Serial ATA II Native Command Queuing Overview

Application Note

April 2003

Order Number: 252664-002

Contents

Figures

Tables

Revision History

Date	Revision	Description
04/15/2003	002	Revised Figures 8—11. Added material to Section 2.
02/19/2003	001	Initial document

1.0 Introduction

As Serial ATA gains momentum in 2003, one of the most anticipated features is the Serial ATA II Native Command Queuing. This feature adds data-handling intelligence that will deliver the performance needed for the next generation of entry-level server, networked storage and high-end PC markets. The focus of this whitepaper is to provide a general understanding of Serial ATA II Native Command Queuing. The *Serial ATA II: Extensions to Serial ATA 1.0 Specification* contains the definition of Serial ATA II Native Command Queuing and should be consulted for detailed information regarding the queuing protocol.

Table 1 provides definitions for some of the terms used throughout this whitepaper.

1.1 Terms and Definitions

Table 1. Acronym Definitions

Prefix	Description
Device	As used in this paper, “device” refers to a Serial ATA hard disk drive.
DMA	Direct Memory Access (DMA) is a means of data transfer between the device and host memory without host processor intervention.
DPA	Direct Port Access (DPA) allows each of the Serial ATA ports to be independently accessed while allowing simultaneous transfer of data across all ports.
FIS	Frame Information Structure (FIS) is the user payload of a frame (not including SOF, CRC and EOF delimiters).
First Party DMA (FPDMA)	The drive uses the First Party DMA mechanism to cause the host controller to select the appropriate destination or source host memory region for the data transfer portion of a queued command. The drive uses the DMA Setup FIS to communicate the tag of the queued command that the drive wants to transfer data for next. In response to the DMA Setup FIS, the host controller will initialize the DMA engine with the appropriate PRD table for the queued command with the tag indicated.
Frame	A frame is the unit of information exchanged between the host and the device. It consists of a start of frame (SOF) primitive, a Frame Information Structure (FIS), a CRC calculated over the contents of the FIS and lastly an end of frame (EOF) primitive.
HBA	A Host Bus Adapter (HBA) is a device that connects to the host system’s expansion bus to provide connectivity for devices. Host Bus Adapters are also often referred to as controller cards or host controllers.
PATA	Parallel ATA – AT Attachment (PATA) defines the physical, electrical, transport and command protocols for the internal attachment of storage devices.
PRD	Physical Region Descriptor (PRD) is a data structure used by DMA engines to describe memory regions for transferring data to/from the device. It is an entry in a scatter/gather list.
PRD Table	Physical Region Descriptor (PRD) Table – is also referred to as a scatter/gather list.
SATA	Serial ATA (SATA) defines the high-speed serialized ATA data link interface.
Scatter/Gather List	Scatter/gather list is a table of descriptors, each of which gives the location and length of one segment in the overall read or write request. This list is used to do DMA data transfers of data that is written to noncontiguous areas of memory. The last one in the list is indicated with an end of table bit set.

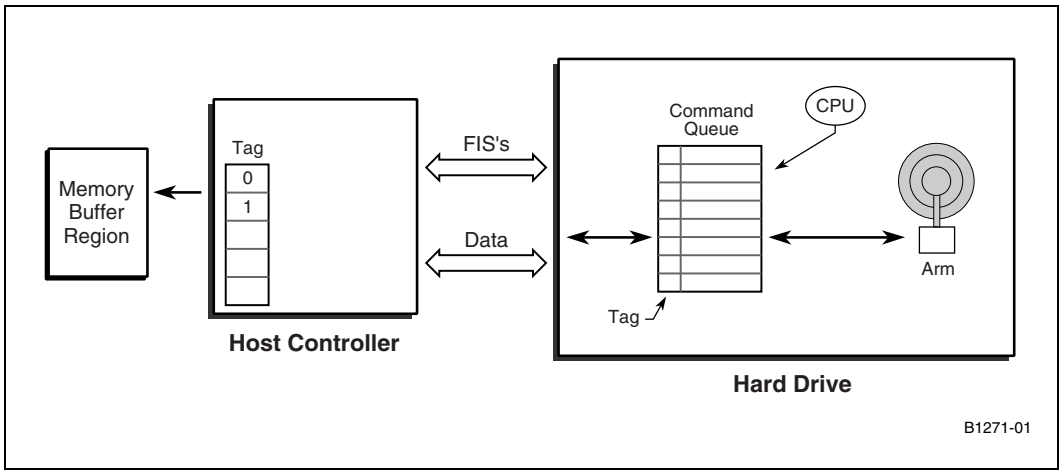
2.0 Command Queuing Overview

Command queuing enables a hard drive to accept multiple commands from the host controller and rearrange the completion order of those commands to maximize throughput. The major portion of the drive's command service time is seek and rotational delay for the drive head to land on the appropriate data to transfer. The drive can use rotational optimizations to select the next command to complete so that the major components of the service time, seek and rotational delay, are minimized.

Rotational optimizations can be achieved when the drive selects the next command based on the fact that it is closest rotationally to the head's current position. The seek optimizations can be achieved when the drive selects the next command based on their cylinder number, compared to the cylinder where the heads currently are located. A major advantage to command queuing is that the command issue and completion overhead may be overlapped with the drive seek and rotational delay for a different command's data transfer. For example, while a new command is being issued to the drive, the drive may be seeking to locate the appropriate track on disk for data for a different command. In essence, the latency for issuing the new command is saved since it was overlapped with the seek for another command.

