATA**
78	Optional	**Serial ATA Features** 15-7 Reserved 6 Software settings preservation supported 5 Reserved 4 In-order data delivery supported ***3 Drive initiation of Link Power Management requests supported*** 2 DMA Setup FIS Auto-Activate optimization supported 1 DMA Setup FIS non-zero buffer offsets supported 0 Reserved (0)
79	Optional	**Serial ATA Features Enabled** 15-7 Reserved 6 Software settings preservation enabled 5 Reserved 4 In-order data delivery enabled ***3 Drive initiates Link Power Management requests*** 2 DMA Setup FIS Auto-Activate optimization enabled 1 DMA Setup FIS non-zero buffer offsets enabled 0 Reserved (0)
80-255		**Defined in ATA/ATAPI-7**

Link Power Management Protocol

The protocol associated with Link Power Management uses the primitives listed in Table 22-2. As discussed in the previous section, both the HBA and Drive may be able to initiate the protocol that will place the link into the Partial or Slumber states. The protocol is the same regardless of who initiates the sequence. The final aspect of this protocol is returning the link to its fully powered and communicating state, which is initiated by COMWAKE OOB signaling.

Table 22-2: Primitives Used in Link Power Management Protocol

Primitive	Name	Description
PMREQ_S	Power Management Request to enter Slumber state	This primitive is sent continuously until PMACK or PMNAK is received. When PMACK is received, current node (host or device) will stop sending PMREQ_S and enter the Slumber power management state.
PMREQ_P	Power Management Request to enter Partial state	This primitive is sent continuously until PMACK or PMNAK is received. When PMACK is received, current node (host or device) will stop PMREQ_P and enter the Partial power management state.
PMACK	Power Management Acknowledge	Sent in response to a PMREQ_S or PMREQ_P when the receiving node is prepared to enter either partial or slumber state.
PMNAK	Power Management No Acknowledge	Sent in response to a PMREQ_S or PMREQ_P when the receiving node is not prepared to enter a low-power state or when power management is not supported.

Host Initiated Entry into Partial/Slumber

The protocol associated with entry into either the Partial or Slumber state is described in this section. Figure 22-1 on page 388 illustrates the handshake performed between the HBA and Drive that causes the link to enter a low-power state. The sequence of events are described below and assumes that the link is currently in the logical idle state (i.e., transmitting and receiving SYNC primitives) and that the power modes are supported.

1. The Application layer initiates requests for entering either the Partial or Slumber state. The specification does not define a mechanism that triggers the Application layer to issue a request and is left as implementation specific. For example the AHCI provides registers that give software the means to both direct the HBA to initiate request to enter a low-power link state, and to enable the HBA hardware to initiate entry into a link low-power state autonomously.
2. The Transport layer receives and forwards the Power Management Request to enter Partial or Slumber to the link layer.
3. The power management request causes the Link layer to enter its Partial or Slumber state (L_TPMPartial or L_TPMSlumber). In this state PMREQ primitives are sent to the Phy layer. PMREQ primitives are delivered continuously until the Power Management Acknowledgement (PMACK) or Power Management No Acknowledgement (PMNAK) primitives are received from the drive.
4. The HBA physical layer transmits the primitives across the physical link.
5. The drive's Phy layer forwards all link traffic to the Link layer where it is decoded.
6. Upon detecting the PMREQ primitives, the Link layer enters its PMOff state.
7. While in the PMOff state the Link layer sends PMACK primitives for transmission by the Phy layer. Note that a minimum of four and a maximum of 16 PMACK primitives must be sent, after which the Link layer enters the ChkPhyRdy state. When in the ChkPhyRdy state the Link layer signals the Phy to enter the Partial or Slumber state.
8. The Phy layer transmits the PMACK primitives to the HBA.
9. The Phy layer detects either the Partial or Slumber signal from the Link layer and transitions to the electrical idle state.
10. The HBA receiver continues forwarding primitives to the Link layer where they are decoded.
11. When the HBA Link layer detects PMACK primitives it transitions to the ChkPhyRdy state and the transmit side discontinues PMREQ transmission.

23 BIST Features

Previous Chapter

SATA defines two levels of power conservation for the link, called Slumber and Partial. The previous chapter described the protocol associated with transitioning the link into a low power state and recovering back to normal power and operation.

This Chapter

A variety of features built into the SATA subsystem assist in diagnosing problems, ensuring compliance, and validating proper operation of the SATA interface. This chapter details these Built In Self Tests (BIST) some of which are required, while others are optional.

Overview

The SATA specification defines several Built-In Self Tests that can be entered under either HBA or Drive control. These tests include:

- Far End Retimed Loopback
- Far End Analog Loopback (optional)
- Far End Transmit Only (optional)
- Near End Analog Loopback (optional)

These tests are intended to assist in a variety of testing environments including:

- Laboratory testing using conventional equipment
- Inspection testing
- Observation testing

These tests were not designed with in-system automated testing in mind.

BIST FIS

Entry into the test modes listed in the overview (except Near End Analog Loopback) requires a BIST (Built-In Self Test) Activate FIS be delivered from the Application layer of the node initiating the test. The device on the opposite end if the link (the far-end device) receives the BIST and enters the appropriate test mode. The specification does define the associated protocols and requirements for BIST. However, no method is specified for triggering the BIST Activate delivery. That is, there is no BIST command defined.

BIST Format and Contents

The BIST Activate FIS is defined as bidirectional; thereby allowing either the HBA or Drive to initiate entry into a particular BIST mode of operation. Figure 23-1 illustrates the content of the BIST Activate FIS and Table 23-1 describes each of the fields.

