Serial ATA

Advanced Host Controller Interface

(AHCI)

Revision 1.0

Please send comments to Amber Huffman
amber.huffman@intel.com

Table of Contents

Table of Figures

1 Introduction

1.1 Overview

AHCI describes a PCI class device that acts as an interface between system memory and SATA devices.

AHCI host devices (referred to as host bus adapters, or HBA) may support from 1 to 32 ports. An HBA must support ATA and ATAPI devices, and must support both the PIO and DMA protocols. The HBA may optionally support a command list on each port for overhead reduction, and to support Serial ATA Native Command Queuing via the FPDMA Queued Command protocol for each device of up to 32 entries. The HBA may optionally support 64-bit addressing.

AHCI describes a system memory structure which contains a generic area for control and status, and a table of entries describing a command list (an HBA which does not support a command list shall have a depth of one for this table). Each command list entry contains information necessary to program an SATA device, and a pointer to a descriptor table for transferring data between system memory and the device.

1.2 Scope

AHCI encompasses a PCI device. It contains a PCI BAR (Base Address Register) to implement native SATA features. AHCI contains definitions for the following features:

- Support for 32 ports
- Elimination of Master / Slave Handling
- Hot Plug
- HW Assisted Native Command Queuing
- Cold device presence detect
- Activity LED generation
- 64-bit addressing
- Large LBA support
- Power Management
- Staggered Spin-up
- Serial ATA superset registers
- Port Multiplier

1.3 Outside of Scope

AHCI does not contain information relevant to implementing the Transport, Link or Phy layers of Serial ATA as this is wholly described in the Serial ATA 1.0a specification. It does not describe enclosure management services, as these are covered in the Serial ATA II: Extensions to Serial ATA 1.0a revision 1.1 specification. It also does not specify ATA legacy behavior, such as the legacy I/O ranges, or bus master IDE. Allowances have been made in AHCI so that an HBA may implement these features for backward compatibility with older operating systems (for example, the location of the memory BAR for AHCI is after the BAR locations for both native IDE and bus master IDE).

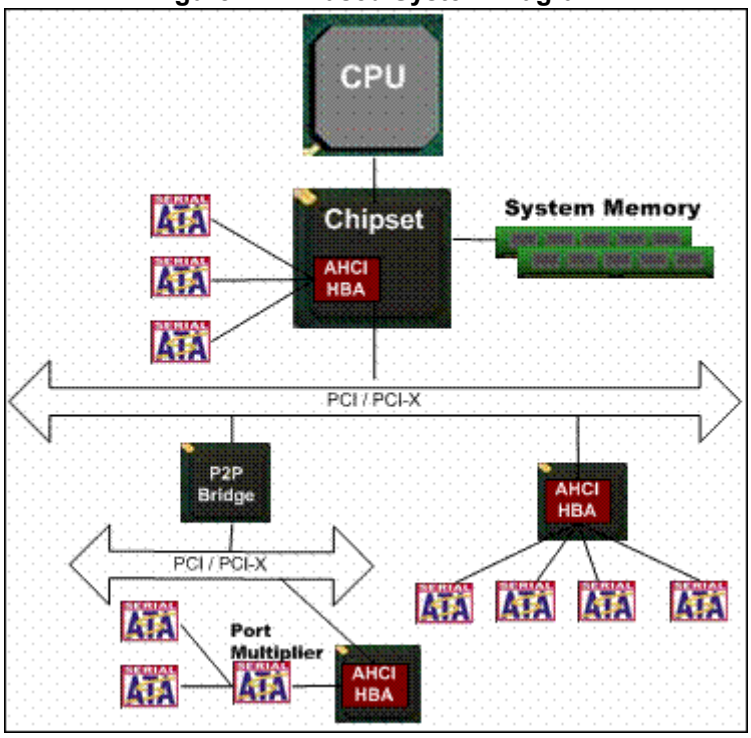
1.4 Block Diagram

In Figure 1, several AHCI HBAs are attached in an Intel Architecture based computer system. One HBA is integrated in the core chipset. Another sits off the first available PCI/PCI-X bus. (PCI is used as a reference name. The bus can be any PCI-like bus, such as PCI-X, PCI-Express, HyperTransport, etc.)

A final HBA sits off a second PCI bus that exists behind a PCI-PCI bridge. This last HBA has one port attached to a Port Multiplier

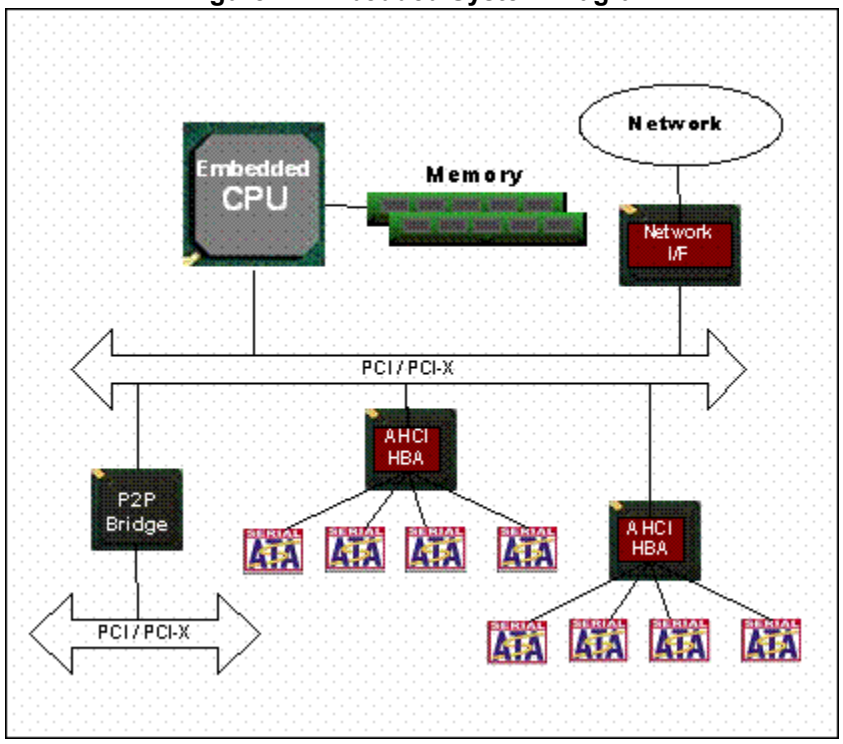
All the devices talk to system memory attached to the chipset.

Figure 1: IA Based System Diagram



In Figure 2, two HBA are connected to an embedded CPU with its own local memory. This configuration would most likely be used in a RAID-type environment.

Figure 2: Embedded System Diagram



1.5 Conventions

Hardware must return '0' for all bits and registers that are marked as reserved, and software must write all reserved bits and registers with the value of '0'.

Inside the register section, the following abbreviations are used:

RO	Read Only
RW	Read Write
RWC	Read/Write '1' to clear
RW1	Read/Write '1' to set
Impl. Spec.	Implementation Specific – the HBA has the freedom to choose its implementation.
HwInit	The default state is dependent on device and system configuration. The value is initialized at reset, either by an expansion ROM, or in the case of integrated devices, by a platform BIOS.

When a register bit is referred to in the document, the convention used is "Register Symbol.Field Symbol". For example, the configuration space PCI command register parity error response bit is referred to by the name CMD.PEE. If the register field is an array of bits, the field will be referred to as "Register Symbol.Field Symbol(array offset)".

When a memory field is referred to in the document, the convention used is "Register Name[Offset Symbol]". For example, the pointer to the Command Header of port 0 is referred to by the name P0CLB[CH0].

1.6 Terminology

HBA	Host Bus Adapter – refers to the silicon that implements the AHCI specification to communicate between system memory and SATA devices.
H2D	HBA to Device
D2H	Device to HBA
System Memory	DRAM or "main" memory of a computer system, used to communicate data and command information between the host processor and the SATA device.
Register Memory	Registers allocated in the memory space of the HBA. These registers are physically implemented in the HBA.
Command List	Defines commands located in system memory that an HBA processes. This is a list that may be 1 to 32 entries called Command Slots, and may contain any type of ATA or ATAPI command. The command list is advanced when the BSY, DRQ, and ERR bits of an SATA device is cleared
Command Slot	One of the entries in the command list, which contains the command to execute. Up to 32 slots are supported in a Command List.
Queue	Indicates the specific ATA command queue inside an SATA device. This is differentiated from the command list in that a queue shall only exist in an SATA device when all the commands in the HBA's command list are of the SATA queued type
Port	A physical port on the HBA, with a set of registers that control the DMA and link operations. A physical port may have several devices attached to it via a Port Multiplier
Device	A physical device, such as an HDD (hard disk drive) or ODD (optical disk drive) that is either directly attached to an HBA port, or is attached through a Port Multiplier to an HBA port.

1.7 Theory of Operation

AHCI is defined to take the basics of the scatter/gather list concept of Bus Master IDE, and expand it to reduce CPU/software overhead and provide support for Serial ATA features such as hot plug, power management, and accessing of several devices without performing master/slave emulation.

Communication between a device and software moves from the task file via byte-wide accesses to a command FIS located in system memory that is fetched by the HBA. This reduces command setup time significantly, allowing for many more devices to be added to a single host controller. Software no longer communicates directly to a device via the task file.

AHCI is defined to keep the HBA relatively simple so that it can act as a data mover. An HBA implementing AHCI does not parse any of the ATA or ATAPI commands as they are transferred to the device, although it is not prohibited from doing so.

All data transfers between the device and system memory occur through the HBA acting as a bus master to system memory. Whether the transaction is of a DMA type or a PIO type, the HBA fetches and stores data to memory, offloading the CPU. There is no data port that can be accessed.

All transfers are performed in a DMA fashion and the use of the PIO command type is strongly discouraged. PIO has limited support for errors – for example, the ending status field of a PIO transfer is given to the HBA during the PIO Setup FIS, before the data is transferred. However, some commands may only be performed via PIO commands (such as IDENTIFY DEVICE). Some HBA implementations may limit PIO support to one DRQ block of data per command.

The AHCI defines a standard mechanism for implementing a SATA command queue using the DMA Setup FIS. An HBA which supports queuing has individual slots in the **Command List** allocated in system memory for all the commands. Software can place a command into any empty slot, and upon notifying the HBA via a register access, the HBA shall fetch the command and transfer it. The tag that is returned in the DMA Setup FIS is used as an index into the command list to get the scatter/gather list used in the transfer.

This command list can be used by system software and the HBA even when non-queued commands need to be transferred. System software can still place multiple commands in the list, whether DMA, PIO, or ATAPI, and the HBA shall walk the list and transfer them.

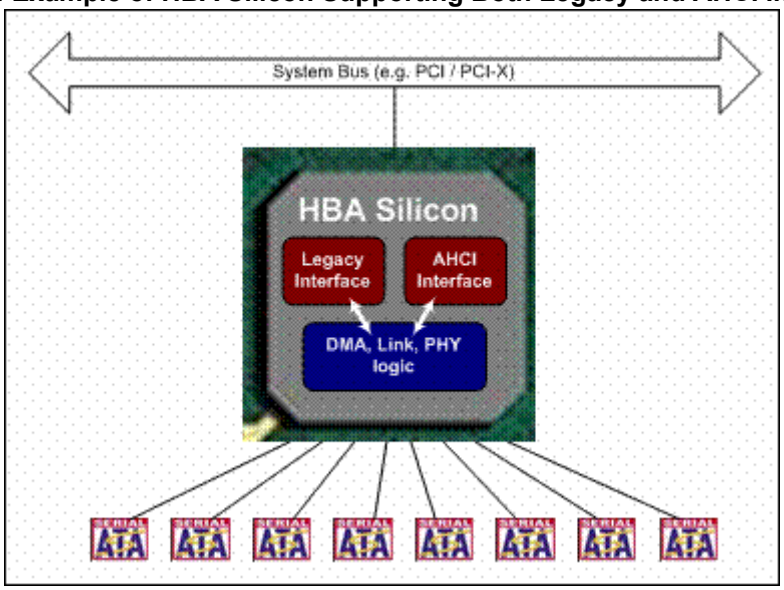
This multiple-use of the command list is achieved by the HBA only moving its command list pointer when the BSY, DRQ, and ERR bits are cleared by the device. It is system software's responsibility to not mix queued and non-queued commands in the command list.

1.8 Interaction with Legacy Software

AHCI is a self-contained specification that is intended to support all aspects of communicating with SATA devices, without having to utilize any legacy features such as shadow copies of the task file, snooping of bits in commands, etc.

HBAs that support legacy software mechanisms must do so in a fashion that is transparent to AHCI. Legacy registers are not allowed to affect any bits in AHCI registers, nor is AHCI software allowed to affect any bits in legacy registers. Software written for AHCI is not allowed to utilize any of the legacy mechanisms to program devices. In essence, an HBA that supports both mechanisms must isolate its legacy and AHCI engines, as shown in Figure 3.

Figure 3: Example of HBA Silicon Supporting Both Legacy and AHCI Interfaces



How an HBA that runs legacy software supports more than 4 ports is beyond the scope of this specification. How software transitions between legacy and AHCI modes of operation is beyond the scope of this specification.

1.9 References

The AHCI utilizes the following documents as references:

PCI Specification, Revision 2.3
- http://www.pcisig.com

PCI Power Management Specification
- http://www.pcisig.com

ATA/ATAPI-6
- http://www.t13.org

Serial ATA 1.0a, Serial ATA II: Extensions to Serial ATA 1.0a revision 1.1, Serial ATA II: Port Multiplier revision 1.1, Serial ATA II: Port Selector revision 1.0
- http://www.serialata.org

Microsoft's Storage Device Class Power Management Specification:
- http://www.microsoft.com/hwdev/resources/specs/pmref/default.asp

2 HBA Configuration Registers

This section details how the PCI Header and PCI Capabilities should be constructed for an AHCI HBA. The fields shown are duplicated from the appropriate PCI specifications. The PCI specifications are the normative specifications for these registers and this section details additional requirements for an AHCI HBA.

Start (hex)	End (hex)	Name
00	3F	PCI Header
PMCAP	PMCAP+7	PCI Power Management Capability
MSICAP	MSICAP+9	Message Signaled Interrupt Capability

2.1 PCI Header

Start (hex)	End (hex)	Symbol	Name
00	03	ID	Identifiers
04	05	CMD	Command Register
06	07	STS	Device Status
08	08	RID	Revision ID
09	0B	CC	Class Codes
0C	0C	CLS	Cache Line Size
0D	0D	MLT	Master Latency Timer
0E	0E	HTYPE	Header Type
0F	0F	BIST	Built In Self Test (Optional)
10	23	BARS	Other Base Address Registres (Optional) <BAR0-4>
24	27	ABAR	AHCI Base Address <BAR5>
2C	2F	SS	Subsystem Identifiers
30	33	EROM	Expansion ROM Base Address (Optional)
34	34	CAP	Capabilities Pointer
3C	3D	INTR	Interrupt Information
3E	3E	MGNT	Min Grant (Optional)
3F	3F	MLAT	Max Latency (Optional)

2.1.1 Offset 00h: ID - Identifiers

Bits	Type	Reset	Description
31:16	RO	Impl. Spec.	**Device ID (DID):** Indicates what device number assigned by the vendor.
15:00	RO	Impl. Spec.	**Vendor ID (VID):** 16-bit field which indicates the company vendor, assigned by the PCI SIG.