Figure 1 shows the host controller communicating to a hard drive. Each queued command has an associated tag value. The host in both the data transfer phase and completion phase of a queued command uses the tag. To transfer data for a particular command, the disk drive communicates to the host the tag of that command. The host then sets up a DMA operation that points to the appropriate host memory region for that command. To complete a command, the disk drive sends the associated tag of the command to be completed. The use of tags allows the host to protect memory regions from being errantly accessed by the device. The drive also sends status information to update the host controller with the status of pending commands and associated errors.

Figure 1. Command Queuing Block Diagram



2.1 Parallel ATA Queuing Overview

Parallel ATA Queuing is a queuing protocol that was added to the ATA specification in 1997. Today, only one drive vendor is shipping drives that support this queuing protocol because of the high overhead. The high overhead of Parallel ATA Queuing causes it to significantly under-perform non-queued commands when the queuing workload is light (especially when the protocol is not fully hardware accelerated). The Parallel ATA Queuing protocol consists of several commands including Read DMA Queued (Ext), Write DMA Queued (Ext), and Service.

The significant overhead of Parallel ATA Queuing is due to a split command issue and data transfer phase. Before the data transfer for a Parallel ATA queued command, the host must issue the Service command to determine which command the data transfer is for, and then the host will set up the DMA engine for the appropriate command. In order to alert the host to issue the Service command, an additional interrupt is taken per IO. There is indeterminate latency for the host software to take control and issue the Service command if the protocol is not fully hardware accelerated. This latency is the major cause for the poor performance of Parallel ATA Queuing.

2.2 Serial ATA II Native Command Queuing Overview

Serial ATA II Native Command Queuing is a queuing protocol that was designed to make efficient use of the Serial ATA protocol and streamline the data transfer portion of queued commands. The Serial ATA II Native Command Queuing protocol does not require any changes to the underlying Serial ATA protocol; the protocol only uses FISes and primitives that are part of the Serial ATA 1.0 specification. There are two commands, Read FPDMA Queued and Write FPDMA Queued, which utilize this queuing protocol.

In order to use Serial ATA II Native Command Queuing commands over a particular Serial ATA port, three requirements must be met:

1. The host controller[1] must support the DMA Setup FIS and the Set Device Bits FIS in order to enable First Party DMA transfers and efficient command completion.
2. The software driver must support the Read FPDMA Queued and Write FPDMA Queued commands.
3. The drive must support Serial ATA II Native Command Queuing. These drives are expected to be widely available in the second half of 2003.

Note that the queuing protocol may be used on a subset of the total number of Serial ATA ports present on the host controller if some of the attached drives do not support Serial ATA II Native Command Queuing. Host driver software will detect drive support for the Serial ATA II Native Command Queuing protocol and will only issue the FPDMA Queued commands to drives that support them.

This allows drives that support and do not support the queuing protocol to be used interchangeably in the same disk subsystem. Drives that support Serial ATA II Native Command Queuing indicate their support of the protocol in their Identify Device page. Devices support the FPDMA Queued commands in addition to all of the standard ATA/6 commands.

The queuing protocol supports up to 32 outstanding commands at one time. Deeper queuing can be done in the host driver or RAID controller for additional performance benefits.

1. This paper assumes that the host controller supports First Party DMA mechanism is implemented using full hardware automation such as Intel's GD31244 Serial ATA Controller.

2.3 Serial ATA II Native Command Queuing

Serial ATA II Native Command Queuing provides an efficient and streamlined data transfer and status return mechanism. The increased performance and efficiency is achieved through features of the SATA II Native Command Queuing protocol that include race-free status return mechanism, interrupt aggregation, and First Party DMA.

2.3.1 Race-free Status Return Mechanism

Serial ATA II Native Command Queuing has a race-free status return mechanism that allows status to be returned on any command at any time. There is no handshake required with the host for the status return. The drive may issue command completions for multiple commands back-to-back or even at the same time.

Parallel ATA Command Queuing does not have a race-free status return mechanism. The drive cannot return status for another command until the host driver software (or hardware if accelerated) explicitly issues a Service command for the next command to be completed. This adds latency and CPU overhead to the command completion phase of the protocol.

2.3.2 Interrupt Aggregation

Serial ATA II Native Command Queuing has a maximum of one interrupt per command. In actuality, the number of interrupts per command is less than one due to a feature called interrupt aggregation. If the drive completes multiple commands in a short time span, the individual interrupts for each command may be aggregated into one interrupt by the host controller. In this case, the host driver software only sees one interrupt for multiple commands. In a highly queued workload this is a frequent occurrence since host driver software interrupt service latency may be long in comparison to the time between command completions.

Parallel ATA Command Queuing has two interrupts for every command, and there is no means to aggregate completion interrupts. The additional interrupt is taken so that host software can set up the DMA engine for the data transfer.

2.3.3 First Party DMA (FPDMA)

Serial ATA II Native Command Queuing has a mechanism such that the drive can select the DMA context for a subsequent data transfer without host driver software intervention using the host controller. This mechanism is called First Party DMA. The drive selects the DMA context by sending a DMA Setup FIS to the host controller specifying the tag of the command that the data transfer is for. The host controller will load the scatter/gather table pointer for that command (based on the tag value) into the DMA engine. Then the DMA transfer may proceed.