Figure 23-1: Format of the BIST Activate FIS

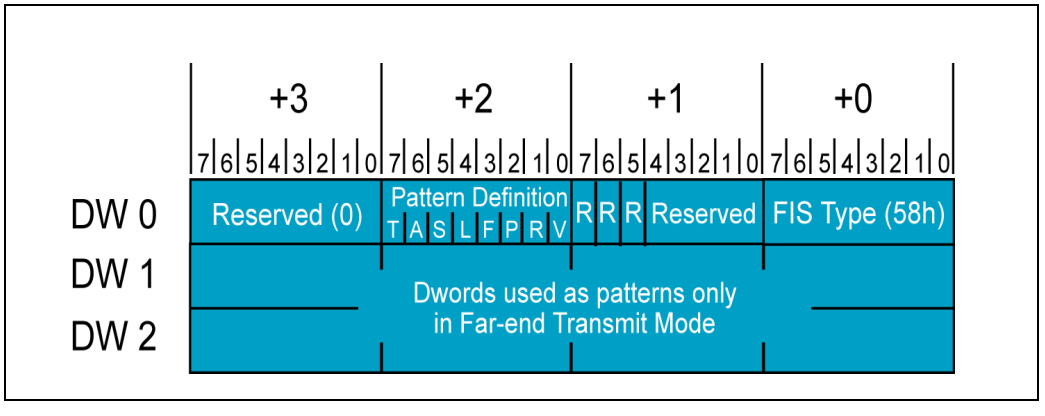


Table 23-1: BIST Activate FIS Field Descriptions

Field	Name	Description
FIS Type	FIS Type	A value of 58h identifies this FIS as a BIST Activate.
PM Port	Port Multiplier Port	This field allows a BIST Activate FIS to be directed to a drive attached to a Port Multiplier port and is only used by the HBA.
V	Vendor-Specific Test Mode	When this bit is set all other Pattern Definition bits must be ignored.
R	Reserved	Bit fields are always zero.
P	Primitive bit	Optional bit that is only valid when the T bit is set. When the P bit is set the DATA1 and DATA2 values are set up as primitives. The lower byte (D7:D0) of each DATA field contains a control (K) value followed by data values that define a valid primitive.
F	Far End Analog Loopback	This optional bit causes the node on the opposite end of the link to enter its analog loopback state (if supported). In this mode the raw data is received and retransmitted without clocking the data using the local receive clock and is intended to simply verify connectivity between the transmitting and receiving nodes.
L	Far End Retimed Loopback	This bit when set causes the node at the opposite end of the link to enter its Retimed loopback state. When data is received it re-clocks and re-transmits the data.
S	Scrambling Bypass	This bit may only be set when the "T" bit is set and must be ignored if the "P" bit is also set. The S bit is set when the node originating the Far End Transmit test wishes the data being returned to be left unscrambled.
A	ALIGN Bypass	This bit is only valid when the "T" bit is set. Setting the bit suppresses the delivery of ALIGN primitives. Otherwise ALIGN primitives are delivered periodically as required by the SATA protocol.
T	Far End Transmit Only	Setting this bit causes the Far End node to transmit repeatedly the contents of the DATA1 and DATA2 fields.

Table 23-1: BIST Activate FIS Field Descriptions

Field	Name	Description
DATA1	Dword One	This 4-byte field is only valid when the "T" bit is set and can be loaded with any value by the node that initiates the Far End Transmit Only mode.
DATA2	Dword Two	This 4-byte field is only valid when the "T" bit is set and can be loaded with any value by the node that initiates the Far End Transmit Only mode.

Table 23-1 makes apparent the requirement that only one test mode can be enabled at a time and that some bits have meaning only for a given BIST mode. The specification provides a clear explanation of the bit combinations that are legal. These combinations are presented in Table 23-2 on page 396 and the specified actions taken are summarized.

Table 23-2: Legal Pattern Definition Bit Combinations

BIST Mode	F	L	T	P	A	S	V
Far End Retimed Loopback	1	0	0	0	0	0	0
Far End Analog Loopback	0	1	0	0	0	0	0
Far End Transmit Only with ALIGN primitives and scrambled data	0	0	1	0	0	0	0
Far End Transmit Only with ALIGN primitives but without scrambled data	0	0	1	0	0	1	0
Far End Transmit Only without ALIGN primitives but with scrambled data	0	0	1	0	1	0	0
Far End Transmit Only without ALIGN primitives and without scrambled data	0	0	1	0	1	1	0
Far End Transmit primitives with ALIGNs	0	0	1	1	0	na	0
Far End Transmit primitives without ALIGNs	0	0	1	1	1	na	0
Vendor Specific BIST mode	na	na	na	na	na	na	1

BIST Transmission/Reception

BIST delivery is initiated by the Application layer in an application-specific manner. Transmission of the BIST FIS is intended to setup the test modes within both the near and far end nodes.

Far End Node Setup

When the far-end node receives the BIST FIS, it first verifies that the FIS has been received successfully and then returns the R_OK primitive to notify the transmitter that the BIS has been received. In addition the Application layer is notified of BIST FIS reception, evaluates the Pattern Definition and sets up the device to fulfill the BIST request. This setup is directed by the local application, which must place the transport, link, and physical layers into the states required for transmitting the requested data stream.

Near End Node Setup

Once successful transmission of the BIST FIS is received (R_OK), the near-end node should transmit SYNC primitives until it is ready to perform the specified operation. In addition, the near-end node notifies its Application layer that R_OK has been received from the far-end node. The local application then places its own Application, Transport, Link and Physical layers into the states necessary to deliver/receive the appropriate data stream.

Far End Retimed Loopback

This test is the only one whose implementation is required by both HBAs and devices. Entry into this test is course achieved by via the BIS FIS with the Pattern Definition field set as listed in Table 23-3 on page 398. Figure 23-2 on page 398 illustrates entry into the Far End Retimed loopback mode and the far-end node's actions. When in this state, data received by the far-end node is extracted by the receiver and associated logic as it does normally. The far-end node forwards the resulting parallel data to the transmit side where it is serialized and re-transmitted using its local clock. The Far-End Interface remains in the Far-End Retimed Loopback state until the originating node initiates a COMRESET sequence.

Appendix A

8b/10b Encoding Tutorial

8b/10b Encoding

The terminology used in this chapter is as follows:

- 8-bit values — data bytes and control bytes referenced by hexidecimal numbers.
- 10-bit values — data and control characters referenced using the Dxx.y and Kxx.y convention or using a 10-bit binary value.