2.1.2 Offset 04h: CMD - Command

Bit	Type	Reset	Description
15:11	*RO*	*0*	*Reserved*
10	RW	0	**Interrupt Disable (ID):** Disables the HBA from generating interrupts. This bit does not have any effect on MSI operation.
09	RW/RO	0	**Fast Back-to-Back Enable (FBE):** Allows the HBA to generate fast back-to-back cycles to different devices.
08	RW/RO	0	**SERR# Enable (SEE):** When set, the HBA is allowed to generate SERR# on any event that is enabled for SERR# generation. When cleared, it is not.
07	*RO*	*0*	***Wait Cycle Enable (WCC):*** *Reserved.*
06	RW/RO	0	**Parity Error Response Enable (PEE):** When set, the HBA shall generate PERR# when a data parity error is detected.
05	*RO*	*0*	***VGA Palette Snooping Enable (VGA):*** *Reserved*
04	RW/RO	Impl Spec	**Memory Write and Invalidate Enable (MWIE):** Allows the HBA to use the memory write and invalidate command. This feature is not necessary if the controller does not generate this command.
03	*RO*	*0*	***Special Cycle Enable (SCE):*** *Reserved*

02	RW	0	**Bus Master Enable (BME):** Controls the HBA's ability to act as a master for data transfers. When this bit is cleared, HBA activity stops and any active DMA engines return to an idle condition. It is the equivalent of clearing the memory space start bits in each port.
01	RW	0	**Memory Space Enable (MSE)**: Controls access to the HBA's register memory space.
00	RW/ RO	Impl Spec	**I/O Space Enable (IOSE):** Controls access to the HBA's target I/O space. If the HBA also supports bus master IDE, this bit must be read/write. This bit should be read-only if bus master IDE is not supported by the HBA.

2.1.3 Offset 06h: STS - Device Status

Bit	Type	Reset	Description
15	RWC	0	**Detected Parity Error (DPE):** Set when the HBA detects a parity error on its interface.
14	RWC	0	**Signaled System Error (SSE):** Set when the HBA host generates SERR#.
13	RWC	0	**Received Master-Abort (RMA):** Set when the HBA receives a master abort to a cycle it generated.
12	RWC	0	**Received Target Abort (RTA):** Set when the HBA receives a target abort to a cycle it generated.
11	RWC	0	**Signaled Target-Abort (STA):** Set when the HBA terminates with a target abort.
10:09	RO	Impl. Spec.	**DEVSEL# Timing (DEVT):** Controls the device select time for the HBA's PCI interface.
08	RWC	0	**Master Data Pariy Error Detected (DPD):** Set when the HBA, as a master, either detects a parity error or sees the parity error line asserted, and the parity error response bit (bit 6 of the command register) is set.
07	RO	Impl. Spec.	**Fast Back-to-Back Capable (FBC):** Indicates whether the HBA can accept fast back-to-back cycles.
06	*RO*	*0*	*Reserved*
05	RO	Impl. Spec.	**66 MHz Capable (C66):** Indicates whether the HBA can operate at 66 MHz.
04	RO	1	**Capabilities List (CL):** Indicates the presence of a capabilities list. The HBA must support the PCI power management capability as a minimum.
03	RO	0	**Interrupt Status (IS):** Indicates the interrupt status of the device (1 = asserted).
02:00	*RO*	*0*	*Reserved*

2.1.4 Offset 08h: RID - Revision ID

Bits	Type	Reset	Description
07:00	RO	00h	**Revision ID (RID):** Indicates stepping of the HBA hardware.

2.1.5 Offset 09h: CC - Class Code

Bits	Type	Reset	Description
23:16	RO	01h	**Base Class Code (BCC):** Indicates that this is a mass storage device.
15:08	RO	Impl Spec	**Sub Class Code (SCC):** When set to 06h, indicates that this is a Serial ATA device.
07:00	RO	Impl Spec	**Programming Interface (PI):** When set to 01h and the Sub Class Code is set to 06h, indicates that this is an AHCI HBA that has a major revision of 1 (as specified in the AHCI Version register).

Informative Note: For HBAs that support RAID, the Sub Class Code reset value should be 04h and the Programming Interface reset value should be 00h.

2.1.6 Offset 0Ch: CLS – Cache Line Size

Bits	Type	Reset	Description
07:00	RW	00h	**Cache Line Size (CLS):** Indicates the cache line size for use with the memory write and invalidate command, and as an indication on when to use the memory read multiple, memory read line, or memory read commands.

2.1.7 Offset 0Dh: MLT – Master Latency Timer

Bits	Type	Reset	Description
07:00	RW	00h	**Master Latency Timer (MLT):** Indicates the number of clocks the HBA is allowed to act as a master on PCI.

2.1.8 Offset 0Eh: HTYPE – Header Type

Bits	Type	Reset	Description
07	RO	Impl Spec	**Multi-Function Device (MFD):** Indicates whether the HBA is part of a multi-function device.
06:00	RO	00h	**Header Layout (HL):** Indicates that the HBA uses a target device layout.

2.1.9 Offset 0Fh: BIST – Built In Self Test (Optional)

The following register is optional, but if implemented, must look as follows.

Bits	Type	Reset	Description
07	RO	1	**BIST Capable (BC):** Indicates whether the HBA has a BIST function. This does not indicate SATA BIST capability – this is only for an HBA related BIST function.
06	RW	0	**Stat BIST (SB):** Software sets this bit to invoke BIST. The HBA clears this bit when BIST is complete.
05:04	*RO*	*00*	*Reserved*
03:00	RO	0h	**Completion Code (CC):** Indicates the completion code status of BIST. A non-zero value indicates a failure.

2.1.10 Offset 10h – 20h: BARS – Other Base Addresses (Optional)

These registers allocate memory or I/O spaces for other BARs. An example application of these BARs is to implement the native IDE and bus master IDE ranges for an HBA that wishes to support legacy software.

2.1.11 Offset 24h: ABAR – AHCI Base Address

This register allocates space for the HBA memory registers defined in section 3. The ABAR must be allocated to contain enough space for the global AHCI registers, the port specific registers for each port, and any vendor specific space (if needed). It is permissible to have vendor specific space after the port specific registers for the last HBA port.

Bit	Type	Reset	Description
31:13	RW	0	**Base Address (BA):** Base address of register memory space. This represents a memory space for support of 32 ports. For HBAs that support less than 32-ports, more bits are allowed to be RW, and therefore less memory space is consumed. For HBAs that have vendor specific space at the end of the port specific memory space, more bits are allowed to be RO such that more memory space is consumed.
12:04	*RO*	*0*	*Reserved*
03	RO	0	**Prefetchable (PF):** Indicates that this range is not pre-fetchable
02:01	RO	00	**Type (TP):** Indicates that this range can be mapped anywhere in 32-bit address space
00	RO	0	**Resource Type Indicator (RTE):** Indicates a request for register memory space.

2.1.12 Offset 2Ch: SS - Sub System Identifiers

Bits	Type	Reset	Description
31:16	RO	Impl Spec	**Subsystem ID (SSID):** Indicates the sub-system identifier.
15:00	RO	Impl Spec	**Subsystem Vendor ID (SSVID):** Indicates the sub-system vendor identifier

2.1.13 Offset 30h: EROM – Expansion ROM (Optional)

Bit	Type	Reset	Description
31:00	RW	Impl Spec	**ROM Base Address (RBA):** Indicates the base address of the HBA expansion ROM..

2.1.14 Offset 34h: CAP – Capabilities Pointer

Bit	Type	Reset	Description
7:0	RO	PMCAP	**Capability Pointer (CP):** Indicates the first capability pointer offset. It points to the PCI Power management capability offset.

2.1.15 Offset 3Ch: INTR - Interrupt Information

Bits	Type	Reset	Description
15:08	RO	Impl Spec	**Interrupt Pin (IPIN):** This indicates the interrupt pin the HBA uses.
07:00	RW	00h	**Interrupt Line (ILINE):** Software written value to indicate which interrupt line (vector) the interrupt is connected to. No hardware action is taken on this register.

Informative Note: If the HBA is a single function PCI device, then INTR.IPIN should be set to 01h to indicate the INTA# pin.

2.1.16 Offset 3Eh: MGNT – Minimum Grant (Optional)

Bits	Type	Reset	Description
07:00	RO	Impl Spec	**Grant (GNT):** Indicates the minimum grant time (in ¼ microseconds) that the device wishes grant asserted.

2.1.17 Offset 3Fh: MLAT – Maximum Latency (Optional)

Bits	Type	Reset	Description
07:00	RO	Impl Spec	**Latency (LAT): I**ndicates the maximum latency (in ¼ microseconds) that the device can withstand.

2.2 PCI Power Management Capabilities

Start (hex)	End (hex)	Symbol	Name
PMCAP	PMCAP+1	PID	PCI Power Management Capability ID
PMCAP+2	PMCAP+3	PC	PCI Power Management Capabilities
PMCAP+4	PMCAP+7	PMCS	PCI Power Management Control and Status

2.2.1 Offset PMCAP: PID - PCI Power Management Capability ID

Bit	Type	Reset	Description
15:08	RO	Impl Spec	**Next Capability (NEXT):** Indicates the location of the next capability item in the list. This can be other capability pointers (such as Message Signaled Interrupts, PCI-X, or PCI-Express) or it can be the last item in the list.
07:00	RO	01h	**Cap ID (CID):** Indicates that this pointer is a PCI power management.

2.2.2 Offset PMCAP + 2h: PC – PCI Power Management Capabilities

Bit	Type	Reset	Description
15:11	RO	Impl Spec	**PME_Support (PSUP):** Indicates the states that can generate PME#. The encoding of this field is as follows:

Bit	State
15	When set, PME# can be generated from $D3_{COLD}$. When cleared, PME# cannot be generated from $D3_{COLD}$. This bit may be a '1' or '0' for AHCI HBAs.
14	When set, PME# can be generated from $D3_{HOT}$. When cleared, PME# cannot be generated from $D3_{HOT}$. This bit must be '1' for AHCI HBAs.
13	This bit must be '0' for AHCI HBAs. D2 is not a supported HBA state. The behavior is undefined if the HBA is placed in this state.
12	This bit must be '0' for AHCI HBAs. D1 is not a supported HBA state. The behavior is undefined if the HBA is placed in this state.
11	This bit must be '0' for AHCI HBAs. PME# from D0 is not supported – interrupts are used in D0.

10	RO	0	**D2_Support (D2S):** The D2 state is not supported for AHCI HBAs.
09	RO	0	**D1_Support (D1S):** The D1 state is not supported for AHCI HBAs.
08:06	RO	Impl Spec	**Aux_Current (AUXC):** Reports the maximum Suspend well current required when in the $D3_{COLD}$ state. Refer to the PCI specification for definition of values.
05	RO	Impl Spec	**Device Specific Initialization (DSI):** Indicates whether device-specific initialization is required.
04	*RO*	*0*	*Reserved*
03	RO	0	**PME Clock (PMEC):** Indicates that PCI clock is not required to generate PME#.
02:00	RO	010	**Version (VS):** Indicates support for Revision 1.1 of the *PCI Power Management Specification.*

2.2.3 Offset PMCAP + 4h: PMCS – PCI Power Management Control And Status

Bit	Type	Reset	Description
15	RWC	Impl Spec	**PME Status (PMES):** Set when the HBA generates PME#. If the HBA supports waking from $D3_{COLD}$, this bit is indeterminate at system boot. If the HBA does not support waking from $D3_{COLD}$, this bit is '0' at system boot.
14:09	*RO*	*0*	*Reserved – AHCI HBA does not implement the data register.*
08	RW	Impl Spec	**PME Enable (PMEE):** When set, the HBA asserts the PME# signal when PMES is set. If the HBA supports waking from $D3_{COLD}$, this bit is indeterminate at system boot. If the HBA does not support waking from $D3_{COLD}$, this bit is '0' at system boot.
07:02	*RO*	*0*	*Reserved*
01:00	R/W	00	**Power State (PS):** This field is used both to determine the current power state of the HBA and to set a new power state. The values are: 00 – D0 state 11 – $D3_{HOT}$ state The D1 and D2 states are not supported for AHCI HBAs. When in the $D3_{HOT}$ state, the HBA's configuration space is available, but the register memory spaces are not. Additionally, interrupts are blocked.

2.3 Message Signaled Interrupt Capability (Optional)

Start (hex)	End (hex)	Symbol	Name
MSICAP	MSICAP+1	MID	Message Signaled Interrupt Capability ID
MSICAP+2	MSICAP+3	MC	Message Signaled Interrupt Message Control
MSICAP+4	MSICAP+7	MA	Message Signaled Interrupt Message Address
MSICAP+8 (MC.C64=0) MSICAP+C (MC.C64=1)	MSICAP+9 MSICAP.D	MD	Message Signaled Interrupt Message Data
MSICAP+8 (MC.C64=1)	MSICAP+B	MUA	Message Signaled Interrupt Upper Address (Optional)

2.3.1 Offset MSICAP: MID – Message Signaled Interrupt Identifiers

Bits	Type	Reset	Description
15:08	RO	Impl Spec	**Next Pointer (NEXT):** Indicates the next item in the list. This can be other capability pointers (such as PCI-X or PCI-Express) or it can be the last item in the list.
07:00	RO	05h	**Capability ID (CID):** Capabilities ID indicates MSI.

2.3.2 Offset MSICAP + 2h: MC – Message Signaled Interrupt Message Control

Bits	Type	Reset	Description
15:08	*RO*	*0*	*Reserved*
07	RO	Impl Spec	**64 Bit Address Capable (C64):** Specifies whether capable of generating 64-bit messages.
06:04	RW	000	**Multiple Message Enable (MME):** Indicates the number of messages the HBA should assert. See section 10.6.2.2. If the value programmed into this field exceeds the MMC field in this register, only a single message shall be generated.
03:01	RO	Impl Spec	**Multiple Message Capable (MMC):** Indicates the number of messages the HBA wishes to assert. See section 10.6.2.2.
00	RW	0	**MSI Enable (MSIE):** If set, MSI is enabled and traditional interrupt pins are not used to generate interrupts.

2.3.3 Offset MSICAP + 4h: MA – Message Signaled Interrupt Message Address

Bits	Type	Reset	Description
31:02	RW	0	**Address (ADDR):** Lower 32 bits of the system specified message address, always DW aligned.
01:00	*RO*	*00*	*Reserved*

2.3.4 Offset MSICAP + (8h or Ch): MD – Message Signaled Interrupt Message Data

Bits	Type	Reset	Description
15:00	RW	0	**Data (Data):** This 16-bit field is programmed by system software if MSI is enabled. Its content is driven onto the lower word (PCI AD[15:0]) during the data phase of the MSI memory write transaction.

2.3.5 Offset MSICAP + 8h: MUA – Message Signaled Interrupt Upper Address (Optional)

Bits	Type	Reset	Description
31:00	RW	0	**Upper Address (UADDR):** Upper 32 bits of the system specified message address. This register is optional and only implemented if MC.C64=1.

2.4 Other Capability Pointers

Though not mentioned in this specification, other capability pointers may be necessary, depending upon the implementation space. Examples would be the PCI-X capability for PCI-X implementations, PCI-Express capability for PCI-Express implementations, and potentially the vendor specific capability pointer.

These capabilities are beyond the scope of this specification.

3 HBA Memory Registers

The memory mapped registers within the HBA exist in non-cacheable memory space. Additionally, locked accesses are not supported. If software attempts to perform locked transactions to the registers, indeterminate results may occur. Register accesses shall have a maximum size of 64-bits; 64-bit access must not cross an 8-byte alignment boundary.

The registers are broken into two sections – global registers and port control. All registers that start below address 100h are global and meant to apply to the entire HBA. The port control registers are the same for all ports, and there are as many register banks as there are ports.

All registers not defined and all reserved bits within registers return '0' when read.

Start	End	Description
00	1F	Generic Host Control
20	*9F*	*Reserved*
A0	FF	Vendor Specific registers
100	17F	Port 0 port control registers
180	1FF	Port 1 port control registers
200	FFF	(Ports 2 – port 29 control registers)
1000	107F	Port 30 port control registers
1080	10FF	Port 31 port control registers

3.1 Generic Host Control

The following registers apply to the entire HBA.