Parallel ATA Command Queuing does not have the First Party DMA mechanism. In this queuing protocol, the drive causes an interrupt when it is ready to transfer data. The host software (or hardware) processes the interrupt and determines that the SERV bit is set in the Status register. This means that the drive is ready to proceed with the data transfer for a particular command. Next the host must issue the Service command to the device to determine the tag of the command the data transfer is for. After the Service command is complete, the host software driver (or hardware) can set up the DMA engine for that command. This process has non-deterministic latency since the host software driver must intervene if the queuing protocol is not fully hardware accelerated.

Note that the significant overhead for the software driver setup of the DMA engine setup for Parallel ATA Command Queuing cannot be masked well by the seek or rotational latency for that transfer. If the drive requested to set up the DMA engine immediately (in order to mask the overhead), then the host could not issue additional commands to the drive until after the data transfer was complete. This would make the queuing scheme ineffective since the host could not issue new commands most of the time.

2.4 Command Queuing Comparison

To summarize, Parallel ATA Command Queuing has significant overhead (especially when there is no hardware acceleration) that impacts its performance appreciably. Serial ATA II Native Command Queuing eliminates the shortcomings of Parallel ATA Command Queuing and achieves a queuing protocol similar in overhead to SCSI queuing.

3.0 Serial ATA II Native Command Queuing Protocol Overview

This section describes the definitions of FIS structures, the SActive register and the commands that are used for SATA II Native Command Queuing. For a complete description of the Serial ATA II Native Command Queuing protocol, refer to *Serial ATA II: Extensions to Serial ATA 1.0 Specification* at http://www.serialata.org/collateral/index.shtml.

3.1 FIS Structures used in SATA II Native Command Queuing Protocol

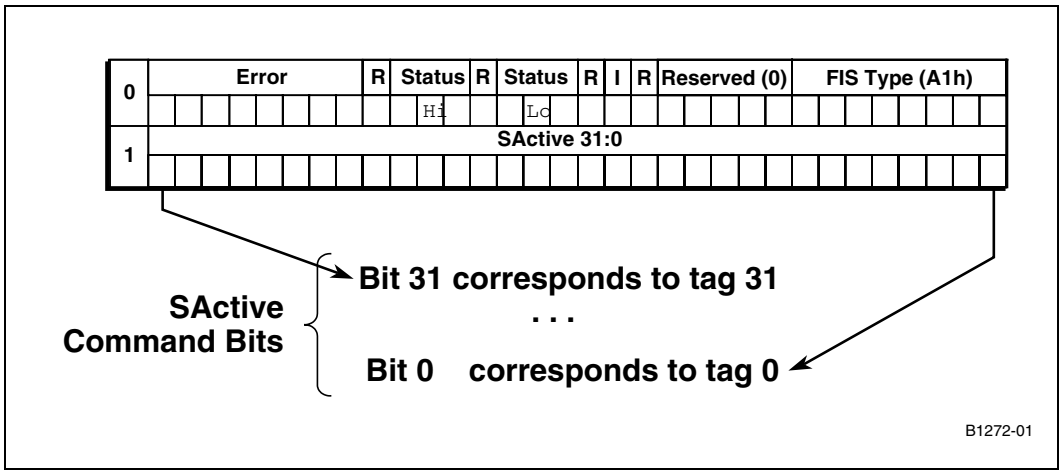
To support Serial ATA II Native Command Queuing the Set Device Bits FIS and DMA Setup FIS have certain fields that are defined to have a particular meaning when queued commands are outstanding. The Set Device Bits FIS is used to return race-free status and allow interrupt aggregation. The DMA Setup FIS is used to provide the First Party DMA mechanism.

3.2 Set Device Bits FIS

The Set Device Bits FIS is sent by the device to the host to update status information and may be sent regardless of the state of the BSY bit in the host Status register. This FIS, shown in Figure 2, is modified to add the SActive bits in order to support the Serial ATA II Native Command Queuing protocol. Each one of these bits corresponds to a tag number of a particular queued command.

To complete a command, the device issues a Set Device Bits FIS with the SActive bit set to 1 for the particular command(s) that the device is completing. The device may complete multiple commands with one Set Device Bits FIS. The device will trigger an interrupt when this FIS is issued to the host so that host driver software may complete the commands.

Figure 2. Set Device Bits FIS



The drive is not required to support a 32 command-deep queue. The number of commands supported in its command queue is indicated in the Identify Device word 75. When a drive supports less than 32 commands, only tags with a value less than the number of commands supported is allowed. For example, if a drive supports a 16 command-deep queue, only tags 0-15 may be used.