General

The HBA and SATA device transmitters implement the 8-bit to 10-bit encoding mechanism that is widely used in high-speed serial bus implementations. The encoder converts 8-bit data and control bytes into 10-bit characters. The coding scheme is documented in the ANSI X3.230-1994 document, clause 11 (and also IEEE 802.3z, 36.2.4) and US Patent Number 4,486,739 entitled "Byte Oriented DC Balanced 8b/10b Partitioned Block Transmission Code"

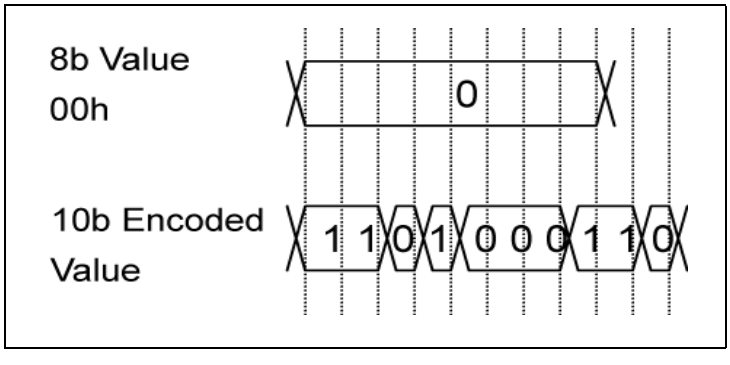
Purpose of Encoding a Byte Stream

The primary purpose of this scheme is to embed a clock into the serial bit stream that is typically transmitted using differential signaling. Consequently, this eliminates the need for a high frequency 1.5 or 3.0 GHz clock signal for the link, which would generate significant EMI noise and would be challenging to route on a standard FR4 board. Link wire routing between the HBA and SATA device is made much easier given that there is no clock to route, thus removing the need to match clock length to Lane signal trace lengths.

Below is a summary of the advantages of 8b/10b encoding scheme:

- **Embedded Clock.** Creates sufficient 0-to-1 and 1-to-0 transition density (i.e., signal changes) to facilitate re-creation of the receive clock on the receiver end using a PLL (by guaranteeing a limited run length of consecutive ones or zeros). The recovered receive clock is used to clock inbound 10-bit characters into an elastic buffer. Figure A-1 on page 410 illustrates the the conversion of 00h to 1101000110b, where an 8-bit value with no transitions converts to a 10b character with five transitions. These transitions keep the receiver PLL synchronized to the transmit circuit clock:
  - Limited 'run length' means that the encoding scheme ensures the signal line will not remain in a high or low state for an extended period of time. The run length does not exceed five consecutive 1s or 0s.
  - 1s and 0s are clocked out on the rising-edge of the transmit clock. At the receiver, a PLL can recreate the clock by sync'ing to the leading edges of 1s and 0s.
  - Limited run length ensures minimum frequency drift in the receiver's PLL relative to the local clock in the transmit circuit.

Figure A-1: Example of 8-bit Value of 00h Encoded to 10-bit character



- **DC Balance.** Keeps the number of 1s and 0s transmitted as close to equal as possible, thus maintaining DC balance on the transmitted bit stream to an average of half the signal threshold voltage. This is very important in capacitive- and transformer-coupled circuits.
  - Maintains a balance between the number of 1s and 0s on the signal line, thereby ensuring that the received signal is free of any DC component. This reduces the possibility of inter-bit interference. Inter-bit interference results from the inability of a signal to switch properly from one logic level to the other because the Lane coupling capacitor or intrinsic wire capacitance is over-charged.
- **Encoding of Special Control Bytes.** Permits the encoding of special control ('K') bytes that can always be recognized by the receiving node. These bytes are always used as the first character of each four character primitive.
- **Error Detection.** A secondary benefit of the encoding scheme is that it facilitates the detection of most transmission errors. A receiver can check for 'running disparity' errors, or the reception of invalid characters. Via the running disparity mechanism (see "Disparity" on page 413), the data bit stream transmitted maintains a balance of 1s and 0s. The receiver checks the difference between the total number of 1s and 0s transmitted since link initialization and ensures that it is as close to zero as possible. If it isn't, a disparity error is detected and reported, implying that a transmission error occurred.

The **disadvantage** of 8b/10b encoding scheme is that, due to the expansion of each 8-bit value into a 10-bit character prior to transmission, the actual transmission performance is degraded by 25% or said another way, the transmission overhead is increased by 25% (everything good has a price tag).

Properties of 10-bit (10b) characters

- For 10-bit character transmissions, the average number of 1s transmitted over time is equal to the number of 0s transmitted, no matter what the 8-bit value to be transmitted is; i.e., the character transmission is DC balanced.
- The bit stream never contains more than five continuous 1s or 0s (limited-run length).
- Each 10-bit character contains:
  - Four 0s and six 1s (not necessarily contiguous), or
  - Six 0s and four 1s (not necessarily contiguous), or
  - Five 0s and five 1s (not necessarily contiguous).

- Each 10-bit character is subdivided into two sub-blocks: the first is six bits wide and the second is four bits wide.
  - The 6-bit sub-block contains no more than four 1s or four 0s.
  - The 4-bit sub-block contains no more than three 1s or three 0s.
- Any character with other than the above properties is considered invalid and a receiver consider this an error.
- An 8-bit value is submitted to the 8b/10b encoder along with a signal indicating whether the byte is a Data (D) or Control (K) byte. The encoder outputs the equivalent 10-bit character along with a current running disparity (CRD) that represents the sum of 1s and 0s for this transmission link since link initialization. See "Disparity" on page 413 for more information.
- The specification defines Control bytes that encode into two Control characters: (see "Control value Encoding" on page 420).