Start	End	Symbol	Description
00	03	CAP	Host Capabilities
04	07	GHC	Global Host Control
08	0B	IS	Interrupt Status
0C	0F	PI	Ports Implemented
10	13	VS	Version

3.1.1 Offset 00h: CAP – HBA Capabilities

This register indicates basic capabilities of the HBA to driver software.

Bit	Type	Reset	Description
31	RO	Impl Spec	**Supports 64-bit Addressing (S64A):** Indicates whether the HBA can access 64-bit data structures. If true, the HBA shall make the 32-bit upper bits of the port DMA Descriptor, the PRD Base, and each PRD entry read/write. If cleared, these are read-only and treated as '0' by the HBA.
30	RO	Impl Spec	**Supports Native Command Queuing (SNCQ):** Indicates whether the HBA supports Serial ATA native command queuing. If set to '1', an HBA shall handle DMA Setup FISes natively, and shall handle the auto-activate optimization through that FIS. If cleared to '0', native command queuing is not supported and software should not issue any native command queuing commands.
29	*RO*	*0*	*Reserved*
28	RO	HwInit	**Supports Interlock Switch (SIS):** Indicates whether the HBA supports interlock switches on its ports for use in hot plug operations. This value is loaded by the BIOS prior to OS initialization.
27	RO	HwInit	**Supports Staggered Spin-up (SSS):** When set to '1', indicates that the HBA supports staggered spin-up on its ports, for use in balancing power spikes. This value is loaded by the BIOS prior to OS initiallization.
26	RO	Impl. Spec	**Supports Aggressive Link Power Management (SALP):** When set to '1', indicates that the HBA can support auto-generating link requests to the Partial or Slumber states when there are no commands to process. Refer to section 8.3.1.3.
25	RO	Impl. Spec	**Supports Activity LED (SAL):** When set to '1', indicates that the HBA supports a single output pin which indicates activity. This pin can be connected to an LED on the platform to indicate device activity on any drive. See section 10.10 for more information.

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>24</td><td>RO</td><td>Impl. Spec</td><td>Supports Command List Override (SCLO): When set to ‘1’, indicates that the HBA supports the PxCMD.CLO bit and its associated function. When cleared to ‘0’, the HBA is not capable of clearing the BSY and DRQ bits in the Status register in order to issue a software reset if these bits are still set from a previous operation.</td></tr>
<tr><td>23:20</td><td>RO</td><td>Impl. Spec</td><td>Interface Speed Support (ISS): Indicates the maximum speed the HBA can support on its ports. These encodings match the PxSCTL.DET.SPD field, which is programmable by system software. Values are:
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>0000</td><td>Reserved</td></tr>
<tr><td>0001</td><td>Gen 1 (1.5 Gbps)</td></tr>
<tr><td>0010</td><td>Gen 1 (1.5 Gbps) and Gen 2 (3 Gbps)</td></tr>
<tr><td>0011 - 1111</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>19</td><td>RO</td><td>0</td><td>Supports Non-Zero DMA Offsets (SNZO): When set to ‘1’, indicates that the HBA can support non-zero DMA offsets for DMA Setup FISes. This bit is reserved for future AHCI enhancements. AHCI 1.0 HBAs must have this bit cleared to ‘0’.</td></tr>
<tr><td>18</td><td>RO</td><td>Impl Spec</td><td>Supports AHCI mode only (SAM): The SATA controller may optionally support AHCI access mechanisms only. A value of '0' indicates that in addition to the native AHCI mechanism (via ABAR), the SATA controller implements a legacy, task-file based register interface such as SFF-8038i. A value of '1' indicates that the SATA controller does not implement a legacy, task-file based register interface.</td></tr>
<tr><td>17</td><td>RO</td><td>Impl. Spec</td><td>Supports Port Multiplier (SPM): Indicates whether the HBA can support a Port Multiplier. When set, a Port Multiplier using command-based switching is supported. When cleared to ‘0’, a Port Multiplier is not supported, and a Port Multiplier may not be attached to this HBA.</td></tr>
<tr><td>16</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>15</td><td>RO</td><td>Impl Spec</td><td>PIO Multiple DRQ Block (PMD): If set to ‘1’, the HBA supports multiple DRQ block data transfers for the PIO command protocol. If cleared to ‘0’ the HBA only supports single DRQ block data transfers for the PIO command protocol.</td></tr>
<tr><td>14</td><td>RO</td><td>Impl Spec.</td><td>Slumber State Capable (SSC): Indicates whether the HBA can support transitions to the Slumber state. When cleared to ‘0’, software must not allow the HBA to initiate transitions to the Slumber state via agressive link power management nor the PxCMD.ICC field in each port, and the PxSCTL.IPM field in each port must be programmed to disallow device initiated Slumber requests. When set to ‘1’, HBA and device initiated Slumber requests can be supported.</td></tr>
<tr><td>13</td><td>RO</td><td>Impl Spec.</td><td>Partial State Capable (PSC): Indicates whether the HBA can support transitions to the Partial state. When cleared to ‘0’, software must not allow the HBA to initiate transitions to the Partial state via agressive link power management nor the PxCMD.ICC field in each port, and the PxSCTL.IPM field in each port must be programmed to disallow device initiated Partial requests. When set to ‘1’, HBA and device initiated Partial requests can be supported.</td></tr>
<tr><td>12:08</td><td>RO</td><td>Impl. Spec.</td><td>Number of Command Slots (NCS): 0’s based value indicating the number of command slots supported by this HBA. A minimum of 1 and maximum of 32 slots can be supported.</td></tr>
<tr><td>07:05</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>04:00</td><td>RO</td><td>Impl. Spec.</td><td>Number of Ports (NP): 0’s based value indicating the maximum number of ports supported by the HBA silicon. A maximum of 32 ports can be supported. A value of ‘0h’, indicating one port, is the minimum requirement. Note that the number of ports indicated in this field may be more than the number of ports indicated in the GHC.PI register.</td></tr>
</table>

3.1.2 Offset 04h: GHC – Global HBA Control

This register controls various global actions of the HBA.

Bit	Type	Reset	Description

31	RW/RO	Impl Spec	**AHCI Enable (AE):** When set, indicates that communication to the HBA shall be via AHCI mechanisms. This can be used by an HBA that supports both legacy mechanisms (such as SFF-8038i) and AHCI to know when the HBA is running under an AHCI driver. When set, software shall only communicate with the HBA using AHCI. When cleared, software shall only communicate with the HBA using legacy mechanisms. When cleared FISes are not posted to memory and no commands are sent via AHCI mechanisms. Software shall set this bit to '1' before accessing other AHCI registers. The implementation of this bit is dependent upon the value of the CAP.SAM bit. If CAP.SAM is '0', then GHC.AE shall be read-write and shall have a reset value of '0'. If CAP.SAM is '1', then AE shall be read-only and shall have a reset value of '1'.
30:02	*RO*	*0*	*Reserved*
01	RW	0	**Interrupt Enable (IE):** This global bit enables interrupts from the HBA. When cleared (reset default), all interrupt sources from all ports are disabled. When set, interrupts are enabled.
00	RW1	0	**HBA Reset (HR):** When set by SW, this bit causes an internal reset of the HBA. All state machines that relate to data transfers and queuing shall return to an idle condition, and all ports shall be re-initialized via COMRESET (if staggered spin-up is not supported). If staggered spin-up is supported, then it is the responsibility of software to spin-up each port after the reset has completed. When the HBA has performed the reset action, it shall reset this bit to '0'. A software write of '0' shall have no effect. For a description on which bits are reset when this bit is set, see section 10.4.3.

3.1.3 Offset 08h: IS – Interrupt Status Register

This register indicates which of the ports within the controller have an interrupt pending and require service.

Bit	Type	Reset	Description
31:0	RWC	0	**Interrupt Pending Status (IPS):** If set, indicates that the corresponding port has an interrupt pending. Software can use this information to determine which ports require service after an interrupt. Only the ports that are implemented have a corresponding bit – all other bits are reserved.

3.1.4 Offset 0Ch: PI – Ports Implemented

This register indicates which ports are exposed by the HBA. It is loaded by the BIOS. It indicates which ports that the HBA supports are available for software to use. For example, on an HBA that supports 6 ports as indicated in CAP.NP, only ports 1 and 3 could be available, with ports 0, 2, 4, and 5 being unavailable.

For ports that are not available, software must not read or write to registers within that port.

The intent of this register is to allow system vendors to build platforms that support less than the full number of ports implemented on the HBA silicon.

Bit	Type	Reset	Description
31:0	RO	HwInit	**Port Implemented (PI):** This register is bit significant. If a bit is set to '1', the corresponding port is available for software to use. If a bit is cleared to '0', the port is not available for software to use. The maximum number of bits set to '1' shall not exceed CAP.NP + 1, although the number of bits set in this register may be fewer than CAP.NP + 1. At least one bit shall be set to '1'.

3.1.5 Offset 10h: VS – AHCI Version

This register indicates the major and minor version of the AHCI specification that the HBA implementation supports. The upper two bytes represent the major version number, and the lower two bytes represent

the minor version number. Example: Version 3.12 would be represented as 00030102h. Two versions of the specification are valid: 0.95, and 1.0.

3.1.5.1 VS Value for 0.95 Compliant HBAs

Bit	Type	Reset	Description
31:16	RO	0000h	**Major Version Number (MJR):** Indicates the major version is "0"
15:00	RO	0905h	**Minor Version Number (MNR):** Indicates the minor version is "95".

3.1.5.2 VS Value for 1.0 Compliant HBAs

Bit	Type	Reset	Description
31:16	RO	0001h	**Major Version Number (MJR):** Indicates the major version is "1"
15:00	RO	0000h	**Minor Version Number (MNR):** Indicates the minor version is "0".

3.2 Vendor Specific Registers

Registers at offset A0h to FFh are vendor specific.

3.3 Port Registers (one set per port)

The following registers describe the registers necessary to implement port 0. Additional ports shall have the same register mapping. Port 1 starts at 180h, port 2 starts at 200h, port 3 at 280h, etc. The algorithm for software to determine the offset is as follows:

- Port offset = 100h + (PI Asserted Bit Position * 80h)

Start	End	Symbol	Description
100	103	P0CLB	Port 0 Command List Base Address
104	107	P0CLBU	Port 0 Command List Base Address Upper 32-Bits
108	10B	P0FB	Port 0 FIS Base Address
10C	10F	P0FBU	Port 0 FIS Base Address Upper 32-Bits
110	113	P0IS	Port 0 Interrupt Status
114	117	P0IE	Port 0 Interrupt Enable
118	11B	P0CMD	Port 0 Command
11C	11F	Reserved	Reserved
120	123	P0TFD	Port 0 Task File Data
124	127	P0SIG	Port 0 Signature
128	12B	P0SSTS	Port 0 Serial ATA Status (SCR0: SStatus)
12C	12F	P0SCTL	Port 0 Serial ATA Control (SCR2: SControl)
130	133	P0SERR	Port 0 Serial ATA Error (SCR1: SError)
134	137	P0SACT	Port 0 Serial ATA Active (SCR3: SActive)
138	13B	P0CI	Port 0 Command Issue
13C	16F	Reserved	Reserved
170	17F	P0VS	Port 0 Vendor Specific

3.3.1 Offset 100h: P0CLB – Port 0 Command List Base Address

Bit	Type	Reset	Description
31:10	RW	Impl Spec	**Command List Base Address (CLB):** Indicates the 32-bit base physical address for the command list for this port. This base is used when fetching commands to execute. The structure pointed to by this address range is 1K-bytes in length. This address must be 1K-byte aligned as indicated by bits 09:00 being read only.
09:00	*RO*	*0*	*Reserved*

3.3.2 Offset 104h: P0CLBU – Port 0 Command List Base Address Upper 32-bits

Bit	Type	Reset	Description
31:00	RW/ RO	Impl Spec	**Command List Base Address Upper (CLBU):** Indicates the upper 32-bits for the command list base physical address for this port. This base is used when fetching commands to execute. This register shall be read only '0' for HBAs that do not support 64-bit addressing.

3.3.3 Offset 108h: P0FB – Port 0 FIS Base Address

Bit	Type	Reset	Description

31:08	RW	Impl Spec	**FIS Base Address (FB):** Indicates the 32-bit base physical address for received FISes. The structure pointed to by this address range is 256 bytes in length. This address must be 256-byte aligned as indicated by bits 07:00 being read only.
07:00	*RO*	*0*	*Reserved*

3.3.4 Offset 10Ch: P0FBU – Port 0 FIS Base Address Upper 32-bits

Bit	Type	Reset	Description
31:00	RW/ RO	Impl Spec	**FIS Base Address Upper (FBU):** Indicates the upper 32-bits for the received FIS base physical address for this port. This register shall be read only '0' for HBAs that do not support 64-bit addressing.

3.3.5 Offset 110h: P0IS – Port 0 Interrupt Status

Bit	Type	Reset	Description
31	RWC	0	**Cold Port Detect Status (CPDS):** When set, a device status has changed as detected by the cold presence detect logic. This bit can either be set due to a non-connected port receiving a device, or a connected port having its device removed. This bit is only valid if the port supports cold presence detect as indicated by PxCMD.CPD set to '1'.
30	RWC	0	**Task File Error Status (TFES):** This bit is set whenever the status register is updated by the device and the error bit (bit 0) is set.
29	RWC	0	**Host Bus Fatal Error Status (HBFS):** Indicates that the HBA encountered a host bus error that it cannot recover from, such as a bad software pointer. In PCI, such an indication would be a target or master abort.
28	RWC	0	**Host Bus Data Error Status (HBDS):** Indicates that the HBA encountered a data error (uncorrectable ECC / parity) when reading from or writing to system memory.
27	RWC	0	**Interface Fatal Error Status (IFS):** Indicates that the HBA encountered an error on the Serial ATA interface which caused the transfer to stop. Refer to section 6.1.2.
26	RWC	0	**Interface Non-fatal Error Status (INFS):** Indicates that the HBA encountered an error on the Serial ATA interface but was able to continue operation. Refer to section 6.1.2.
25	*RO*	*0*	*Reserved*
24	RWC	0	**Overflow Status (OFS):** Indicates that the HBA received more bytes from a device than was specified in the PRD table for the command.
23	RWC	0	**Incorrect Port Multiplier Status (IPMS):** Indicates that the HBA received a FIS from a device whose Port Multiplier field did not match what was expected.
22	RO	0	**PhyRdy Change Status (PRCS):** When set to '1' indicates the internal PhyRdy signal changed state. This bit reflects the state of P0SERR.DIAG.N. To clear this bit, software must clear P0SERR.DIAG.N to '0'.
21:08	*RO*	*0*	*Reserved*
07	RWC	0	**Device Interlock Status (DIS):** When set, indicates that an interlock switch attached to this port has been opened or closed, which may lead to a change in the connection state of the device. This bit is only valid if both CAP.SIS and P0CMD.ISP are set to '1'.
06	RO	0	**Port Connect Change Status (PCS):** 1=Change in *Current Connect Status*. 0=No change in *Current Connect Status*. This bit reflects the state of PxSERR.DIAG.X. This bit is only cleared when PxSERR.DIAG.X is cleared.
05	RWC	0	**Descriptor Processed (DPS):** A PRD with the 'I' bit set has transferred all of its data. Refer to section 5.3.2.
04	RO	0	**Unknown FIS Interrupt (UFS):** When set to '1', indicates that an unknown FIS was received and has been copied into system memory. This bit is cleared to '0' by software clearing the PxSERR.DIAG.F bit to '0'. Note that this bit does not directly reflect the PxSERR.DIAG.F bit. PxSERR.DIAG.F is set immediately when an unknown FIS is detected, whereas this bit is set when that FIS is posted to memory. Software should wait to act on an unknown FIS until this bit is set to '1' or the two bits may become out of sync.
03	RWC	0	**Set Device Bits Interrupt (SDBS):** A Set Device Bits FIS has been received with the 'I' bit set and has been copied into system memory.
02	RWC	0	**DMA Setup FIS Interrupt (DSS):** A DMA Setup FIS has been received with the 'I' bit set and has been copied into system memory.
01	RWC	0	**PIO Setup FIS Interrupt (PSS):** A PIO Setup FIS has been received with the 'I' bit set, it has been copied into system memory, and the data related to that FIS has been transferred. This bit shall be set even if the data transfer resulted in an error.