3.3 DMA Setup FIS

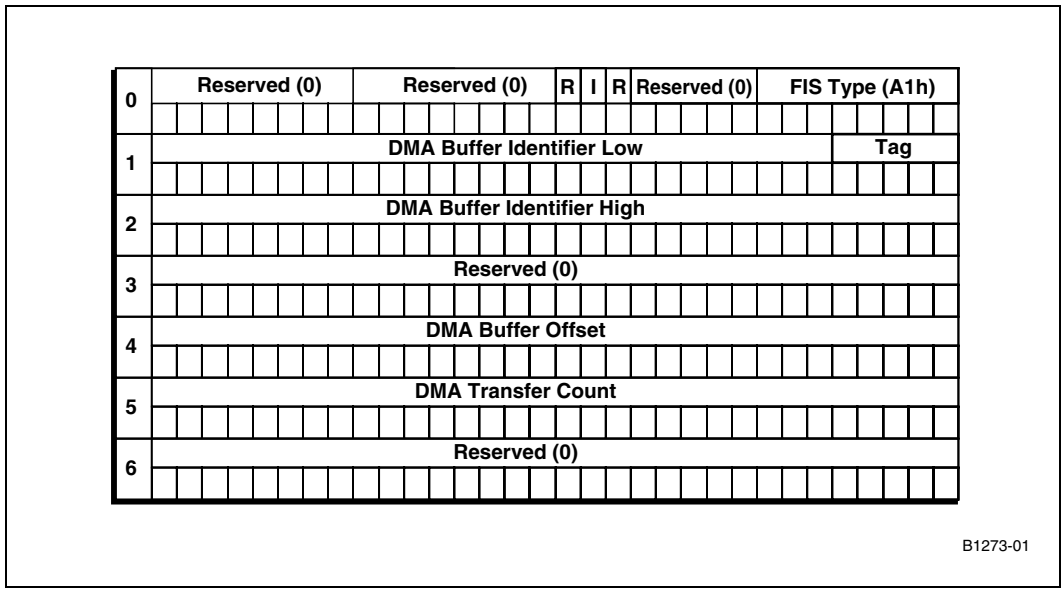
The DMA Setup FIS is sent by the device to the host to provide the First Party DMA mechanism. The device places the tag of the command to perform a data transfer for in the DMA Setup FIS. This FIS is shown in Figure 3. The following list describes the fields in this FIS.

- **TAG** - The **TAG** field identifies the command for which the data is being transferred. The host uses this information to setup the DMA engine. The remaining bits in the DMA Buffer Identifier Low/High fields are zero.
- **A** - The **A or Auto-Activate** bit allows a further streamlining of the protocol. If the A bit is set to 1 it eliminates the need for a DMA Activate FIS immediately following a DMA Setup FIS on a queued write command.
- **D** - The **D** bit identifies the direction of the transfer such that host hardware need not be aware of whether this corresponds to a read or write command.
- **DMA Transfer Count** -The **DMA Transfer Count** field identifies the number of bytes in the transfer to allow for easy hardware automation of this function in the host controller.

An interrupt is not generated by this FIS since it is handled by the hardware.

After receiving this FIS, the host controller will set the DMA engine to point to the scatter/gather list for the command with the corresponding tag.

Figure 3. DMA Setup FIS



3.4 SActive Register

The SActive register is used to track completion status of queued commands. This register is part of the control, status and error superset registers defined in the Serial ATA specification. Figure 4 shows this 32-bit register.

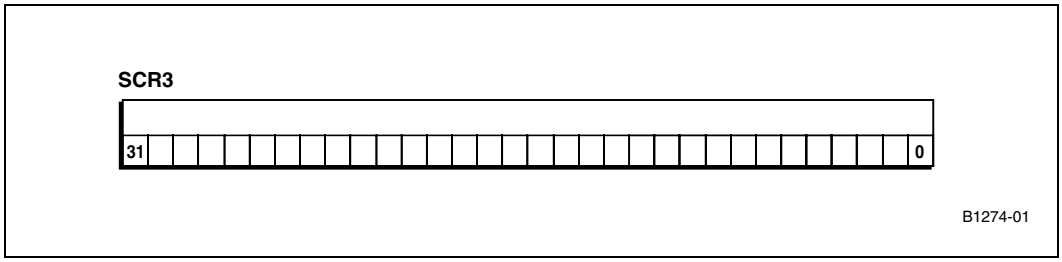
The host sets the SActive register bits. Before host driver software issues a queued command to the device, it sets the bit corresponding to the tag of the queued command it is about to issue. The device clears the SActive register bits. When the device completes a queued command, it clears the bit corresponding to the tag of the queued command in the SActive register. Host driver software queries the SActive register to determine which commands are complete.

The device clears bits in the SActive register using the Set Device Bits FIS. The device sets bits in the SActive field of the Set Device Bits FIS that correspond to bits that it wants to clear in the SActive register. Upon receiving a Set Device Bits FIS from the device, the host controller will clear bits in the SActive register that correspond to bits that are set in the SActive field of the Set Device Bits FIS. This mechanism of the host setting bits in the register and the device clearing bits in the register ensures that no race condition can occur.

Here are a few examples of how the bit field relates to the status of queued commands:

- 1 in bit location 0 signifies that the command with tag 0 is still pending
- 1 in bit location 16 signifies that the command with tag 16 is still pending
- 0 in bit location 16 signifies that the command with tag 16 is complete (if the bit was previously set)

Figure 4. SActive Register



3.5 Command Definitions

This section describes the two commands that use the Serial ATA II Native Command Queuing protocol.

3.5.1 Read FPDMA Queued

The Read FPDMA Queued command is used to perform a queued First Party DMA read. This command supports LBA mode only and uses 48-bit formatting. Notice that the Sector Count register contains the tag number that is assigned by the host driver software. Figure 5 shows the format of this command.

Figure 5. Read FPDMA Queued Command Format

REGISTER	7	6	5	4	3	2	1	0
Features	Sector Count 7:0							
Features (exp)	Sector Count 15:8							
Sector Count	TAG					Reserved		
Sector Count (exp)	Reserved							
Sector Number	LBA 0:7							
Sector Number (exp)	LBA 31:24							
Cylinder Low	LBA 15:8							
Cylinder Low (exp)	LBA 39:32							
Cylinder High	LBA 23:16							
Cylinder High (exp)	LBA 47:40							
Device/Head	FUA	1	Res	0	Reserved			
Command	60h							

B1275-01

The device signals command completion status by issuing a Set Device Bits FIS to the host. The Set Device Bits FIS will trigger an interrupt so that the host driver software can complete the command. host driver software will interrogate the SActive register to determine that the command is complete.