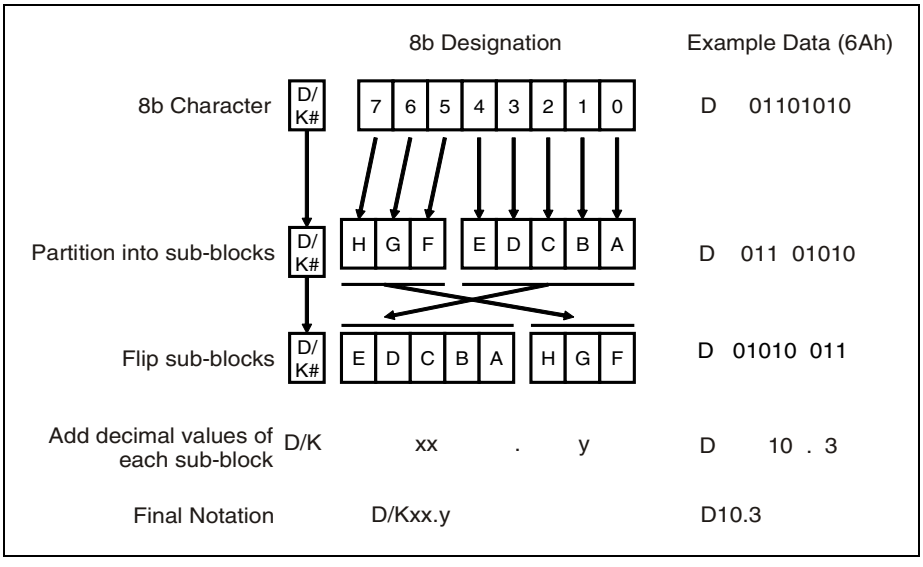
Preparing 8-bit/10-bit Notation

8b/10b conversion lookup tables refer to all 10-bit characters using a special notation (represented by Dxx.y for Data characters and Kxx.y. for Control bytes). Figure 2 on page 413 illustrates the notation equivalent for any 8-bit D or K value. The steps below covert the 8-bit (hex) value to 10-bit (Dxx.y) value.

In Figure 2 on page 413, the example Data byte value is 6Ah.

1. The bits in the data byte are identified by the capitalized alpha designators A through H
2. The byte is partitioned into two sub-blocks: one 3-bits wide and the other 5-bits wide
3. The two sub-blocks are flipped
4. The written form of the 10 bit value begins with either Dxx.y (Data) or Kxx.y (Control)
5. xx = the decimal value of the 5-bit field
6. y = the decimal value of the 3-bit field
7. The example value is represented as D10.3 in the 8b/10b lookup tables

Figure A-2: Preparing 8-bit value for Encode



Disparity

value disparity refers to the difference between the number of 1s and 0s in a 10-bit character:

- When a character has more 0s than 1s, the character has negative (–) disparity (e.g., 0101000101b).
- When a character has more 1s than 0s, the character has positive (+) disparity (e.g., 1001101110b).
- When a character has an equal number of 1s and 0s, the character has neutral disparity (e.g., 0110100101b).
- Each 10-bit character contains one of the following numbers of ones and zeros (not necessarily contiguous):
  - Four 0s and six 1s (+ disparity).
  - Six 0s and four 1s (– disparity).
  - Five 0s and five 1s (neutral disparity).

There are two categories of 8-bit values:

- Those that encode into 10-bit characters with + or – disparity.
- Those that encode into 10-bit characters with neutral disparity.

Appendix B
ATA & SATA Commands

Essential Background

The command set for ATA disk drives has been evolving since the early 1980s. When the Serial ATA interface was introduced, a primary goal was maintaining backward compatibility with software written for parallel ATA applications. This meant that the same commands could be issued to either a parallel or serial ATA disk drive and would produce the same results. It was agreed by all parties involved that there should be only one official definition of the command set.

The standards committee for the parallel ATA interface, the T13 Standards Committee of Accredited Standards Committee INCITS, maintains the ATA Command Set document. See References, below, for details.

There are presently four SATA-only commands and three ATA commands that have been modified to include SATA-only information. These new commands and modifications are documented in the SATA specification. They are briefly discussed below in the section on SATA-only Commands (See page 445).

References

ATA/ATAPI-7 Vol. 1 – Register Delivered Command Set, Logical Register Set. Dated 21 April 2004.

The approved final draft of this document can be downloaded from the standards committee website, www.t13.org.

AT Attachment-8 ATA/ATAPI Command Set (ATA-8 ACS)

This is a working document, not yet approved. The most recent version can be downloaded from www.t13.org.

Serial ATA 2.6 is the latest approved version of the SATA specification.

The specification is developed and maintained by the Serial ATA International Organization. Member companies can download the spec from www.sata-io.org. Non-member companies must pay a fee (currently $25) in order to download a copy.

General Characteristics of ATA Commands

All ATA commands have several characteristics in common.

First, all commands are initiated by writing to a set of 1-byte registers in the device. This register set is formally called the Command Block; it is also often informally called the Task File.

Register 7 is used for the command code, which defines the command to be executed. Registers 1 through 6 are used for command parameters. Host software must write any required command parameters to the device before writing the command code because writing to register 7, the Command Register, signals the device to begin command execution. (See note below on shared register addresses.)

Some register names have changed between early and recent versions of the ATA standard. The two name sets are illustrated in Figure B-1 and Figure B-2 on page 425. In normal usage the early and recent names are used interchangeably.