Bit	Type	Reset	Description
00	RWC	0	**Device to Host Register FIS Interrupt (DHRS):** A D2H Register FIS has been received with the 'I' bit set, and has been copied into system memory.

3.3.6 Offset 114h: P0IE – Port 0 Interrupt Enable

This register enables and disables the reporting of the corresponding interrupt to system software. When a bit is set ('1') and the corresponding interrupt condition is active, then an interrupt is generated. Interrupt sources that are disabled ('0') are still reflected in the status registers. This register is symmetrical with the P0IS register.

Bit	Type	Reset	Description
31	RW/ RO	0	**Cold Presence Detect Enable (CPDE):** When set, GHC.IE is set, and P0S.CPDS is set, the HBA shall generate an interrupt. For systems that do not support cold presence detect, this bit shall be a read-only '0'.
30	RW	0	**Task File Error Enable (TFEE):** When set, GHC.IE is set, and P0S.TFES is set, the HBA shall generate an interrupt.
29	RW	0	**Host Bus Fatal Error Enable (HBFE):** When set, GHC.IE is set, and P0IS.HBFS is set, the HBA shall generate an interrupt.
28	RW	0	**Host Bus Data Error Enable (HBDE):** when set, GHC.IE is set, and P0IS.HBDS is set, the HBA shall generate an interrupt..
27	RW	0	**Interface Fatal Error Enable (IFE):** When set, GHC.IE is set, and P0IS.IFS is set, the HBA shall generate an interrupt..
26	RW	0	**Interface Non-fatal Error Enable (INFE):** When set, GHC.IE is set, and P0IS.INFS is set, the HBA shall generate an interrupt.
25	*RO*	*0*	*Reserved*
24	RW	0	**Overflow Enable (OFE):** When set, and GHC.IE and P0IS.OFS are set, the HBA shall generate an interupt.
23	RW	0	**Incorrect Port Multiplier Enable (IPME):** When set, and GHC.IE and P0IS.IPMS are set, the HBA shall generate an interupt.
22	RW	0	**PhyRdy Change Interrupt Enable (PRCE):** When set to '1', and GHC.IE is set to '1', and P0IS.PRCS is set to '1', the HBA shall generate an interrupt.
21:08	*RO*	*0*	*Reserved*
07	RW/ RO	0	**Device Interlock Enable (DIE):** When set, and P0IS.DIS is set, the HBA shall generate an interrupt. For systems that do not support an interlock switch, this bit shall be a read-only '0'.
06	RW	0	**Port Change Interrupt Enable (PCE):** When set, GHC.IE is set, and P0IS.PCS is set, the HBA shall generate an interrupt.
05	RW	0	**Descriptor Processed Interrupt Enable (DPE):** When set, GHC.IE is set, and P0IS.DPS is set, the HBA shall generate an interrupt.
04	RW	0	**Unknown FIS Interrupt Enable (UFE):** When set, GHC.IE is set, and P0IS.UFS is set to '1', the HBA shall generate an interrupt.
03	RW	0	**Set Device Bits FIS Interrupt Enable (SDBE):** When set, GHC.IE is set, and P0IS.SDBS is set, the HBA shall generate an interrupt.
02	RW	0	**DMA Setup FIS Interrupt Enable (DSE):** When set, GHC.IE is set, and P0IS.DSS is set, the HBA shall generate an interrupt.
01	RW	0	**PIO Setup FIS Interrupt Enable (PSE):** When set, GHC.IE is set, and P0IS.PSS is set, the HBA shall generate an interrupt.
00	RW	0	**Device to Host Register FIS Interrupt Enable (DHRE):** When set, GHC.IE is set, and P0IS.DHRS is set, the HBA shall generate an interrupt.

3.3.7 Offset 118h: P0CMD – Port 0 Command

Bit	Type	Reset	Description
31:28	RW	0h	**Interface Communication Control (ICC):** This field is used to control power management states of the interface. If the Link layer is currently in the L_IDLE state, writes to this field shall cause the HBA to initiate a transition to the interface power management state requested. If the Link layer is not currently in the L_IDLE state, writes to this field shall have no effect. **Value** **Definition** *Fh - 7h* *Reserved* 6h **Slumber:** This shall cause the HBA to request a transition of the interface to the Slumber state. The SATA device may reject the request and the interface shall remain in its current state. *5h - 3h* *Reserved* 2h **Partial:** This shall cause the HBA to request a transition of the interface to the Partial state. The SATA device may reject the request and the interface shall remain in its current state. 1h **Active:** This shall cause the HBA to request a transition of the interface into the active state. 0h **No-Op / Idle:** When software reads this value, it indicates the HBA is ready to accept a new interface control command, although the transition to the previously selected state may not yet have occurred. When system software writes a non-reserved value other than No-Op (0h), the HBA shall perform the action and update this field back to Idle (0h). If software writes to this field to change the state to a state the link is already in (i.e. interface is in the active state and a request is made to go to the active state), the HBA shall take no action and return this field to Idle. If the interface is in a low power state and software wants to transition to a different low power state, software must first bring the link to active and then initiate the transition to the desired low power state.
27	RW/ RO	0	**Aggressive Slumber / Partial (ASP):** When set, and ALPE is set, the HBA shall aggressively enter the Slumber state when it clears the PxCI register and the PxSACT register is cleared or when it clears the PxSACT register and PxCI is cleared. When cleared, and ALPE is set, the HBA shall aggressively enter the Partial state when it clears the PxCI register and the PxSACT register is cleared or when it clears the PxSACT register and PxCI is cleared. See section 8.3.1.3 for details.
26	RW/ RO	0	**Aggressive Link Power Management Enable (ALPE):** When set, the HBA shall aggressively enter a lower link power state (Partial or Slumber) based upon the setting of the ASP bit. See section 8.3.1.3 for details.
25	RW	0	**Drive LED on ATAPI Enable (DLAE):** When set, the HBA shall drive the LED pin active for commands regardless of the state of P0CMD.ATAPI. When cleared, the HBA shall only drive the LED pin active for commands if P0CMD.ATAPI set to '0'. See section 10.10 for details on the activity LED.
24	RW	0	**Device is ATAPI (ATAPI):** When set, the connected device is an ATAPI device. This bit is used by the HBA to control whether or not to generate the desktop LED when commands are active. See section 10.10 for details on the activity LED.
23:21	*RO*	*0*	*Reserved*
20	RO	HwInit	**Cold Presence Detection (CPD):** If set to '1', the platform supports cold presence detection on this port. If cleared to '0', the platform does not support cold presence detection on this port. When this bit is set to '1', P0CMD.HPCP should also be set to '1'.
19	RO	HwInit	**Interlock Switch Attached to Port (ISP):** If set to '1', the platform supports an interlocked switch attached to this port. If cleared to '0', the platform does not support an interlocked switch attached to this port. When this bit is set to '1', P0CMD.HPCP should also be set to '1'.
18	RO	HwInit	**Hot Plug Capable Port (HPCP):** If set to '1', the platform has exposed this port to the user for hot plug insertion/removal. If cleared to '0', the platform has not exposed this port to the user for hot plug insertion/removal. If P0CPD or P0ISP is set to '1', then this bit should be set to '1'. If the device connected to this port is screwed into the system chassis such that it is not hot pluggable, this bit should be cleared to '0'.

Bit	Type	Reset	Description
17	RW/ RO	0	**Port Multiplier Attached (PMA):** This bit is read/write for HBAs that support a Port Multiplier (CAP.SPM = '1'). This bit is read-only for HBAs that do not support a port Multiplier (CAP.SPM = '0'). When set to '1' by software, a Port Multiplier is attached to the HBA for this port. When cleared to '0' by software, a Port Multiplier is not attached to the HBA for this port. Software is responsible for detecting whether a Port Multiplier is present; hardware does not auto-detect the presence of a Port Multiplier.
16	RO	See Desc	**Cold Presence State (CPS):** The CPS bit reports whether a device is currently detected on this port. If CPS is set to '1', then the HBA detects via cold presence that a device is attached to this port. If CPS is cleared to '0' , then the HBA detects via cold presence that there is no device attached to this port.
15	RO	0	**Command List Running (CR):** When this bit is set, the command list DMA engine for the port is running. See the AHCI state machine in section 5.2.2 for details on when this bit is set and cleared by the HBA.
14	RO	0	**FIS Receive Running (FR):** When set, the FIS Receive DMA engine for the port is running. See section 10.3.2 for details on when this bit is set and cleared by the HBA.
13	RO	See Desc	**Interlock Switch State (ISS):** The ISS bit reports the state of an interlocked switch attached to this port. If CAP.SIS is set to '1' and the interlocked switch is closed then this bit is cleared to '0'. If CAP.SIS is set to '1' and the interlocked switch is open then this bit is set to '1'. If CAP.SIS is set to '0' then this bit is cleared to '0'. Software should only use this bit if both CAP.SIS and P0CMD.ISP are set to '1'.
12:08	RO	0h	**Current Command Slot (CCS):** This field is valid when P0CMD.ST is set to '1' and shall be set to the command slot value of the command that is currently being issued by the HBA. When P0CMD.ST transitions from '1' to '0', this field shall be reset to '0'. After P0CMD.ST transitions from '0' to '1', the highest priority slot to issue from next is command slot 0. After the first command has been issued, the highest priority slot to issue from next is P0CMD.CCS + 1. For example, after the HBA has issued its first command, if CCS = 0h and P0CI is set to 3h, the next command that will be issued is from command slot 1.
07:05	*RO*	*0*	*Reserved*
04	RW	0	**FIS Receive Enable (FRE):** When set, the HBA may post received FISes into the FIS receive area pointed to by PxFB (and for 64-bit HBAs, PxFBU). When cleared, received FISes are not accepted by the HBA, except for the first D2H register FIS after the initialization sequence, and no FISes are posted to the FIS receive area. System software must not set this bit until PxFB (PxFBU) have been programmed with a valid pointer to the FIS receive area, and if software wishes to move the base, this bit must first be cleared, and software must wait for the FR bit in this register to be cleared. Refer to section 10.3.2 for important restrictions on when FRE can be set and cleared.
03	RW	0	**Command List Override (CLO):** Setting this bit to '1' causes PxTFD.STS.BSY and PxTFD.STS.DRQ to be cleared to '0'. This allows a software reset to be transmitted to the device regardless of whether the BSY and DRQ bits are still set in the PxTFD.STS register. The HBA sets this bit to '0' when PxTFD.STS.BSY and PxTFD.STS.DRQ have been cleared to '0'. A write to this register with a value of '0' shall have no effect. This bit shall only be set to '1' immediately prior to setting the PxCMD.ST bit to '1' from a previous value of '0'. Setting this bit to '1' at any other time is not supported and will result in indeterminate behavior.
02	RW/ RO	0/1	**Power On Device (POD):** This bit is read/write for HBAs that support cold presence detection on this port as indicated by PxCMD.CPD set to '1'. This bit is read only '1' for HBAs that do not support cold presence detect. When set, the HBA sets the state of a pin on the HBA to '1' so that it may be used to provide power to a cold-presence detectable port.
01	RW/ RO	0/1	**Spin-Up Device (SUD):** This bit is read/write for HBAs that support staggered spin-up via CAP.SSS. This bit is read only '1' for HBAs that do not support staggered spin-up. On an edge detect from '0' to '1', the HBA shall start a COMRESET initializatoin sequence to the device. Clearing this bit causes no action on the interface.

Bit	Type	Reset	Description
00	RW	0	**Start (ST):** When set, the HBA may process the command list. When cleared, the HBA may not process the command list. Whenever this bit is changed from a '0' to a '1', the HBA starts processing the command list at entry '0'. Whenever this bit is changed from a '1' to a '0', the PxCI register is cleared by the HBA upon the HBA putting the controller into an idle state. Refer to section 10.3.1 for important restrictions on when ST can be set to '1'.

3.3.8 Offset 120h: P0TFD – Port 0 Task File Data

This is a 32-bit register that copies specific fields of the task file when FISes are received. The FISes that contain this information are:

- D2H Register FIS
- PIO Setup FIS
- Set Device Bits FIS (BSY and DRQ are not updated with this FIS)

Bit	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:08	RO	0	**Error (ERR):** Contains the latest copy of the task file error register.
07:00	RO	7Fh	**Status (STS):** Contains the latest copy of the task file status register. Fields of note in this register that affect AHCI: Bit / Field / Definition 7 / BSY / Indicates the interface is busy 6:4 / n/a / Not applicable 3 / DRQ / Indicates a data transfer is requested 2:1 / n/a / Not applicable 0 / ERR / Indicates an error during the transfer. The HBA shall update the entire 8-bit field, not just the bits noted above.

3.3.9 Offset 124h: P0SIG – Port 0 Signature

This is a 32-bit register which contains the initial signature of an attached device when the first D2H Register FIS is received from that device. It is updated once after a reset sequence.

Bit	Type	Reset	Description
31:00	RO	FFFFFFFFh	**Signature (SIG):** Contains the signature received from a device on the first D2H Register FIS. The bit order is as follows: Bit / Field 31:24 / LBA High Register 23:16 / LBA Mid Register 15:08 / LBA Low Register 07:00 / Sector Count Register

3.3.10 Offset 128h: P0SSTS – Port 0 Serial ATA Status (SCR0: SStatus)

This is a 32-bit register that conveys the current state of the interface and host. The HBA updates it continuously and asynchronously. When the HBA transmits a COMRESET to the device, this register is updated to its reset values.

Bit	Type	Reset	Description
31:12	*RO*	*0*	*Reserved*
11:08	RO	0	**Interface Power Management (IPM):** Indicates the current interface state: 0h Device not present or communication not established 1h Interface in active state 2h Interface in Partial power management state 6h Interface in Slumber power management state All other values reserved

07:04	RO	0	**Current Interface Speed (SPD):** Indicates the negotiated interface communication speed. 0h Device not present or communication not established 1h Generation 1 communication rate negotiated 2h Generation 2 communication rate negotiated All other values reserved
03:00	RO	0	**Device Detection (DET):** Indicates the interface device detection and Phy state. 0h No device detected and Phy communication not established 1h Device presence detected but Phy communication not established 3h Device presence detected and Phy communication established 4h Phy in offline mode as a result of the interface being disabled or running in a BIST loopback mode All other values reserved

3.3.11 Offset 12Ch: P0SCTL – Port 0 Serial ATA Control (SCR2: SControl)

This is a 32-bit read-write register by which software controls SATA capabilities. Writes to this register result in an action being taken by the host adapter or interface. Reads from the register return the last value written to it.