3.5.2 Write FPDMA Queued

The Write FPDMA Queued command is used to perform a queued First Party DMA write command. This command supports LBA mode only and uses 48-bit formatting. Notice that the Sector Count register contains the tag number that is assigned by the host driver software. Figure 6 shows the format of this command.

Figure 6. WriteFPDMA Command Format

REGISTER	7	6	5	4	3	2	1	0
Features	Sector Count 7:0							
Features (exp)	Sector Count 15:8							
Sector Count	TAG					Reserved		
Sector Count (exp)	Reserved							
Sector Number	LBA 0:7							
Sector Number (exp)	LBA 31:24							
Cylinder Low	LBA 15:8							
Cylinder Low (exp)	LBA 39:32							
Cylinder High	LBA 23:16							
Cylinder High (exp)	LBA 47:40							
Device/Head	FUA	1	0	0	Reserved			
Command	61h							

B1276-01

The device signals command completion status by issuing a Set Device Bits FIS to the host. The Set Device Bits FIS will trigger an interrupt so that the host driver software can complete the command. host driver software will interrogate the SActive register to determine that the command is complete.

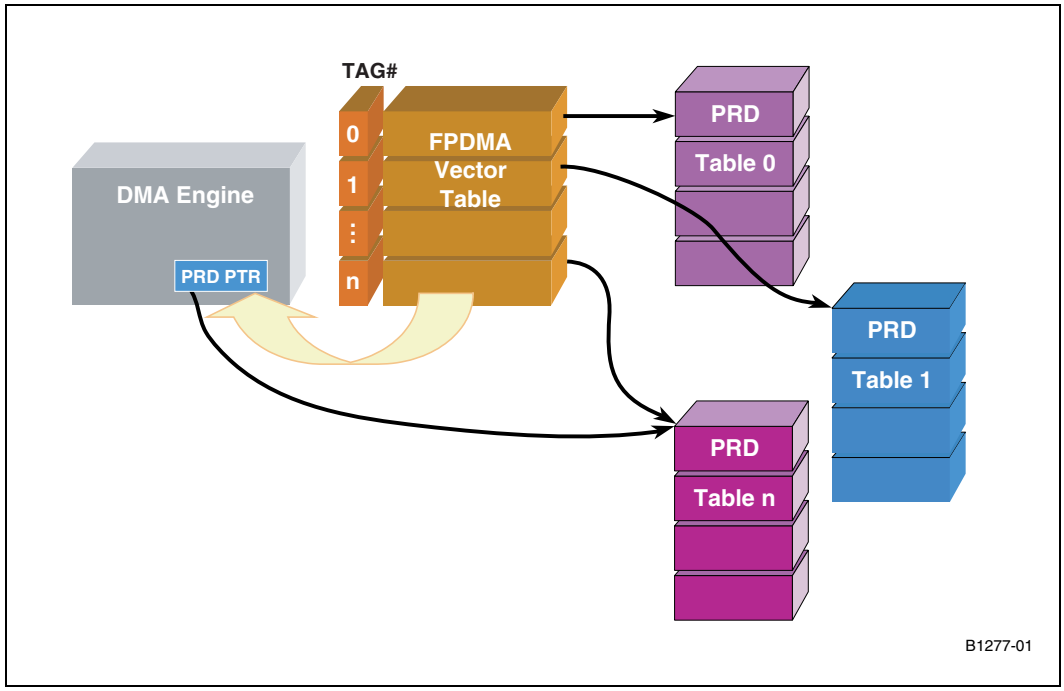
4.0 Serial ATA II Native Command Queuing Implementation

This section outlines the specific hardware support and registers implemented in a host controller in order to provide a high performance Serial ATA II Native Command Queuing implementation.

4.1 First Party DMA Mechanism

The host hardware must provide a mechanism by which the DMA engine can be programmed appropriately when a DMA Setup FIS is received. The block diagram in Figure 7 shows how the First Party DMA mechanism is implemented using full hardware automation.

Figure 7. First Party DMA Mechanism



The FPDMA Vector Table is set up by the host in nonpaged memory. This table has 32 entries with each entry corresponding to a tag number. Each table entry contains the physical address of the scatter/gather list (PRD table) for the command with the associated tag number. The scatter/gather list describes the host memory regions used in the data transfer for a particular queued command.

Host driver software sets the host controller's Queuing Table Base Address register to the physical address of the beginning of the FPDMA Vector Table. host driver software must fill in the PRD table for a particular command and set the FPDMA Vector Table entry appropriately prior to issuing the command to the device.

When the host controller receives a DMA Setup FIS from the device, it uses the TAG field as an index into the FPDMA Vector Table. The host controller sets the DMA engine PRD pointer to the value specified in the FPDMA Vector Table. The host controller sets up the direction of the DMA transfer using the D bit in the DMA Setup FIS.

4.2 Command Completion Mechanism

The Serial ATA II Native Command Queuing command completion mechanism is implemented using the SActive register. The host controller implements the SActive register in accordance with the *Serial ATA II: Extensions to Serial ATA 1.0 Specification*. The SActive register bits are set by the host and cleared by the device using the Set Device Bits FIS.

5.0 Example of Serial ATA II Native Command Queuing

This section provides an example of how a Read FPDMA Queued command would be implemented with SATA II Native Command Queuing.

5.1 Initialization

The host driver must perform a few steps to initialize the host controller for queuing operation and ensure that the drive supports Serial ATA II Native Command Queuing. The host will first issue an Identify Device command to the drive to determine if it supports the queuing protocol and if so how many outstanding queued commands it supports. If the drives supports the queuing protocol, the host driver software will proceed to initialize the host controller for queuing operation.