Figure B-1: Legacy Register Names

Cmd Reg	Reads		Writes		Notes
Address	7	0	7	0	
01F0	Data				16-bit accesses
01F1	Error		Feature		8-bit access only
01F2	Sector Count				8-bit access only
01F3	Sector Number				8-bit access only
01F4	Cylinder Low				8-bit access only
01F5	Cylinder High				8-bit access only
01F6	Device	Head	Device	Head	8-bit access only
01F7	Status		Command		8-bit access only
Ctrl Reg					
03F6	Alternate Status		Device Control		8-bit access only

Figure B-2: LBA Address (28-Bits)

Cmd Reg	Reads		Writes		Notes
Address	7	0	7	0	
01F0	Data				16-bit accesses
01F1	Error		Feature		8-bit access only
01F2	Sector Count				8-bit access only
01F3	LBA Low (7:0)				8-bit access only
01F4	LBA Middle (15:8)				8-bit access only
01F5	LBA High (23:16)				8-bit access only
01F6	Device	LBA (27:24)	Device	LBA (27:24)	8-bit access only
01F7	Status		Command		8-bit access only
Ctrl Reg					
03F6	Alternate Status		Device Control		8-bit access only

Command Structure

Each command in the ATA standard is defined in terms of:

- COMMAND CODE – See the table of command codes in Annex B of ATA/ATAPI-7.
- FEATURE SET – See the section entitled "Feature Sets" on page 432 for a description of the ATA feature set.
- PROTOCOL – the type of transport activity that will be used to execute this command. See Table B-1 for a description of different ATA transport protocols.
- INPUTS – the mandatory and optional register contents supplied by the host system when the command is initiated. An example of the Inputs register map for one ATA command is shown in Figure B-3. Descriptive text accompanies the diagram in the ATA standard.

Figure B-3: Inputs – READ DMA command

Register	7	6	5	4	3	2	1	0
Features	na							
Sector Count	Sector count							
LBA Low	LBA (7:0)							
LBA Mid	LBA (15:8)							
LBA High	LBA (23:16)							
Device	obs	LBA	obs	DEV	LBA (27:24)			
Command	C8h							

Normal Outputs. The register contents after the command completes successfully. Register 7, the Status Register, is usually the only register of interest. (See note below on shared register addresses.) An example of the Normal Outputs register map for one ATA command is shown in Figure B-4 on page 427. Descriptive text accompanies the diagram in the ATA standard.

Figure B-4: Normal Outputs – READ DMA command

Register	7	6	5	4	3	2	1	0
Error	na							
Sector Count	na							
LBA Low	na							
LBA Mid	na							
LBA High	na							
Device	obs	na	obs	DEV	na	na	na	na
Status	BSY	DRDY	DF	na	DRQ	na	na	ERR

Error Outputs. the register contents after the command completes with an error. The Status Register and the Error Register are the most important but other registers may contain useful information. An example of the Error Outputs register map for one ATA command is shown in Figure B-5. Descriptive text accompanies the diagram in the ATA standard.

Figure B-5: Error Outputs – READ DMA command

Register	7	6	5	4	3	2	1	0
Error	ICRC	UNC	MC	IDNF	MCR	ABRT	NM	obs
Sector Count	na							
LBA Low	LBA (7:0)							
LBA Mid	LBA (15:8)							
LBA High	LBA (23:16)							
Device	obs	na	obs	DEV	LBA (27:24)			
Status	BSY	DRDY	DF	na	DRQ	na	na	ERR

Appendix C
Commands by Code

ATA Commands By Code

Code	Protocol	Command	No Packet Cmd	Packet Cmd
00h	ND	NOP	O	M
03h	ND	CFA Request Extended Error	O	N
08h	DR	Device Reset	N	M
25h	DMA	Read DMA Ext	O	N
26h	DMAQ	Read DMA Queued Ext	O	N
27h	ND	Read Native Max Address Ext	O	N
29h	PIO-IN	Read Multiple Ext	O	N
2Bh	PIO-IN	Read Stream Ext	O	N
2Fh	PIO-IN	Read Log Ext	O	O
30h	PIO-OUT	Write Sector(s)	M	N
34h	PIO-OUT	Write Sector(s) Ext	O	N
35h	DMA	Write DMA Ext	O	N
36h	DMAQ	Write DMA Queued Ext	O	N
37h	ND	Set Max Address Ext	O	N
38h	PIO-OUT	CFA Write Sectors Without Erase	O	N
38h	PIO-OUT	Write Stream Ext	O	N
39h	PIO-OUT	Write Multiple Ext	O	N
3Ah	DMA	Write Stream DMA Ext	O	N
3Dh	DMA	Write DMA FUA Ext	O	N
3Eh	DMAQ	Write DMA Queued FUA Ext	O	N
3Fh	PIO-OUT	Write Log Ext	O	O
40h	ND	Read Verify Sector(s)	M	N
42h	ND	Read Verify Sector(s) Ext	O	N
51h	ND	Configure Stream	O	O
87h	PIO-IN	CFA Translate Sector	O	N
90h	DD	Execute Device Diagnostics	M	N
92h	PIO-OUT	Download Microcode	O	N
A0h	P	Packet	N	M
A1h	PIO-IN	Identify Packet Device	N	M
A2h	PIO/DMAQ	Service	O	O
B0h	ND	Smart Disable Operations	O	N
B0h	ND	Smart Enable Operations	O	N
B0h	ND	Smart Enable/Disable Autosave	O	N