Bit	Type	Reset	Description
31:20	*RO*	*0*	*Reserved*
19:16	*RO*	*0h*	*Port Multiplier Port (PMP): This field is not used by AHCI.*
15:12	*RO*	*0h*	*Select Power Management (SPM): This field is not used by AHCI*
11:08	RW	0h	**Interface Power Management Transitions Allowed (IPM):** Indicates which power states the HBA is allowed to transition to. If an interface power management state is disabled, the HBA is not allowed to initiate that state and the HBA must $PMNAK_P$ any request from the device to enter that state. 0h No interface restrictions 1h Transitions to the Partial state disabled 2h Transitions to the Slumber state disabled 3h Transitions to both Partial and Slumber states disabled All other values reserved
07:04	RW	0h	**Speed Allowed (SPD):** Indicates the highest allowable speed of the interface. 0h No speed negotiation restrictions 1h Limit speed negotiation to Generation 1 communication rate 2h Limit speed negotiation to a rate not greater than Generation 2 communication rate All other values reserved
03:00	RW	0h	**Device Detection Initialization (DET):** Controls the HBA's device detection and interface initialization. 0h No device detection or initialization action requested 1h Perform interface communication initialization sequence to establish communication. This is functionally equivalent to a hard reset and results in the interface being reset and communications reinitialized. While this field is 1h, COMRESET is continuously transmitted on the interface. Software should leave the DET field set to 1h for a minimum of 1 millisecond to ensure that a COMRESET is sent on the interface. 4h Disable the Serial ATA interface and put Phy in offline mode. All other values reserved This field may only be modified when P0CMD.ST is '0'. Changing this field while the P0CMD.ST bit is set to '1' results in undefined behavior. When P0CMD.ST is set to '1', this field should have a value of 0h.

3.3.12 Offset 130h: P0SERR – Port 0 Serial ATA Error (SCR1: SError)

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:16</td><td>RWC</td><td>0000h</td><td>Diagnostics (DIAG) - Contains diagnostic error information for use by diagnostic software in validating correct operation or isolating failure modes:

<table>
<tr><td>31:27</td><td>Reserved</td></tr>
<tr><td>26</td><td>Exchanged (X): When set to one this bit indicates a COMINIT signal was received. This bit is reflected in the P0IS.PCS bit.</td></tr>
<tr><td>25</td><td>Unknown FIS Type (F): Indicates that one or more FISs were received by the Transport layer with good CRC, but had a type field that was not recognized/known.</td></tr>
<tr><td>24</td><td>Transport state transition error (T): Indicates that an error has occurred in the transition from one state to another within the Transport layer since the last time this bit was cleared.</td></tr>
<tr><td>23</td><td>Link Sequence Error (S): Indicates that one or more Link state machine error conditions was encountered. The Link Layer state machine defines the conditions under which the link layer detects an erroneous transition.</td></tr>
<tr><td>22</td><td>Handshake Error (H): Indicates that one or more R_ERR handshake response was received in response to frame transmission. Such errors may be the result of a CRC error detected by the recipient, a disparity or 8b/10b decoding error, or other error condition leading to a negative handshake on a transmitted frame.</td></tr>
<tr><td>21</td><td>CRC Error (C): Indicates that one or more CRC errors occurred with the Link Layer.</td></tr>
<tr><td>20</td><td>Disparity Error (D): This field is not used by AHCI.</td></tr>
<tr><td>19</td><td>10B to 8B Decode Error (B): Indicates that one or more 10B to 8B decoding errors occurred.</td></tr>
<tr><td>18</td><td>Comm Wake (W): Indicates that a Comm Wake signal was detected by the Phy.</td></tr>
<tr><td>17</td><td>Phy Internal Error (I): Indicates that the Phy detected some internal error.</td></tr>
<tr><td>16</td><td>PhyRdy Change (N): Indicates that the PhyRdy signal changed state. This bit is reflected in the P0IS.PRCS bit.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>15:00</td><td>RWC</td><td>0000h</td><td>Error (ERR): The ERR field contains error information for use by host software in determining the appropriate response to the error condition.

If one or more of bits 11:8 of this register are set, the controller shall stop the current transfer.

<table>
<tr><td>15:12</td><td>Reserved</td></tr>
<tr><td>11</td><td>Internal Error (E): The SATA controller failed due to a master or target abort when attempting to access system memory.</td></tr>
<tr><td>10</td><td>Protocol Error (P): A violation of the Serial ATA protocol was detected.</td></tr>
<tr><td>9</td><td>Persistent Communication or Data Integrity Error (C): A communication error that was not recovered occurred that is expected to be persistent. Persistent communications errors may arise from faulty interconnect with the device, from a device that has been removed or has failed, or a number of other causes.</td></tr>
<tr><td>8</td><td>Transient Data Integrity Error (T): A data integrity error occurred that was not recovered by the interface.</td></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td>Recovered Communications Error (M): Communications between the device and host was temporarily lost but was re-established. This can arise from a device temporarily being removed, from a temporary loss of Phy synchronization, or from other causes and may be derived from the PhyNRdy signal between the Phy and Link layers.</td></tr>
<tr><td>0</td><td>Recovered Data Integrity Error (I): A data integrity error occurred that was recovered by the interface through a retry operation or other recovery action.</td></tr>
</table>
</td></tr>
</table>

3.3.13 Offset 134h: P0SACT – Port 0 Serial ATA Active (SCR3: SActive)

Bit	Type	Reset	Description
31:0	R/W1	0	**Device Status (DS):** System software sets this bit for SATA queuing operations prior to setting the PxCI.CI bit in the same command slot entry. This field is cleared via the Set Device Bits FIS. This field is also cleared when PxCMD.ST is written from a ‘1’ to a ‘0’ by software. This field is not cleared by a COMRESET or a software reset.

3.3.14 Offset 138h: P0CI – Port 0 Command Issue

Bit	Type	Reset	Description
31:0	R/W1	0	**Commands Issued (CI):** This field is set by software to indicate to the HBA that a command has been built in system memory for a command slot and may be sent to the device. When the HBA receives a FIS which clears the BSY, DRQ, and ERR bits for the command, it clears the corresponding bit in this register for that command slot. This field is also cleared when PxCMD.ST is written from a ‘1’ to a ‘0’ by software.

3.3.15 Offset 170h to 17Fh: P0VS – Vendor Specific

The registers at offset 170h to 17Fh are vendor specific.

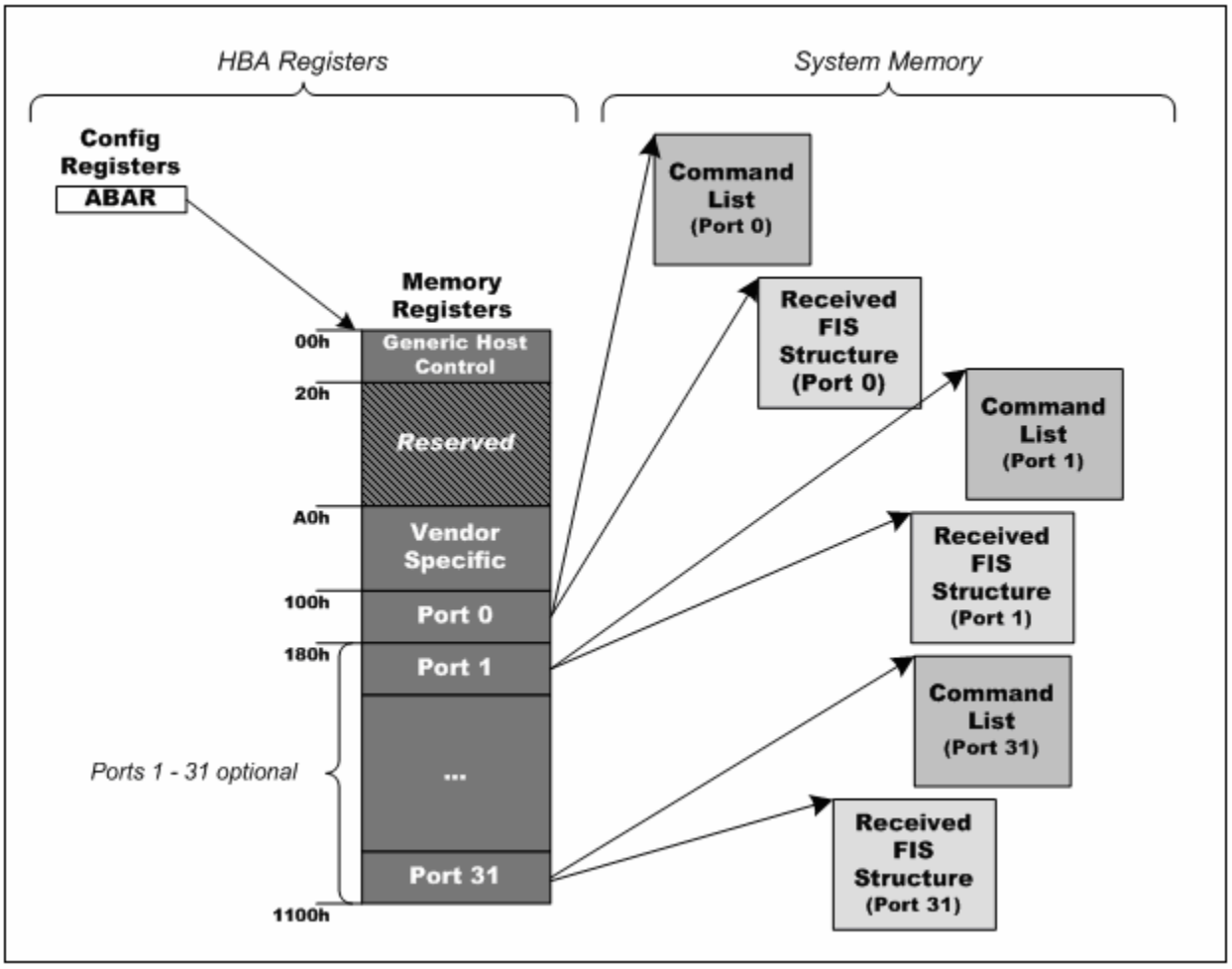
4 System Memory Structures

4.1 HBA Memory Space Usage

Most communication between software and an SATA device is through the HBA via system memory descriptors, which indicates the status of all received and sent FISes, as well as pointers for data transfers. Some additional communication is done via registers in the HBA, for each port and for global control.

A visual breakdown of the HBA memory space is shown in Figure 4.

Figure 4: HBA Memory Space Usage

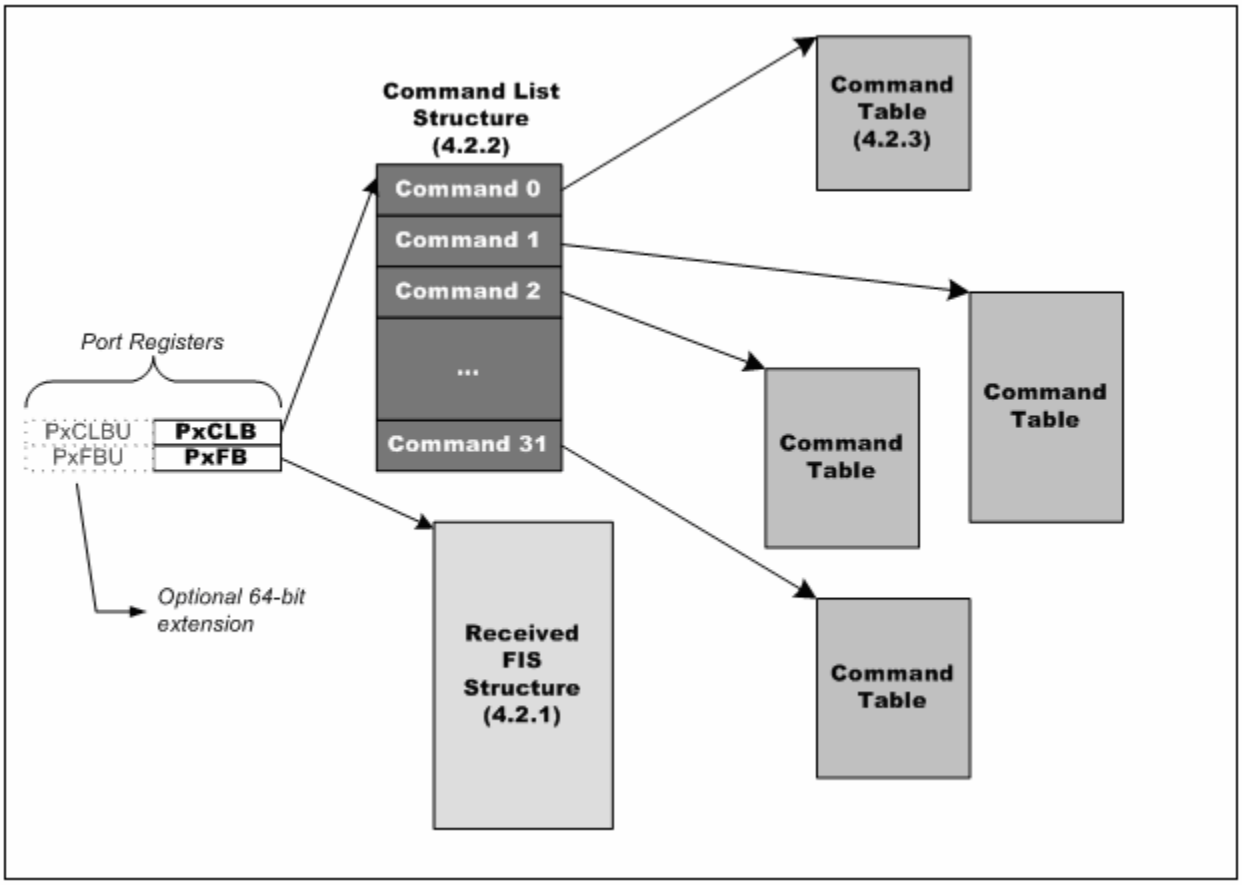


4.2 Port Memory Usage

There are two descriptors per port that are used to convey information. One is the FIS descriptor, which contains FISes received from a device, and the other is the Command List, which contains a list of 1 to 32 commands available for a port to execute.

The base for each pointer is a 64-bit value (32-bits for HBAs that do not support 64-bit addressing). An overview of the overall structure shown in Figure 5, and the following sections describe each area.

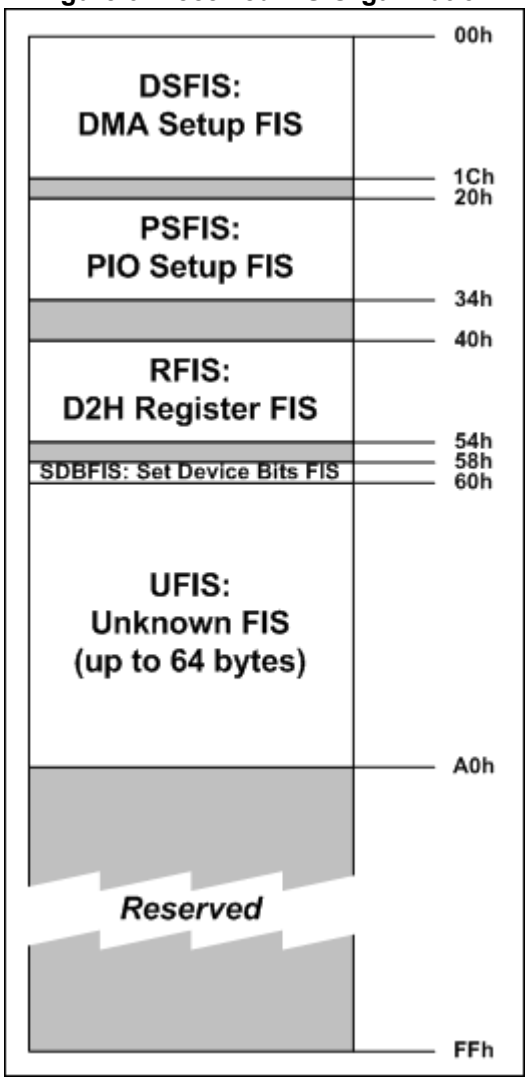
Figure 5: Port System Memory Structures



4.2.1 Received FIS Structure

The HBA uses an area of system memory to communicate information on received FISes. This structure is pointed to by PxFBU and PxFB. The structure is shown in Figure 6.

Figure 6: Received FIS Organization



When a DMA setup FIS arrives from the device, the HBA copies it to the DSFIS area of this structure.

When a PIO setup FIS arrives from the device, the HBA copies it to the PSFIS area of this structure.