The host driver software will set up the Queuing Table Base Address registers with the physical address of the beginning of the FPDMA Vector Table in host memory. The driver will allocate an FPDMA Vector Table and corresponding PRD tables to support the maximum number of queued commands that the drive indicated (up to 32).

Next the host driver software will alert the drive that the host controller supports the First Party DMA auto-activation feature for queued write commands. The host controller informs the drive of this capability by issuing a Set Features command.

5.2 Single SATA II Native Command Queued Read

Table 2 lists the steps in a typical sequence between the host and the device for a Read FPDMA Queued command.

Table 2. Interaction Between Host and Device

Host Actions	Device Actions
Host-to-device Register FIS to issue the Read FPDMA Queued command to the device, BSY=1.	
	Device-to-host Register FIS to allow the host to issue more commands by setting BSY=0.
	DMA Setup FIS issued to host to set the appropriate DMA context for the transfer.
	Data FIS(es) issued to the host until all data for the queued command is transferred.
	Set Device Bits FIS issued to the host to complete the command with interrupt bit set.
Host completes the command and triggers an interrupt	

Note that after issuing the command, host driver software is uninvolved until after the command is complete. The host controller hardware handles setting up the DMA context.

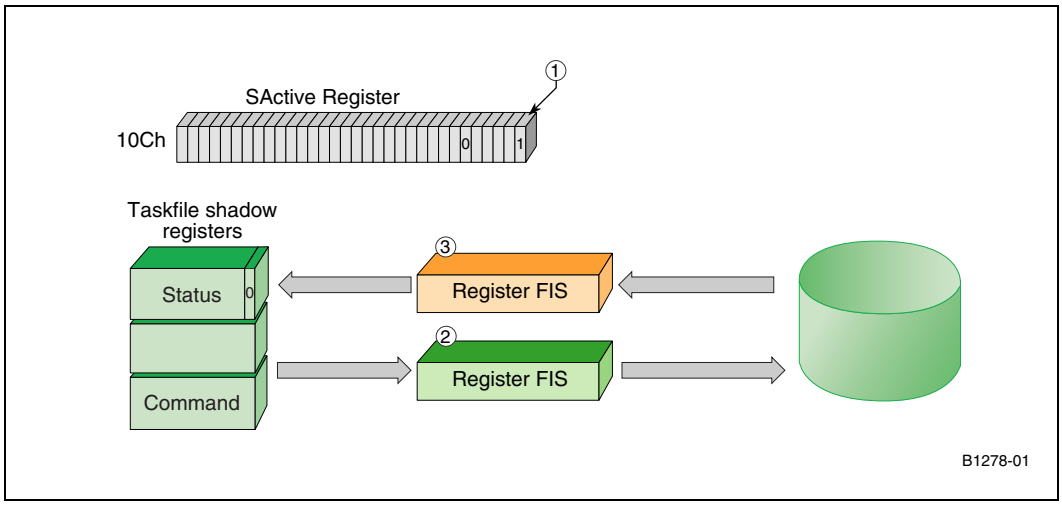
5.3 Performing Multiple Read Commands using SATA II Native Command Queuing

Figure 8, Figure 9, Figure 10 and Figure 11 describe the mechanism in which the multiple reads are completed using the Serial ATA II Native Command Queuing protocol. This example demonstrates commands completing in a different order than the order they were issued in.

5.3.1 Host Issues a Read FPDMA Queued Command with Tag=0

1. The bit 0 in the SActive Register is set to 1 in the host controller.
2. The host issues a Read FPDMA Queued command with Tag=0.
3. This read command is transmitted to the device using a Register FIS. The host controller Status register has BSY=1 meaning no new commands may be issued.
4. The device accepts the command and clears the BSY bit by transmitting a Register FIS to the host controller.

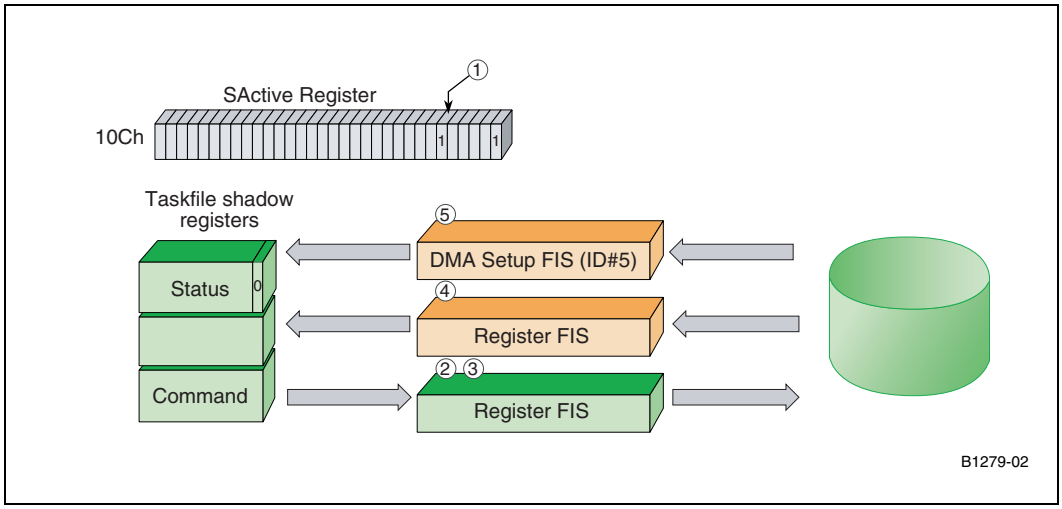
Figure 8. Read FPDMA Queued Command Issued with Tag=0



5.3.2 Host Issues a Read FPDMA Queued Command with Tag=5

1. The bit 5 in the SActive Register is set to 1 in the host controller.
2. The host issues a Read FPDMA Queued command with Tag=5.
3. This read command is transmitted to the device using a Register FIS. The host controller Status register has BSY=1 meaning no new commands may be issued.
4. The device accepts the command and clears the BSY bit by transmitting a Register FIS to the host.
5. The device sends a DMA Setup FIS for Tag=5.