Code	Protocol	Command	No Packet Cmd	Packet Cmd
B0h	ND	Smart Execute Off-Line Immediate	O	N
B0h	PIO-IN	Smart Read Data	O	N
B0h	PIO-IN	Smart Read Log	O	N
B0h	ND	Smart Return Status	O	N
B0h	PIO-OUT	Smart Write Log	O	N
B1h	ND	Device Configuration Freeze Lock	O	O
B1h	PIO-IN	Device Configuration Identify	O	O
B1h	ND	Device Configuration Restore	O	O
B1h	PIO-OUT	Device Configuration Set	O	O
B1h	DMA	Read Stream DMA Ext	O	N
C0h	ND	CFA Erase Sectors	F	N
C4h	PIO-IN	Read Multiple	M	N
C5h	PIO-OUT	Write Multiple	M	N
C6h	ND	Set Multiple Mode	M	N
C7h	DMAQ	Read DMA Queued	O	N
C8h	DMA	Read DMA	M	N
CAh	DMA	Write DMA	M	N
CCh	DMAQ	Write DMA Queued	O	N
CDh	PIO-OUT	CFA Write Multiple Without Erase	O	N
CEh	PIO-OUT	Write Multiple FUA Ext	O	N
D1h	ND	Check Media Card Type	O	N
DAh	ND	Get Media Status	O	O
DEh	ND	Media Lock	O	N
DEh	PIO-IN	Read Sector(s) Ext	O	N
DFh	ND	Media Unlock	O	N
E0h	ND	Standby Immediate	M	M
E1h	ND	Idle Immediate	M	M
E2h	ND	Standby	M	O
E3h	ND	Idle	M	O
E4h	PIO-IN	Read Buffer	O	N
E5h	ND	Check Power Mode	M	M
E6h	ND	Sleep	M	M
E7h	ND	Flush Cache	M	O
E8h	PIO-OUT	Write Buffer	O	N
EAh	ND	Flush Cache Ext	O	N
ECh	PIO-IN	Identify Device	M	M
EDh	ND	Media Eject	O	N
EDh	PIO-IN	Read Sector(s)	M	M
EFh	ND	Set Features	M	M
F1h	PIO-OUT	Security Set Password	O	O
F2h	PIO-OUT	Security Unlock	O	O
F3h	ND	Security Erase Prepare	O	O
F4h	PIO-OUT	Security Erase Unit	O	O
F5h	ND	Security Freeze Lock	O	O
F6h	PIO-OUT	Security Disable Password	O	O
F8h	ND	Read Native Max Address	O	O
F9h	ND	Set Max	O	O

Appendix D
Commands By Type

ATA Commands By Type

Protocol Type	Command	Code	No Packet Cmd	Packet Cmd
DD	Execute Device Diagnostics	90h	M	N
DMA	Read DMA Ext	25h	O	N
DMA	Write DMA Ext	35h	O	N
DMA	Write Stream DMA Ext	3Ah	O	N
DMA	Write DMA FUA Ext	3Dh	O	N
DMA	Read Stream DMA Ext	B1h	O	N
DMA	Read DMA	C8h	M	N
DMA	Write DMA	CAh	M	N
DMAQ	Read DMA Queued Ext	26h	O	N
DMAQ	Write DMA Queued Ext	36h	O	N
DMAQ	Write DMA Queued FUA Ext	3Eh	O	N
DMAQ	Read DMA Queued	C7h	O	N
DMAQ	Write DMA Queued	CCh	O	N
DR	Device Reset	08h	N	M
ND	NOP	00h	O	M
ND	CFA Request Extended Error	03h	O	N
ND	Read Native Max Address Ext	27h	O	N
ND	Set Max Address Ext	37h	O	N
ND	Read Verify Sector(s)	40h	M	N
ND	Read Verify Sector(s) Ext	42h	O	N
ND	Configure Stream	51h	O	O
ND	Smart Disable Operations	B0h	O	N
ND	Smart Enable Operations	B0h	O	N
ND	Smart Enable/Disable Autosave	B0h	O	N
ND	Smart Execute Off-Line Immediate	B0h	O	N
ND	Smart Return Status	B0h	O	N
ND	Device Configuration Freeze Lock	B1h	O	O
ND	Device Configuration Restore	B1h	O	O
ND	CFA Erase Sectors	C0h	F	N
ND	Set Multiple Mode	C6h	M	N
ND	Check Media Card Type	D1h	O	N
ND	Get Media Status	DAh	O	O
ND	Media Lock	DEh	O	N

Protocol Type	Command	Code	No Packet Cmd	Packet Cmd
ND	Media Unlock	DFh	O	N
ND	Standby Immediate	E0h	M	M
ND	Idle Immediate	E1h	M	M
ND	Standby	E2h	M	O
ND	Idle	E3h	M	O
ND	Check Power Mode	E5h	M	M
ND	Sleep	E6h	M	M
ND	Flush Cache	E7h	M	O
ND	Flush Cache Ext	EAh	O	N
ND	Media Eject	EDh	O	N
ND	Set Features	EFh	M	M
ND	Security Erase Prepare	F3h	O	O
ND	Security Freeze Lock	F5h	O	O
ND	Read Native Max Address	F8h	O	O
ND	Set Max	F9h	O	O
P	Packet	A0h	N	M
PIO/DMAQ	Service	A2h	O	O
PIO-IN	Read Multiple Ext	29h	O	N
PIO-IN	Read Stream Ext	2Bh	O	N
PIO-IN	Read Log Ext	2Fh	O	O
PIO-IN	CFA Translate Sector	87h	O	N
PIO-IN	Identify Packet Device	A1h	N	M
PIO-IN	Smart Read Data	B0h	O	N
PIO-IN	Smart Read Log	B0h	O	N
PIO-IN	Device Configuration Identify	B1h	O	O
PIO-IN	Read Multiple	C4h	M	N
PIO-IN	Read Sector(s) Ext	DEh	O	N
PIO-IN	Read Buffer	E4h	O	N
PIO-IN	Identify Device	ECh	M	M
PIO-IN	Read Sector(s)	EDh	M	M
PIO-OUT	Write Sector(s)	30h	M	N
PIO-OUT	Write Sector(s) Ext	34h	O	N
PIO-OUT	CFA Write Sectors Without Erase	38h	O	N
PIO-OUT	Write Stream Ext	38h	O	N
PIO-OUT	Write Multiple Ext	39h	O	N
PIO-OUT	Write Log Ext	3Fh	O	O
PIO-OUT	Download Microcode	92h	O	N
PIO-OUT	Smart Write Log	B0h	O	N
PIO-OUT	Device Configuration Set	B1h	O	O
PIO-OUT	Write Multiple	C5h	M	N
PIO-OUT	CFA Write Multiple Without Erase	CDh	O	N
PIO-OUT	Write Multiple FUA Ext	CEh	O	N
PIO-OUT	Write Buffer	E8h	O	N
PIO-OUT	Security Set Password	F1h	O	O
PIO-OUT	Security Unlock	F2h	O	O
PIO-OUT	Security Erase Unit	F4h	O	O
PIO-OUT	Security Disable Password	F6h	O	O