When a D2H Register FIS arrives from the device, the HBA copies it to the RFIS area of this structure.

When a Set Device Bits FIS arrives from the device, the HBA copies it to the SDBFIS area of this structure.

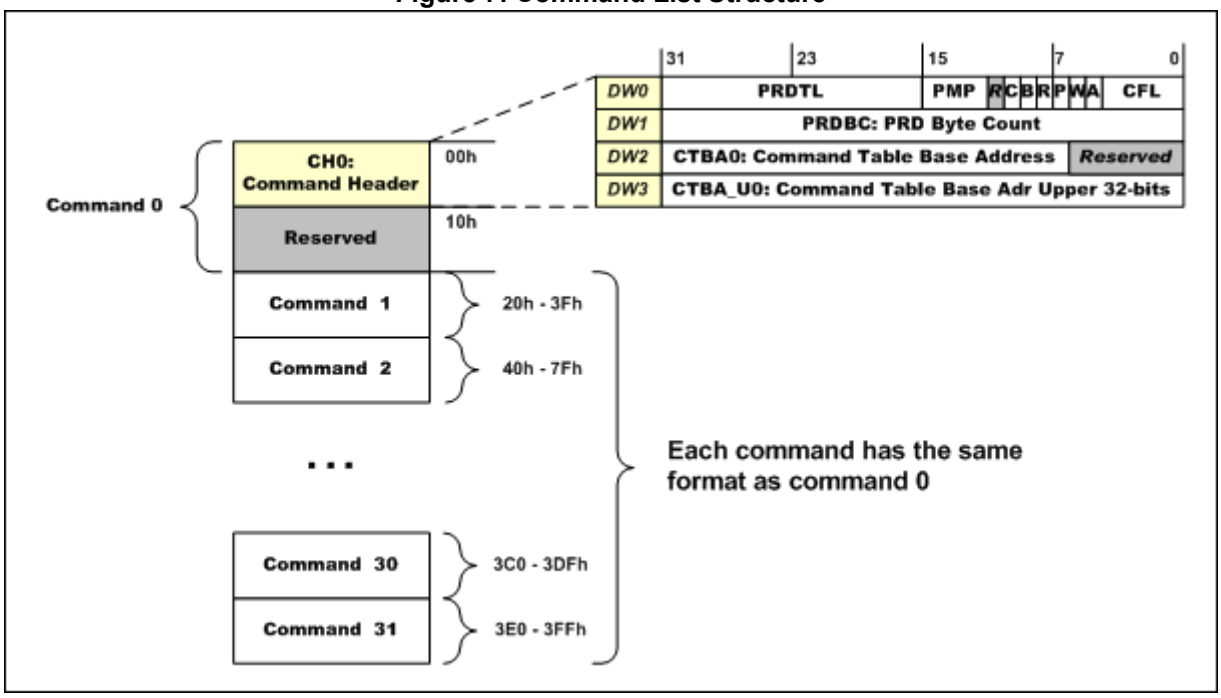
When an unknown FIS arrives from the device, the HBA copies it to the UFIS area in this structure, and sets PxSERR.DIAG.F, which is reflected in PxIS.UFS when the FIS is posted to memory. A maximum of 64-bytes of an unknown FIS type may be sent to an HBA. If an unknown FIS arrives that is longer than 64-bytes, the FIS is considered illegal and is handled as described in section 6.1.2. While the length of the FIS is unknown to the HBA, it is expected to be known by system software, and therefore only the valid bytes shall be processed by software. The HBA is not required to tolerate receiving an unknown FIS when the HBA is expecting a Data FIS from the device or when the HBA is about to transfer a Data FIS to the device based on the command protocol being used.

The HBA shall take the actions described in 5.2 when a FIS is received, which includes updating the Received FIS structure as previously outlined, generating interrupts as appropriate, etc.

4.2.2 Command List Structure

Figure 7 shows the command list structure. Each entry contains a command header, which is a 16-byte structure that details the direction, type, and scatter/gather pointer of the command. Further details of each field are listed below.

Figure 7: Command List Structure



The fields inside the command header are:

Figure 8: DW 0 – Description Information

Bit	Description
31:16	**Physical Region Descriptor Table Length (PRDTL):** Length of the scatter/gather descriptor table in entries, called the Physical Region Descriptor Table. Each entry is 4 DW. A '0' represents 0 entries, FFFFh represents 65,535 entries. The HBA uses this field to know when to stop fetching PRDs. If this field is '0', then no data transfer shall occur with the command.
15:12	**Port Multiplier Port (PMP):** Indicates the port number that should be used when constructing data FISes on transmit, and to check against all FISes received for this command. This value must be set to 0h if a Port Multiplier is not attached (PxCMD.PMA is '0').
11	*Reserved*
10	**Clear Busy upon R_OK (C):** When set, the HBA shall clear PxTFD.STS.BSY and PxCI.CI(z) after transmitting this FIS and receiving R_OK. When cleared, the HBA shall not clear PxTFD.STS.BSY nor PxCI.CI(z) after transmitting this FIS and receiving R_OK.
09	**BIST (B):** When '1', indicates that the command that software built is for sending a BIST FIS. The HBA shall send the FIS and enter a test mode. The tests that can be run in this mode are outside the scope of this specification.
08	**Reset (R):** When '1', indicates that the command that software built is for a device reset. The HBA must perform a SYNC escape (if necessary) to get the device into an idle state before sending the command. See section 10.4 for details on reset.
07	**Prefetchable (P):** This bit is only valid if the PRDTL field is non-zero. When set and PRDTL is non-zero, the HBA may prefetch PRDs in anticipation of performing a data transfer. System software should not set this bit when using native command queuing commands. **Note:** The HBA may prefetch the ATAPI command, PRD entries, and data regardless of the state of this bit. However, it is recommended that the HBA use this information from software to avoid prefetching needlessly.
06	**Write (W):** When set, indicates that the direction is a device write (data from system memory to device). When cleared, indicates that the direction is a device read (data from device to system memory). If this bit is set and the P bit is set, the HBA may prefetch data in anticipation of receiving a DMA Setup FIS, a DMA Activate FIS, or PIO Setup FIS, in addition to prefetching PRDs.
05	**ATAPI (A):** When '1', indicates that a PIO setup FIS shall be sent by the device indicating a transfer for the ATAPI command. The HBA may prefetch data from CTBAz[ACMD] in anticipation of receiving the PIO Setup FIS.
04:00	**Command FIS Length (CFL):** Length of the Command FIS. A '0' represents 0 DW, '4' represents 4 DW. A length of '0' or '1' is illegal. The maximum value allowed is 10h, or 16 DW. The HBA uses this field to know the length of the FIS it shall send to the device.

Figure 9: DW 1 - Command Status

Bit	Description
31:00	**Physical Region Descriptor Byte Count (PRDBC):** Indicates the current byte count that has transferred on device writes (system memory to device) or device reads (device to system memory). For rules on when this field is updated, refer to section 5.3.1

Figure 10: DW 2 – Command Table Base Address

Bit	Description
31:07	**Command Table Descriptor Base Address (CTBA):** Indicates the 32-bit physical address of the command table, which contains the command FIS, ATAPI Command, and PRD table. This address must be aligned to a 128-byte cache line, indicated by bits 06:00 being reserved.
06:00	*Reserved*

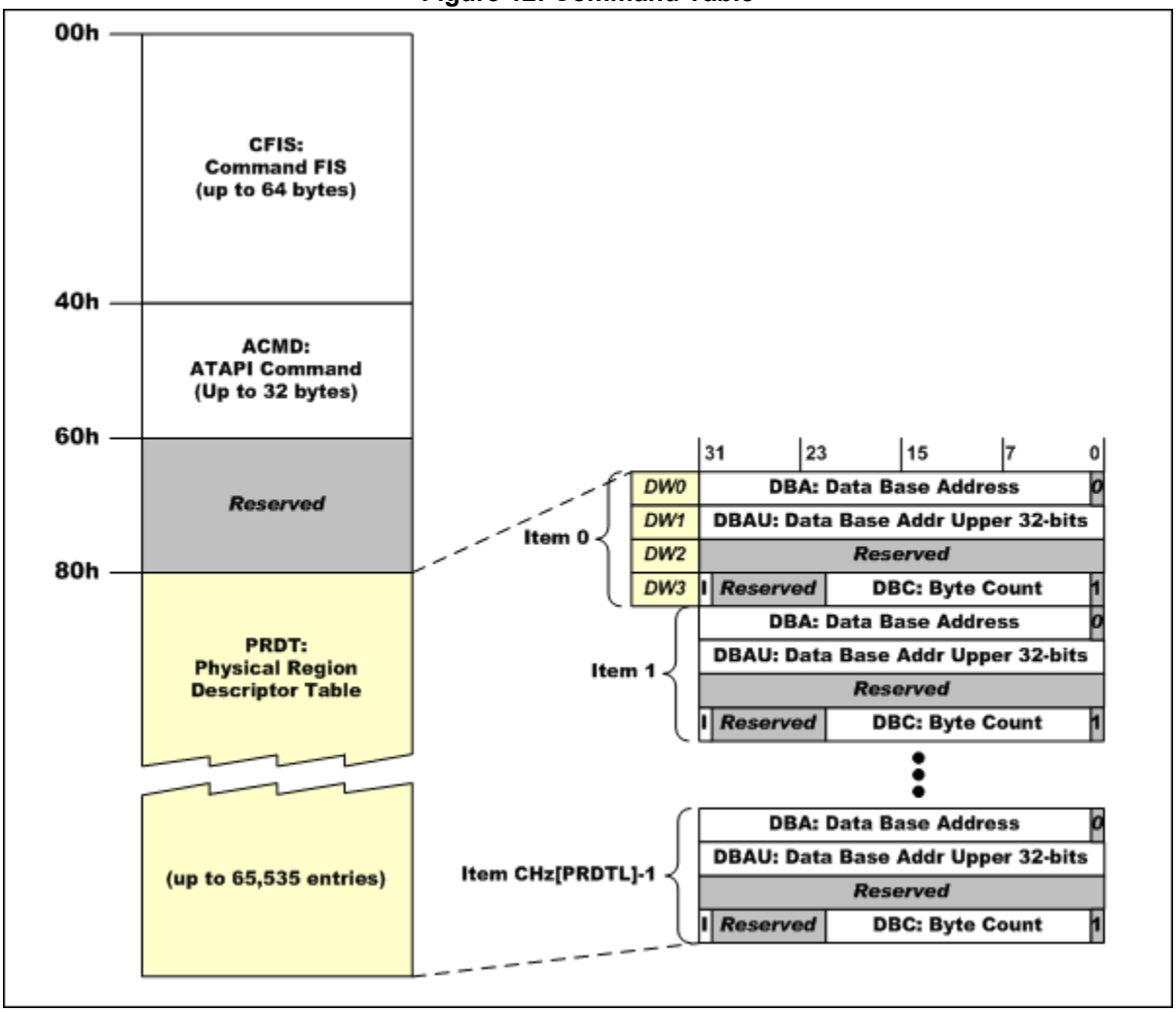
Figure 11: DW 3 – Command Table Base Address Upper

Bit	Description
31:00	**Command Table Descriptor Base Address Upper 32-bits (CTBAU):** This is the upper 32-bits of the Command Table Base. It is only valid if the HBA indicated that it can support 64-bit addressing through the S64A bit in the capabilities register, and is ignored otherwise.

4.2.3 Command Table

Each entry in the command list points to a structure called the command table.

Figure 12: Command Table



Each command contains several fields. The fields break down as follows:

4.2.3.1 Command FIS (CFIS)

This is a software constructed FIS. For data transfer operations, this is the H2D Register FIS format as specified in the Serial ATA 1.0a specification. The HBA sets PxTFD.STS.BSY, and then sends this structure to the attached port. If a Port Multiplier is attached, this field must have the Port Multiplier port number in the FIS itself – it shall not be added by the HBA. Valid CFIS lengths are 2 to 16 Dwords and must be in Dword granularity.

4.2.3.2 ATAPI Command (ACMD)

This is a software constructed region of 12 or 16 bytes in length that contains the ATAPI command to transmit if the “A” bit is set in the command header. The ATAPI command must be either 12 or 16 bytes in length. The length transmitted by the HBA is determined by the PIO setup FIS that is sent by the device requesting the ATAPI command.

4.2.3.3 Physical Region Descriptor Table (PRDT)

This table contains the scatter / gather list for the data transfer. It contains a list of 0 (no data to transfer) to up to 65,535 entries. A breakdown of each field in a PRD table is shown below. Item 0 refers to the first entry in the PRD table. Item "CHz[PRDTL] – 1" refers to the last entry in the table, where the length field comes from the PRDTL field in the command list entry for this command slot.

Figure 13: DW 0 – Data Base Address

Bit	Description
31:01	**Data Base Address (DBA):** Indicates the 32-bit physical address of the data block. The block must be word aligned, indicated by bit 0 being reserved.
00	*Reserved*

Figure 14: DW 1 – Data Base Address Upper

Bit	Description
31:00	**Data Base Address Upper 32-bits (DBAU):** This is the upper 32-bits of the data block physical address. It is only valid if the HBA indicated that it can support 64-bit addressing through the S64A bit in the capabilities register, and is ignored otherwise.

Figure 15: DW 2 – Reserved

Bit	Description
31:00	*Reserved*

Figure 16: DW 3 – Description Information

Bit	Description
31	**Interrupt on Completion (I):** When set, indicates that hardware should assert an interrupt when the data block for this entry has transferred, which means that no data is in the HBA hardware. Data may still be in flight to system memory (disk reads), or at the device (an R_OK or R_ERR has yet to be received). The HBA shall set the PxIS.DPS bit after completing the data transfer, and if enabled, generate an interrupt.
30:22	*Reserved*
21:00	**Data Byte Count (DBC):** A '0' based value that Indicates the length, in bytes, of the data block. A maximum of length of 4MB may exist for any entry. Bit '0' of this field must always be '1' to indicate an even byte count. A value of '1' indicates 2 bytes, '3' indicates 4 bytes, etc.

5 Data Transfer Operation

5.1 Introduction

The data structures of AHCI are built assuming that each port in an HBA contains its own DMA engine, that is, each port can be executed independently of any other port. This is a natural flow for software, since each device is generally treated as a separate execution thread. It is strongly recommended that HBA implementations proceed in this fashion.

Software presents a list of commands to the HBA for a port, which then processes them. For HBAs that have a command list depth of '1', this is a single step operation, and software only presents a single command. For HBAs that support a command list, multiple commands may be posted.

Software posts new commands received by the OS to empty slots in the list, and sets the corresponding slot bit in the PxCI register. The HBA continuously looks at PxCI to determine if there are commands to transmit to the device.

The HBA processes the command list in a linear fashion, as described by the HBA state machine below.

5.2 HBA State Machine (Normative)

The state machine arcs are in priority order and are not required to be mutually exclusive. For example, if both the first arc and second arc of a state are true, the first arc is always taken. Subsequent arcs are not evaluated until the previous arc is determined to be false.

The state machine does not comprehend multiple MSI message interrupts. The proper interrupt behavior for multiple MSI is defined in section 10.6.2.2. A future AHCI revision will incorporate multiple MSI behavior into the state machine.