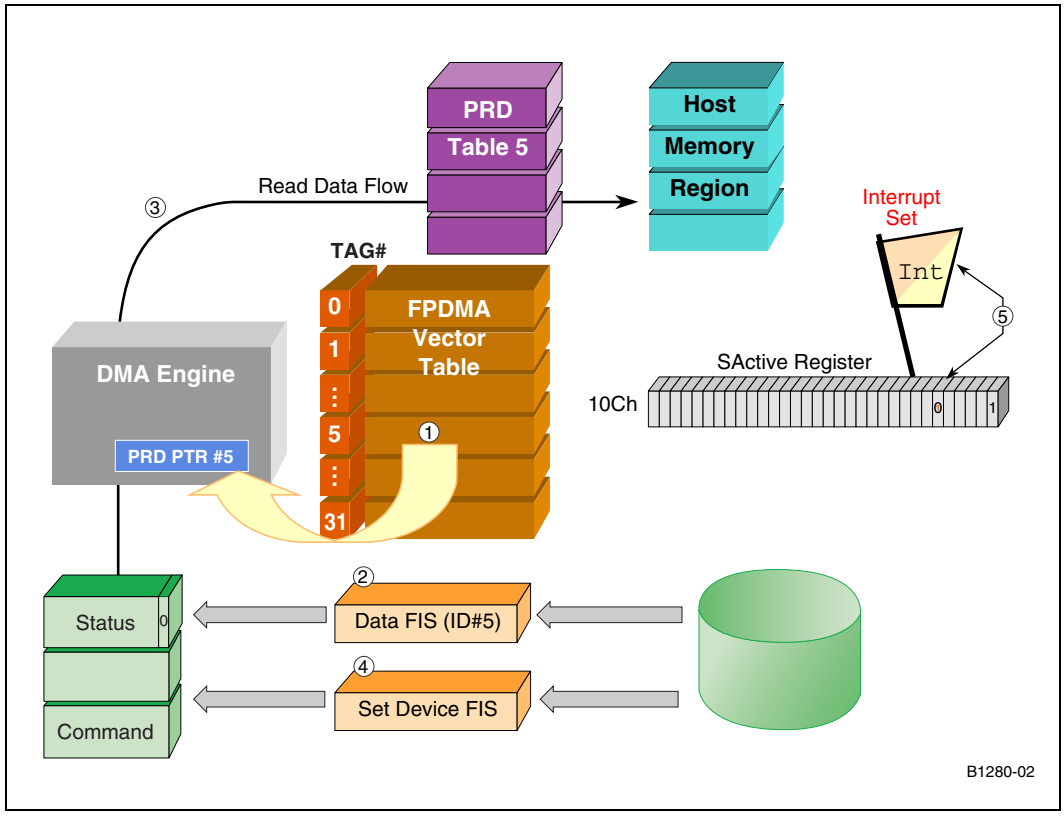
Figure 9. Read FPDMA Queued Command Issued with Tag=5



5.3.3 Data Transferred for Command with Tag=5

1. The PRD pointer that corresponds to Tag=5 is loaded into the host controller DMA engine.
2. The device sends the Data FIS(es) corresponding to Tag=5.
3. The host controller DMA engine directs incoming data into the memory region for the command with Tag=5.
4. The device sends a Set Device Bits FIS with the I interrupt bit set and SActive bit 5 set to 1. This bit being set indicates that the command with Tag=5 is complete.
5. The SActive register bit 5 is cleared in the host controller, and an interrupt is triggered.