Appendix E

Commands By Name

ATA Commands By Name

Command	Code	Protocol	No Packet Cmd	Packet Cmd
CFA Erase Sectors	C0h	ND	F	N
CFA Request Extended Error	03h	ND	O	N
CFA Translate Sector	87h	PIO-IN	O	N
CFA Write Multiple Without Erase	CDh	PIO-OUT	O	N
CFA Write Sectors Without Erase	38h	PIO-OUT	O	N
Check Media Card Type	D1h	ND	O	N
Check Power Mode	E5h	ND	M	M
Configure Stream	51h	ND	O	O
Device Configuration Freeze Lock	B1h	ND	O	O
Device Configuration Identify	B1h	PIO-IN	O	O
Device Configuration Restore	B1h	ND	O	O
Device Configuration Set	B1h	PIO-OUT	O	O
Device Reset	08h	DR	N	M
Download Microcode	92h	PIO-OUT	O	N
Execute Device Diagnostics	90h	DD	M	N
Flush Cache	E7h	ND	M	O
Flush Cache Ext	EAh	ND	O	N
Get Media Status	DAh	ND	O	O
Identify Device	ECh	PIO-IN	M	M
Identify Packet Device	A1h	PIO-IN	N	M
Idle	E3h	ND	M	O
Idle Immediate	E1h	ND	M	M
Media Eject	EDh	ND	O	N
Media Lock	DEh	ND	O	N
Media Unlock	DFh	ND	O	N
NOP	00h	ND	O	M
Packet	A0h	P	N	M
Read Buffer	E4h	PIO-IN	O	N
Read DMA	C8h	DMA	M	N
Read DMA Ext	25h	DMA	O	N
Read DMA Queued	C7h	DMAQ	O	N
Read DMA Queued Ext	26h	DMAQ	O	N
Read Log Ext	2Fh	PIO-IN	O	O

Command	Code	Protocol	No Packet Cmd	Packet Cmd
Read Multiple	C4h	PIO-IN	M	N
Read Multiple Ext	29h	PIO-IN	O	N
Read Native Max Address	F8h	ND	O	O
Read Native Max Address Ext	27h	ND	O	N
Read Sector(s)	EDh	PIO-IN	M	M
Read Sector(s) Ext	DEh	PIO-IN	O	N
Read Stream DMA Ext	B1h	DMA	O	N
Read Stream Ext	2Bh	PIO-IN	O	N
Read Verify Sector(s)	40h	ND	M	N
Read Verify Sector(s) Ext	42h	ND	O	N
Security Disable Password	F6h	PIO-OUT	O	O
Security Erase Prepare	F3h	ND	O	O
Security Erase Unit	F4h	PIO-OUT	O	O
Security Freeze Lock	F5h	ND	O	O
Security Set Password	F1h	PIO-OUT	O	O
Security Unlock	F2h	PIO-OUT	O	O
Service	A2h	PIO/DMAQ	O	O
Set Features	EFh	ND	M	M
Set Max	F9h	ND	O	O
Set Max Address Ext	37h	ND	O	N
Set Multiple Mode	C6h	ND	M	N
Sleep	E6h	ND	M	M
Smart Disable Operations	B0h	ND	O	N
Smart Enable Operations	B0h	ND	O	N
Smart Enable/Disable Autosave	B0h	ND	O	N
Smart Execute Off-Line Immediate	B0h	ND	O	N
Smart Read Data	B0h	PIO-IN	O	N
Smart Read Log	B0h	PIO-IN	O	N
Smart Return Status	B0h	ND	O	N
Smart Write Log	B0h	PIO-OUT	O	N
Standby	E2h	ND	M	O
Standby Immediate	E0h	ND	M	M
Write Buffer	E8h	PIO-OUT	O	N
Write DMA	CAh	DMA	M	N
Write DMA Ext	35h	DMA	O	N
Write DMA FUA Ext	3Dh	DMA	O	N
Write DMA Queued	CCh	DMAQ	O	N
Write DMA Queued Ext	36h	DMAQ	O	N
Write DMA Queued FUA Ext	3Eh	DMAQ	O	N
Write Log Ext	3Fh	PIO-OUT	O	O
Write Multiple	C5h	PIO-OUT	M	N
Write Multiple Ext	39h	PIO-OUT	O	N
Write Multiple FUA Ext	CEh	PIO-OUT	O	N
Write Sector(s)	30h	PIO-OUT	M	N
Write Sector(s) Ext	34h	PIO-OUT	O	N
Write Stream DMA Ext	3Ah	DMA	O	N
Write Stream Ext	38h	PIO-OUT	O	N

Glossary

Term	Description
8b/10b	Shorthand that describes the encoding scheme used by SATA and many other serial transports. Encoding each 8-bit byte into a 10-bit value for transmission introduces some overhead but provides a number of benefits.
AHCI	Advance Host Controller Interface. This controller is used in many PC implementations as the interface between the PCI bus and SATA link.
ALIGN	A primitive used during initialization to help receivers detect aligned 10-bit symbol boundaries. Also used by receivers during normal link operation to manage clock compensation via an elastic buffer.
ATAPI	The ATA Packet Interface protocol that delivers SCSI command to an ATAPI device. CD-ROM, Tape, and DVD drives typically use this protocol. This support is extended to SATA.
Back Bus	A SATA link uses a half duplex signaling scheme using transmit and receive differential pairs. The device transmitting a Frame receives feedback from the receiving node on its back bus (i.e., receive differential pair).
EMI	Electro-Magnetic Interference.
EOF	End Of Frame — A primitive sent at the end of each Frame.
eSATA	External SATA — The interconnect used for external desktop applications.
FIS	Frame Information Structure. A packet of information used for basic communications between the HBA and drive.
Frame	A packet of information transmitted across the link that includes SOF, FIS, CRC, and EOF.