5.2.1 Variables

hbaIssueTag The hbaIssueTag variable contains the tag of the last command issued. On power-up or reset of the HBA port, hbaIssueTag is cleared to 0.

hbaDataTag The hbaDataTag variable contains the tag of the command to transfer data for next. On power-up or reset of the HBA port, hbaDataTag is cleared to 0.

hbaPMP The hbaPMP variable contains the value in the PMP field of the command table of the last command FIS transferred to the device. On power-up or reset of the HBA port, hbaPMP is cleared to 0.

hbaXferAtapi The hbaXferAtapi variable is set to 1 when a command is issued that had the A bit set for a particular transfer. The hbaXferAtapi variable is cleared to 0 when a Data FIS is transferred to the device that contains the ATAPI command from the command list. On power-up or reset of the HBA port, hbaXferAtapi is cleared to 0.

hbaPioXfer The hbaPioXfer variable is set to 1 when a PIO Setup FIS is received. This variable is used after a data transfer occurs in order to update the Status register appropriately.

hbaPioESts The hbaPioESts variable is set to the E_Status field of the PIO Setup FIS to be stored until the data for the DRQ block is transferred. On power-up or reset of the HBA port, hbaPioESts is cleared to 0.

hbaPioErr The hbaPioErr variable is set to the Error field of the PIO Setup FIS to be stored until the data for the DRQ block is transferred. On power-up or reset of the HBA port, hbaPioErr is cleared to 0.

hbaPioIbit The hbaPioIbit variable is set to the I bit of the PIO Setup FIS to be stored until the data for the DRQ block is transferred. On power-up or reset of the HBA port, hbaPioIbit is cleared to 0.

hbaDmaXferCnt The hbaDmaXferCnt variable is set to the DMA transfer count for a particular DMA transfer. The DMA transfer may consist of multiple Data FISes. The hbaDmaXferCnt variable is decremented by the size of a Data FIS on each successful reception of a Data FIS. On power-up or reset of the HBA port, hbaDmaXferCnt is cleared to 0. An hbaDmaXferCnt = '0' signals that there was no DMA Setup FIS or PIO Setup FIS corresponding to the data transfer and that the transfer lengths should be constructed based on the PRD table entries only.

hbaFatal The hbaFatal variable is set whenever the HBA has detected a fatal error. When this variable is set, no new commands shall be issued. On power-up or reset of the HBA port, hbaFatal is cleared to 0.

hbaCmdToIssue This variable is set whenever the currently fetched command still needs to be transmitted to the device. It is used by the state machine to ensure the command is actually transmitted to the device, especially after a command transmission failure. On power-up or reset of the HBA port, hbaCmdToIssue is cleared to 0.

hbaPrdIntr This variable is set whenever the HBA completes a PRD in either the data transmission or data reception states. It is used to generate a PRD interrupt at the end of a successful data FIS. On power-up or reset of the HBA port, hbaPrdIntr is clerad to 0.

hbaUpdateSig This variable is set whenever the HBA needs to update the PxSIG register due to a hard or software reset. It is cleared when the D2H Register FIS which updates the signature is received. On power-up or reset of the HBA port, hbaUpdateSig is set to 1.

hbaSActive This variable is set to the value of the SActive field in a received Set Device Bits FIS. On power-up or reset of the HBA port, hbaSActive is cleared to 0.

5.2.2 HBA Idle States

5.2.2.1 H:Init

H:Init	The HBA performs the following actions: 1. Resets state machine variables to their reset values, as specified in section 5.2.1 2. Resets GHC.AE, GHC.IE, and the IS register to their reset values. 3. Resets all port specific register fields except those fields marked as HwInit and the PxFB/PxFBU/PxCLB/PxCLBU registers.

1.	GHC.HR is set to '1'	→	H:Init
2.	Else	→	H:NotRunning

NOTE:
This state is entered asynchronously when GHC.HR is set to '1' from a previous value of '0'.

The H:Init state is entered to initialize or reset a port. This state is only entered when the HBA powers up or an entire HBA reset is performed. If cold presence detect is supported as specified in PxCMD.CPD, the power to each port is off by default and remains off in this state until software enables power to the port.

5.2.2.2 H:NotRunning

H:NotRunning HBA sets hbaIssueTag = 32. HBA sets z = CAP.NCS.

1.	PxCMD.POD written to '1' from a '0'	→	H:PowerOn
2.	PxCMD.POD written to '0' from a '1'	→	H:PowerOff
3.	PxSCTL.DET written to '4h' from any other value	→	H:Offline
4.	PxSCTL.DET written to 1h from any other value and PxCMD.SUD = '1'	→	H:StartComm
5.	PxCMD.SUD written to '1' from '0' and PxSCTL.DET = '0h'	→	H:StartComm
6.	PxCMD.SUD written to '0' from '1' and PxSCTL.DET = '0h'	→	H:PhyListening
7.	PxCMD.ST = '1'	→	H:Idle
8.	D2H Register FIS received	→	NDR:Entry
9.	PxCMD.FRE written to '1' from a '0' and Register FIS is in receive FIFO and PxSERR.DIAG.X = '0'	→	H:RegFisPostToMem
10.	Else	→	H:NotRunning

The H:NotRunning state is entered when the port specific DMA engines are not running. The DMA engines may not be running for numerous reasons, including because the PxCMD.ST is cleared to '0' or the communication link with the device has not been established.

Software is only allowed to program the PxCMD.FRE, PxCMD.POD, PxSCTL.DET, and PxCMD.SUD register bits when PxCMD.ST is set to '0', If software programs these bits in any other state (when PxCMD.ST is set to '1'), indeterminate results may occur.

5.2.2.3 H:RegFisUpdate

H:RegFisUpdate	HBA performs the following actions: 1. HBA accepts the FIS with a status of R_OK. 2. HBA updates PxSIG with appropriate fields from D2H Register FIS as outlined in 3.3.9, and clears hbaUpdateSig to '0'.		
	1. PxCMD.FRE = '1'	→	H:RegFisPostToMem
	2. Else	→	H:NotRunning

The H:RegFisUpdate state is entered to update the PxSIG registers when a D2H Register FIS is received while the PxCMD.ST bit is cleared to '0'.

5.2.2.4 H:RegFisPostToMem

H:RegFisUpdate	HBA performs the following actions: 1. HBA posts the Register FIS in the receive FIFO to PxFB[RFIS] 2. Updates PxTFD.ERR register with value in the Error field of the Register FIS. 3. Updates PxTFD.STS register with value in the Status field of the Register FIS.		
	1. Unconditional	→	H:NotRunning

The H:RegFisUpdate state is entered to post the initial Register FIS and update the PxTFD register once PxCMD.FRE is set to '1'.

5.2.2.5 H:Offline

H:Offline	HBA puts Phy into offline mode and the Phy voltage level is brought to common-mode.		
	1. Unconditional	→	H:NotRunning

The H:Offline state is entered to place the Phy for the port into offline mode.

5.2.2.6 H:StartBitCleared

H:StartBitCleared	HBA clears PxCI to 0h, PxSACT to 0h, PxCMD.CCS to 0h, and PxCMD.CR to '0'.		
	1. DMA engines for the port not idle	→	H:StartBitCleared
	2. Unconditional	→	H:NotRunning
	NOTE: This state is entered asynchronously when software writes the PxCMD.ST bit to a '0' from a previous value of '1' and the HBA is not in state H:Init.		

The H:StartBitCleared state is entered when the PxCMD.ST bit is cleared to '0' from a previous value of '1'.

5.2.2.7 H:Idle

H:Idle	HBA sets PxCMD.CR to '1'.		
	1. PxSSTS.DET != 3h	→	H:NotRunning
	2. PxCI != 0h and hbaIssueTag = 32	→	H:SelectCmd
	3. PxCI.CI(hbaIssueTag) = '1' and hbaCmdToIssue = '1' and CTBA(hbaIssueTag)[R] is set to '1' and hbaDmaXferCnt = '0' and PxTFD.STS.BSY = '0' and PxTFD.STS.DRQ = '0'	→	CFIS:SyncEscape
	4. Data FIS received	→	DR:Entry
	5. Non-Data FIS received	→	NDR:Entry
	6. PxCI.CI(hbaIssueTag) = '1' and hbaCmdToIssue = '1' and hbaDmaXferCnt = '0' and PxTFD.STS.BSY = '0' and PxTFD.STS.DRQ = '0'	→	CFIS:Xmit

7.	Else	→	H:Idle

The H:Idle state is entered when in normal operation to determine the next thing to do.

5.2.2.8 H:SelectCmd

H:SelectCmd	HBA sets z = (z + 1) mod (CAP.NCS + 1).		
1.	PxCI.CI(z) = '0'	→	H:SelectCmd
2.	Else (command is selected)	→	H:FetchCmd

The H:SelectCmd state is entered to select the next command to issue to the device.

Note: The search for a new command to issue is described as an iterative process, where each clock checks a particular CI value. It is shown this way to explicitly indicate the next command to be sent. An implementation may optimize the search in a single clock, so long as the correct command is selected.

5.2.2.9 H:FetchCmd

H:FetchCmd	The HBA performs the following actions in the order specified: 1. HBA fetches command header from PxCLB[CHz] 2. HBA sets hbaIssueTag = z 3. HBA sets PxCMD.CCS = z 4. HBA sets hbaCmdToIssue = '1'		
1.	CTBA(hbaIssueTag).A (ATAPI) = '1' and CTBA(hbaIssueTag).P[1] (Prefetchable) = '1'	→	CFIS:PrefetchACMD
2.	CTBA(hbaIssueTag).P[1] (Prefetchable) = '1'	→	CFIS:PrefetchPRD
3.	Unconditional	→	H:Idle

NOTE:
1. The P bit may be ignored by hardware although it is recommended to be observed to avoid prefetching unnecessary data.
2. An HBA implementation may choose to ignore arcs 1 and 2 if it is not desireable to prefetch after the command FIS is fetched from memory.

The H:FetchCmd state is entered to fetch the command header for the next command to issue from memory. Note that when a command is selected to be transferred next, an implementation may choose to set PxTFD.STS.BSY. The requirement is that PxTFD.STS.BSY must be set before the command is issued to the device.

5.2.2.10 H:StartComm

H:StartComm	The HBA tells link layer to start communication, which involves sending COMRESET to device. The HBA resets PxTFD to 7Fh, and sets hbaUpdateSig to '1'.		
1.	PxSCTL.DET = 1h	→	H:StartComm
2.	Unconditional	→	H:NotRunning

NOTE:
Hardware polling to determine if a device is present is an implementation specific detail. A polling strategy is not specified in AHCI.

The H:StartComm state is entered to start communication with the device by issuing a COMRESET.

5.2.2.11 H:PowerOn

H:PowerOn	The HBA drives the appropriate external pin to a level which enables the FET to supply power to the device.		
1.	Unconditional	→	H:NotRunning

The H:PowerOn state is used to power on a device port.

5.2.2.12 H:PowerOff

H:PowerOff	The HBA drives the appropriate external pin to a level which disables the FET from supplying power to the device.		
1.	Unconditional	→	H:NotRunning

The H:PowerOff state is used to power off a device port.

5.2.2.13 H:PhyListening

H:PhyListening	The HBA sets the Phy into listen mode, as defined in section 10.9.1.		
	1. Unconditional	→	H:NotRunning

This state is entered when PxCMD.SUD is set to '0' from a previous value of '1' and PxSCTL.DET = '0'.

5.2.3 Aggressive Power Management States

These states are entered when the HBA has no more commands to operate on and may aggressively enter a lower power state on the link.

5.2.3.1 AggrPM:Entry

AggrPM:Entry			
	1. PxCI != 0h or PxSACT != 0h	→	H:Idle
	2. PxCMD.ALPE = '1' and PxCMD.ASP = '0' and CAP.PSC = '1' and PxSCTL.IPM != '1h' and PxSCTL.IPM != '3h'	→	AggrPM:Partial
	3. PxCMD.ALPE = '1' and PxCMD.ASP = '1' and CAP.SSC = '1' and PxSCTL.IPM != '2h' and PxSCTL.IPM != '3h'	→	AggrPM:Slumber
	4. Else	→	H:Idle

In this state, the HBA determines whether it can place the link into a low power state if aggressive power management is enabled. The link can only be placed into a low power state if both the PxCI and the PxSACT registers are cleared to zero which indicates there are no commands outstanding.

5.2.3.2 AggrPM:Partial

H:AggrPartial	HBA attempts to place the link in the Partial interface power management state		
	1. Unconditional	→	H:Idle

The HBA is not guaranteed to succeed in placing the link in the Partial interface power management state due to a number of reasons including the current value of the PxSSTS.IPM field and whether the device responds with $PMNAK_P$ to the request.

5.2.3.3 AggrPM:Slumber

H:AggrSlumber	HBA attempts to place the link in the Slumber interface power management state		
	1. Unconditional	→	H:Idle

The HBA is not guaranteed to succeed in placing the link in the Slumber power management state due to a number of reasons including the current value of the PxSSTS.IPM field and whether the device responds with $PMNAK_P$ to the request.

5.2.4 Non-Data FIS Receive States

5.2.4.1 NDR:Entry

NDR:Entry	Receive up to 64 bytes of FIS into internal FIFO.		
	1. Reception error (FIS reception failed)	→	ERR:Non-fatal
	2. FIS is longer than 64 bytes	→	ERR:Fatal
	3. FIS Type is D2H Register and PxCMD.ST=0	→	H:RegFisUpdate
	4. Else	→	NDR:Accept

This state is entered when the HBA has received a non-Data FIS.

5.2.4.2 NDR:Accept

NDR:Accept	HBA accepts the FIS with a status of R_OK.		
	1. FIS type is D2H Register FIS or PIO Setup FIS and (PxTFD.STS.BSY = '0' and PxTFD.STS.DRQ = '0')	→	H:Idle
	2. FIS Type is D2H Register	→	RegFIS:Entry
	3. FIS Type is Set Device Bits	→	SDB:Entry
	4. FIS Type is DMA Activate	→	DX:Entry

5.	FIS Type is DMA Setup	→	DmaSet:Entry
6.	FIS Type is BIST Activate and far-end loopback is enabled in the FIS	→	BIST:FarEndLoopback
7.	FIS Type is BIST Activate and far-end loopback is not enabled in the FIS	→	BIST:TestOngoing
8.	FIS Type is PIO Setup	→	PIO:Entry
9.	Else (FIS type is unknown)	→	UFIS:Entry

In this state, the non-Data FIS received has been verified to be valid. If the FIS is a Register FIS or PIO Setup FIS and PxTFD.STS.BSY and PxTFD.STS.DRQ are cleared, the HBA accepts the FIS with R_OK but does not perform any updates based on it; see section 6.1.2.

5.2.5 Command Transfer States

5.2.5.1 CFIS:SyncEscape

CFIS:SyncEscape	HBA performs a SYNC escape until the interface is quiescent. HBA sets hbaUpdateSig to '1' to flag that the signature needs to be updated.

1.	Unconditional	→	CFIS:Xmit

This state is entered when a reset FIS has been constructed to send to the device. In this state, the HBA shall perform a SYNC escape on the interface until the interface is quiescent.

5.2.5.2 CFIS:Xmit

CFIS:Xmit	HBA performs the following actions in the order specified: 1. HBA sets PxTFD.STS.BSY to '1' 2. Sets hbaDataTag = hbaIssueTag, hbaPMP = CTBA(hbaIssueTag)[PMP], hbaXferAtapi = CTBA(hbaIssueTag)[A], hbaDmaXferCnt = 0 3. Fetches command FIS from CTBA(hbaIssueTag)[CFIS] if not prefetched. 4. Transmits FIS to device, with a length given by CTBA(hbaIssueTag)[CFL]

1.	X_RDY/X_RDY collision on interface, delay FIS transfer	→	H:Idle
2.	SYNC escape received for FIS	→	ERR:SyncEscapeRecv
3.	R_OK status received for FIS	→	CFIS:Success
4.	Else (FIS transfer failed)	→	ERR:Non-fatal

This state is entered to transmit a command FIS to the device. If an X_RDY/X_RDY collision occurs on the interface, the HBA will return to idle and attempt the FIS transmission again at a later time. If the FIS transmission starts but then fails for some reason, non-fatal error bits will be set in the non-fatal error state and the HBA will attempt to retransmit the command FIS at a later time.

5.2.5.3 CFIS:Success

CFIS:Success	HBA clears hbaCmdToIssue to '0'.