Figure 10. Data Transfer and Command Completion for Tag=5

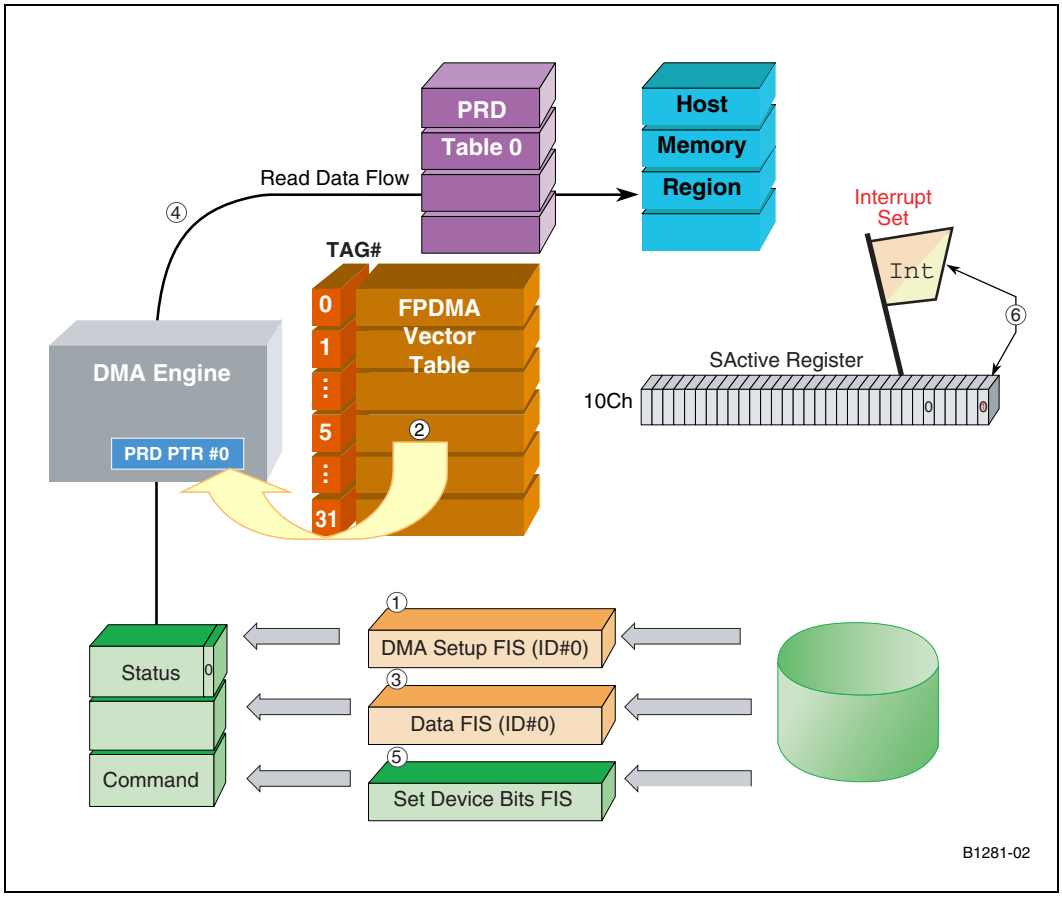


5.3.4 Data Transferred for Command with Tag=0

1. The device sends a DMA Setup FIS for Tag=0.
2. The PRD pointer that corresponds to Tag=0 is loaded into the host controller DMA engine.
3. The device sends the Data FIS(es) corresponding to the Tag=0.
4. The host controller DMA engine directs incoming data into the host memory region for the command with Tag=0.
5. The device sends a Set Device Bits FIS with the I interrupt bit set and the SActive bit 0 set to 1 to indicate the command with Tag=0 is complete.
6. The SActive register bit 0 is cleared in the host controller, and an interrupt is triggered.

Note: **Interrupt Aggregation—**It is possible that the processor has not cleared the previous interrupt (set in Section 5.3.3 step 5) at this point in step 6 above. In this case the host processor will service command completions for both commands. This provides increased efficiency discussed in Section 2.3.2.

Figure 11. Data Transfer and Command Completion for Tag=0



5.4 Error Handling

If the BSY bit is set in the host Status register and an error condition is detected, the device transmits a Register FIS to the host controller with the ERR bit set to one, the BSY bit cleared to zero, and the Error register set to an appropriate value. If the BSY bit is not set in the host Status register and an error condition is detected, the device transmits a Set Device Bits FIS to the host controller with the ERR bit set to one and the Error register set to an appropriate value. The device waits until a Read Log Ext command with a log page of 10h is issued. When this occurs the device aborts any outstanding queued commands and performs cleanup before returning detailed error information including a tag value for the failed last command. The device will also clear all SActive register bits by sending a Set Device Bits FIS to the host with all the bits in SActive field set to one. The host can now reissue any aborted commands and begin sending new commands. The device will cease waiting for the Read Log Ext command if a reset is received.

6.0 Conclusion

The goal of this whitepaper is to provide an overview of the SATA II Native Command Queuing. For a greater understanding of the Serial ATA II Native Command Queuing protocol, an overview of general command queuing and the performance advantages of Serial ATA II Native Command Queuing are discussed as well.

While not discussed in detail in this paper, the host driver software also performs a significant role in making SATA II Native Command Queuing work. For the SATA Host Controller the driver functionality will be implemented in the DPA mode driver. This driver will need to provide the device initialization, set up the Queuing Table Base Address registers, allocate and manage the PRD tables and keep track of queued commands and handle associated errors.

Using command queuing will optimize data transfers between the host and device by allowing 32 different commands to be active simultaneously. By overlapping commands, the disk can reorder commands to maximize throughput. As discussed, Serial ATA II Native Command Queuing delivers much higher performance than Parallel ATA Command Queuing through a streamlined protocol, utilizing hardware to automate the setup of DMA memory regions, and minimizing the number of interrupts required for data transfers. In conclusion, the SATA II Native Command Queuing adds data-handling intelligence that will deliver the performance needed for the next generation of entry-level server, networked storage and high-end PC markets.

7.0 References

The following are additional documents that are referenced in this white paper.

Table 3. Referenced Documents

Document	URL
Intel 31244 PCI-X to Serial ATA Controller Developer's Manual	http://developer.intel.com/design/storage/manuals/273603.htm
Serial ATA 1.0a Specification	http://www.serialata.org/collateral/index.shtml
Serial ATA II: Extensions to Serial ATA 1.0 Specification	http://www.serialata.org/collateral/index.shtml
Serial ATA background white papers and technical briefs	http://www.serialata.org/about/member_wp.shtml
ATA/ATAPI 6 Specification	http://www.techstreet.com/cgi-bin/detail?product_id=932242