Term	Description
Gen 1	Generation One of the SATA interface based initially on the 1.0a version of the SATA specification. Also, commonly refers to the transmission rate of 1.5Gb/s.
Gen1i	Generation One Internal Connections — refers to the internal SATA interconnect and related electrical specifications for Gen 1 speeds (1.5 Gbps). The maximum cable length can be up to 1 meter.
Gen1m	Generation One Medium Difficulty Environments — refers to External Desktop connections with maximum cable lengths of 2 meters and short backplane applications and related electrical specifications operating at Gen 1 speeds.
Gen1x	Generation One Extreme Environments — refers to long backplane implementations and data center Applications and the related electrical specifications for supporting these environments. The maximum cable length can be greater than two meters.
Gen 2	Generation Two of the SATA interface commonly referred to as SATA II. Also, refers to the SATA II transmission rate of 3.0Gb/s.
Gen2i	Generation Two Internal Connections — refers to the internal SATA interconnect and related electrical specifications at Gen 2 speeds (3.0 Gbps). The maximum cable length can be up to 1 meter.
Gen2m	Generation Two Medium Difficulty Environments — refers to External Desktop connections (maximum cable lengths of 2 meters) and short backplane applications and the related electrical specifications for supporting these environments.
Gen2x	Generation Two Extreme Environments — refers to long backplane implementations and data center Applications and the related electrical specifications for supporting these environments. The maximum cable length can be greater than two meters.
HBA	Host Bus Adapter. The device that acts as a bridge between the system bus (host) and the storage bus.

Term	Description
ISI	Inter-symbol interference - results when the pattern of the previous symbol's bits interferes with proper reception of the current symbol.
JBOD	Just a Bunch Of Disks — a very simple collection of disk drives without a specific organization. See also RAID.
NAS	Network Attached Storage — a server dedicated to file sharing. It allows storage to be added to an existing server network without the need to shut down the network.
NCQ	Native Command Queuing — NCQ permits drives to queue up to 32 commands and execute them in the order that will result in the best overall performance.
Near-line storage	Defined as data storage on removable media, the purpose of near-line storage is to provide inexpensive, reliable, and unlimited backup. The access time for recovery depends on where the needed media resides — it can be slow because the media could be off-line when the data is needed. The combination of near-line and off-line storage accounts for 90% of a typical system's storage allocation. (See also *On-line* storage and *Off-line* storage.)
Off-line storage	Storage of data that must be retained but will rarely be accessed. It is typically stored on slow but reliable tape. (See also *On-line* storage and *Off-line* storage.)
OOB	Out Of Band — defined as communication that is considered outside the normal flow of information because it takes place over the link during initialization.
Overlap	Overlap provides a method for drives (usually slow CD-ROM drives) to relinquish ownership of the shared ATA interface, so that faster hard drives can receive command while the slower device processes the request is has received.

MindShare Press

MindShare Press is constantly looking for high-quality technical material to publish. If you have the desire, experience and ability to author a book on an interesting technical topic, contact MindShare Press to learn about our publishing services and the numerous benefits to having the MindShare name on your book.

Upcoming Titles

- PCI Express 2.0 System Architecture
- Certified Wireless USB System Architecture
- USB Embedded System Architecture
- HyperTransport 3.0 System Architecture
- PC Architecture
- Core and Core2 Processor Architecture
- AMD64 Software Architecture
- Opteron Processor Architecture

MindShare Press - publishing@mindshare.com

SATA Storage Technology Serial ATA

"This *SATA Storage Technology Serial ATA* book from MindShare is the ultimate tutorial on the SATA specification and its implementation. Any SATA based product developer will find the clear and concise explanations of the various features found in this protocol to be invaluable. Particular emphasis is placed on the sections of the spec that enable SATA to be such an important interface for enterprise storage systems."

– Mark Adams, Product Marketing Manager of the LeCroy Corporation

The SATA (Serial ATA) interface is intended to expand and eventually replace the parallel ATA interface that has been the standard connection for hard drives and CD-ROM/DVD drives in Personal Computers for nearly twenty years. SATA, a high-speed serial version of ATA, is designed to maintain compatibility with software written for the standard ATA implementation, thus easing the transition to SATA. New advances in SATA provide new capabilities such as Native Command Queuing that improve overall drive performance and greater reliability. In addition, the latest version of SATA (called SATA II) adds new features such as higher transfer rates, Port Selectors, Port Multipliers, and enclosure services, making it easier to implement low cost, high capacity storage subsystems for server implementations. This book provides essential information for Systems Engineers, System Validation Engineers, Hardware Designers, and Software Developers.

SATA Storage Technology Serial ATA provides ATA background information necessary to understand the Serial ATA implementation. This book provides detailed information regarding the serial protocol, commands, and electrical interface. A tutorial approach is taken to facilitate thorough understanding, using numerous illustrations, tables and examples. The text is also designed to provide easy to access valuable reference information.

Don Anderson has over 30 years of experience in the technical electronics and computer industry. He has authored fourteen books covering various aspects of computer hardware and system design. Topics include system architectures, parallel bus technologies, serial bus technologies, and processor architectures. Don has trained thousands of engineers in the US and around the world. Before joining MindShare, Don worked for Compaq, Schlumberger, Geosource and Hewlett Packard.

MindShare, Inc. was founded in 1988 to bring computer architecture-related training to the hardware and software design communities. Since then, we have expanded our course offerings to a broad range of technical topics which include SAS™, SATA™, PCI Express™ 2.0, HyperTransport™ 3.0, DRAM, Certified Wireless USB, System Architectures as well as Intel Core and Core 2 and AMD Opteron™ Processor Architectures and Virtualization Technology. MindShare instructors travel the world delivering customized training courses to the very companies that lead the high-tech industry, including IBM, HP, PLX, Sun, AMD, and Texas Instruments. Our instructors have tremendous computer engineering experience. Our consistent research and training efforts ensure that instructors and authors are always up-to-date on the latest technological trends.

Contact MindShare at **www.mindshare.com**
or 1-800-633-1440 for training on this book subject or other subjects. **$59.99 USA**

ISBN-13: 978-0-9770878-1-5
ISBN-10: 0-9770878-1-6
55999
9 780977 087815