1.	CTBA(hbaIssueTag).B (BIST) = '1'	→	BIST:TestOngoing
2.	CTBA(hbaIssueTag).C (Clear BSY upon R_OK) = '1'	→	CFIS:ClearCI
3.	CTBA(hbaIssueTag).A (ATAPI) = '1' and CTBA(hbaIssueTag).P[1] (Prefetchable) = '1'	→	CFIS:PrefetchACMD
4.	CTBA(hbaIssueTag).P[1] (Prefetchable) = '1'	→	CFIS:PrefetchPRD
5.	Else	→	H:Idle

NOTE:
1. The P bit may be ignored by hardware although it is recommended to be observed to avoid prefetching unnecessary data.
2. An HBA implementation may choose to ignore arcs 3 and 4 if it is not desired to prefetch after the command FIS is sent or if the prefetch has already been done.

This state is entered after a command FIS is transferred successfully.

5.2.5.4 CFIS:ClearCI

CFIS:ClearCI	HBA clears PxTFD.STS.BSY to '0'. HBA clears PxCI.CI(hbaIssueTag) to '0'. HBA sets hbaIssueTag to 32.

1.	Unconditional	→	AggrPM:Entry

This state is entered when the HBA successfully transmits a command FIS to the device and the Clear Bsy upon R_OK (C) bit is set in the command header. The PRD byte count is not required to be updated in this state since no data transfer can take place.

5.2.5.5 CFIS:PrefetchACMD

CFIS:PrefetchACMD	HBA may prefetch all or part of the ATAPI command at CTBA(hbaDataTag)[ACMD] based on implementation specific algorithm and buffer space.

1.	CTBA(hbaIssueTag).P[1] (Prefetchable) = '1'	→	CFIS:PrefetchPRD
2.	Else	→	H:Idle

NOTE:
1. The P bit may be ignored by hardware although it is recommended to be observed to avoid prefetching unnecessary data.

This state is entered after a command FIS has been transferred for an ATAPI command in order to prefetch the ATAPI command to be sent to the device.

5.2.5.6 CFIS:PrefetchPRD

CFIS:PrefetchPRD	HBA prefetches some number of PRDs, based on implementation specific algorithm and buffer space.

1.	CH(hbaIssueTag).W (Write) set	→	CFIS:PrefetchData
2.	Else	→	H:Idle

This state is entered after a command FIS has been transferred and there are PRD entries that can be prefetched for the command.

5.2.5.7 CFIS:PrefetchData

CFIS:PrefetchData	HBA prefetches some amount of data, based on implementation specific algorithm and buffer space available.

1.	Unconditional	→	H:Idle

This state is entered after a command FIS has been transferred and there is data to be written to the device that can be prefetched.

5.2.6 ATAPI Command Transfer States

The states in this portion of the state machine are responsible for transmitting the ACMD field to the device.

5.2.6.1 ATAPI:Entry

ATAPI:Entry	The HBA constructs a Data FIS with the minimum of hbaDmaXferCnt bytes or 16 bytes of data fetched from CTBA(hbaDataTag)[ACMD] as necessary. The PMP field of the Data FIS is set to the value in PxCLB[CH(hbaDataTag)][PMP]. HBA transmits the Data FIS to the device. HBA clears hbaXferAtapi to 0.

1.	Error occurred on transmission	→	ERR:Fatal
2.	Else (transmission successful, R_OK received)	→	H:Idle

This state is entered to transmit the ATAPI command to the device.

5.2.7 D2H Register FIS Receive States

5.2.7.1 RegFIS:Entry

RegFIS:Entry	HBA performs the following actions in order: 1. Copies Register FIS to system memory at PxFB[RFIS]. 2. Updates PxTFD.ERR register with value in the Error field of the Register FIS 3. Updates PxTFD.STS register with value in the Status field of the Register FIS

1.	PxTFD.STS.ERR = '1'	→	ERR:Fatal

2.	PxTFD.STS.BSY ='0' and PxTFD.STS.DRQ ='0'	→	RegFIS:ClearCI
3.	Else	→	RegFIS:UpdateSig

This state is entered after a valid D2H Register FIS arrives from the device. The purpose of this state is to update HBA registers and system memory appropriately based on the D2H Register FIS received.

5.2.7.2 RegFIS:ClearCI

RegFIS:ClearCI	PxCLB[CH(hbaIssueTag)][PRDBC] updated to reflect total number of bytes successfully transferred. HBA clears PxCI.CI(hbaIssueTag) to '0'. HBA sets hbaIssueTag to 32.

1.	Register FIS I bit set	→	RegFIS:SetIntr
2.	Else	→	RegFIS:UpdateSig

This state is entered to clear the command issue bit in the PxCI register corresponding to the last command issued. In this state the HBA updates the PRD byte count according to rules in section 5.3.1, clears PxCI.CI(hbaIssueTag) to '0' and sets hbaIssueTag to 32.

5.2.7.3 RegFIS:SetIntr

RegFIS:SetIntr	HBA sets PxIS.DHRS to '1'.

1.	PxIE.DHRE = '1'	→	RegFIS:SetIS
2.	Else	→	RegFIS:UpdateSig

This state is entered when the interrupt 'I' bit is set in the D2H Register FIS received.

5.2.7.4 RegFIS:SetIS

RegFIS:SetIS	HBA sets IS.IPS(x) to '1'.

1.	GHC.IE = '1'	→	RegFIS:GenIntr
2.	Else	→	RegFIS:UpdateSig

This state is entered when the interrupt 'I' bit is set in the D2H Register FIS received and the enable for D2H Register FIS interrupts is set.

5.2.7.5 RegFIS:GenIntr

RegFIS:GenIntr	HBA generates an interrupt.

1.	Unconditional	→	RegFIS:UpdateSig

The RegFIS:GenIntr state is entered to generate an interrupt for the port after a Register FIS interrupt has occurred. See section 10.6 for information on how an HBA generates an interrupt.

5.2.7.6 RegFIS:UpdateSig

RegFIS:UpdateSig	

1.	hbaUpdateSig = '1'	→	RegFIS:SetSig
2.	Else	→	AggrPM:Entry

The RegFIS:UpdateSig state is entered to check whether the PxSIG register needs to be updated.

5.2.7.7 RegFIS:SetSig

RegFIS:SetSig	HBA updates PxSIG with appropriate fields from D2H Register FIS as outlined in 3.3.9, and clears hbaUpdateSig to '0'

1.	Unconditional	→	AggrPM:Entry

The RegFIS:SetSig state updates the PxSIG register.

5.2.8 PIO Setup Receive States

5.2.8.1 PIO:Entry

PIO:Entry	HBA performs the following actions in order: 1. Copies the PIO Setup FIS to system memory at PxFB[PSFIS]. 2. Sets hbaPioXfer to '1' 3. Sets hbaPioESts to E_Status field of the FIS 4. Sets hbaPioErr to Error field of the FIS 5. Sets hbaDmaXferCnt to Transfer Count field of the FIS 6. Sets PxTFD.STS register to Status field of the FIS

1.	PxTFD.STS.ERR = '1'	→	ERR:Fatal
2.	D bit in the FIS is cleared to 0 and hbaXferAtapi is set to 0 (Data FIS should be transmitted)	→	DX:Entry
3.	D bit in the FIS is cleared to 0 and hbaXferAtapi is set to 1 (ATAPI command should be transmitted)	→	ATAPI:Entry
4.	Else (Data FIS will be received from device)	→	H:Idle

This state receives a PIO Setup FIS and updates the appropriate state based on the PIO Setup FIS.

5.2.8.2 PIO:Update

PIO:Update	HBA performs the following actions: 1. Updates PxTFD.STS with hbaPioESts 2. Updates PxTFD.ERR with hbaPioErr 3. Clears hbaPioXfer = '0'

1.	PxTFD.STS.ERR = '1'	→	ERR:Fatal
2.	PxTFD.STS.BSY = '0' and PxTFD.STS.DRQ = '0'	→	PIO:ClearCI
3.	hbaPioIbit = '1'	→	PIO:SetIntr
4.	Else	→	H:Idle

This state updates appropriate registers after the PIO transfer has completed.

5.2.8.3 PIO:ClearCI

PIO:ClearCI	PxCLB[CH(hbaIssueTag)][PRDBC] updated to reflect total number of bytes successfully transferred. The HBA clears PxCI.CI(hbaIssueTag) to '0' and sets hbaIssueTag to 32.

1.	hbaPioIbit = '1'	→	PIO:SetIntr
2.	Else	→	AggrPM:Entry

This state is entered to mark the current command as completed and allow the HBA to fetch a new command. In this state, the PRD byte count is updated according to the rules in section 5.3.1

5.2.8.4 PIO:SetIntr

PIO:SetIntr	Set PxIS.PSS to '1'

1.	PxIE.PSE = '1'	→	PIO:GenIntr
2.	Else	→	AggrPM:Entry

This state is entered when the PIO Setup FIS has the interrupt 'I' bit set.

5.2.8.5 PIO:GenIntr

PIO:GenIntr	HBA generates an interrupt.

1.	Unconditional	→	AggrPM:Entry

This state is entered to generate an interrupt for the port. See section 10.6 for information on how an HBA generates an interrupt.

5.2.9 Data Transmit States

5.2.9.1 DX:Entry

DX:Entry	The HBA constructs a Data FIS for command list entry hbaDataTag. The PMP field of the Data FIS is set to the value in PxCLB[CH(hbaDataTag)][PMP]. The HBA fetches PRDs and data from locations specified in the hbaDataTag command list entry.

1. No data to transmit and hbaPioXfer = '1'	→	PIO:Update
2. No data to transmit and hbaPioXfer = '0'	→	H:Idle
3. Else	→	DX:Start

This state starts to construct a Data FIS to transmit to the device. This state is entered after a PIO Setup, DMA Activate, or DMA Setup FIS is received.

5.2.9.2 DX:Transmit

DX:XmitStart	HBA transmits the Data FIS to the device. Continue to fetch PRDs and data as necessary to complete the Data FIS. As PRDs are completed and R_OK is received for the FIS containing the data in that PRD, log the 'I' bit in the PRD into hbaPrdIntr.

1. SYNC escape received for Data FIS	→	ERR:SyncEscapeRecv
2. Transmission failed	→	ERR:Fatal
3. End of PRD table reached, or maximum Data FIS length of 8KB has been transferred.	→	DX:UpdateByteCount

This state transmits a Data FIS to the device. The HBA will fetch PRDs and data as necessary to complete the Data FIS.

5.2.9.3 DX:UpdateByteCount

DX:UpdateByteCount	HBA performs the following actions: 1. Optionally increments PxCLB[CH(hbaDataTag)][PRDBC] by size of Data FIS transferred. 2. Decrements hbaDmaXferCnt by size of Data FIS transferred.

1. hbaPrdIntr = '1'	→	DX:PrdSetIntr
2. hbaPioXfer = '1'	→	PIO:Update
3. Else	→	H:Idle

This state updates transfer byte counts based on the Data FIS that was successfully transmitted. Update of the PRD byte count is optional, refer to section 5.3.1.

5.2.9.4 DX:PrdSetIntr

DX:PrdIntr	HBA sets PxIS.DPS to '1'. HBA clears hbaPrdIntr

1. PxIE.DPE = '1'	→	DX:PrdSetIS
2. hbaPioXfer = '1'	→	PIO:Update
3. Else	→	H:idle

The HBA enters this state when a PRD interrupt needs to be generated.

5.2.9.5 DX:PrdSetIS

DX:PrdSetIS	HBA sets IS.IPS(x) to '1'.

1. GHC.IE = '1'	→	DX:PrdGenIntr
2. hbaPioXfer = '1'	→	PIO:Update
3. Else	→	H:Idle

The HBA enters this state to set an interrupt on the controller for this port.

5.2.9.6 DX:PrdGenIntr

DX:GenIntr	HBA generates an interrupt.

1. hbaPioXfer = '1'	→	PIO:Update
2. Else	→	H:Idle

This state is entered to generate an interrupt. See section 10.6 for information on how an HBA generates an interrupt

5.2.10 Data Receive States

5.2.10.1 DR:Entry

DR:Entry			
1.	PMP field in the Data FIS does not equal hbaPMP	→	ERR:Non-fatal
2.	Else	→	DR:Receive

This state is entered when a Data FIS is received. This state checks to make sure the Port Multiplier Port field is valid.

5.2.10.2 DR:Receive

DR:Receive	HBA fetches PRD entries from the command list entry for hbaDataTag as necessary and transfers data from FIS to system memory. As PRDs complete and a valid CRC is received for the FIS containing the data in that PRD, if their 'I' bit is set, the HBA sets hbaPrdIntr.		
1.	Reception Failed	→	ERR:Fatal
2.	Reception Successful, more data arrived than could fit in PRD table	→	ERR:Non-fatal
3.	Else (Data FIS reception successful)	→	DR:UpdateByteCount

This state is entered when a Data FIS is received. The state transfers the data in the FIS to system memory until there is no data left to transfer or the end of the PRD table is reached.

5.2.10.3 DR:UpdateByteCount

DR:UpdateByteCount	HBA accepts FIS with status of R_OK. HBA optionally increments PxCLB[CH(hbaDataTag)][PRDBC] by number of bytes transferred to system memory. HBA decrements hbaDmaXferCnt by number of bytes transferred to system memory.		
1.	hbaPrdIntr = '1'	→	DR:PrdSetIntr
2.	hbaPioXfer = '1'	→	PIO:Update
3.	Else	→	H:Idle

This state is entered after the appropriate number of bytes in a Data FIS have been transferred to system memory and the Data FIS CRC has been verified as correct. Update of the PRD byte count is optional, refer to section 5.3.1.

5.2.10.4 DR:PrdSetIntr

DR:PrdSetIntr	HBA sets PxIS.DPS. HBA clears variable hbaPrdIntr		
1.	PxIE.DPE = '1'	→	DR:PrdSetIS
2.	hbaPioXfer = '1'	→	PIO:Update
3.	Else	→	H:Idle

The HBA enters this state to set the PRD status interrupt bit when a PRD interrupt occurs.

5.2.10.5 DR:PrdSetIS

DR:PrdSetIS	HBA sets IS.IPS(x) to '1'.		
1.	GHC.IE = '1'	→	DR:PrdGenIntr
2.	hbaPioXfer = '1'	→	PIO:Update
3.	Else	→	H:Idle

This state is entered to set the interrupt for this port on the controller.

5.2.10.6 DR:PrdGenIntr

DR:PrdGenIntr	HBA generates an interrupt.		
1.	hbaPioXfer = '1'	→	PIO:Update
2.	Else	→	H:Idle

This state is entered to generate an interrupt. See section 10.6 for information on how an HBA generates an interrupt.

5.2.11 DMA Setup Receive States

5.2.11.1 DmaSet:Entry

DmaSet:Entry	The HBA performs the following actions: 1. HBA stores TAG field from the DMA Setup FIS in hbaDataTag 2. HBA stores TransferCount field from the DMA Setup FIS in hbaDmaXferCnt 3. HBA writes the FIS to system memory at PxFB[DSFIS]

1.	I bit in the FIS is set to 1	→	DmaSet:SetIntr
2.	Else	→	DmaSet:AutoActivate

This state is entered when a correctly formed DMA Setup FIS is received.

5.2.11.2 DmaSet:SetIntr

DmaSet:SetIntr	HBA sets PxIS.DSS to '1'.

1.	PxIE.DSE = '1'	→	DmaSet:SetIS
2.	Else	→	DmaSet:AutoActivate

This state is entered when a DMA Setup FIS is received that has the I bit set.

5.2.11.3 DmaSet:SetIS

DmaSet:SetIntr	HBA sets IS.IPS(x) to '1'.

1.	GHC.IE = '1'	→	DmaSet:GenIntr
2.	Else	→	DmaSet:AutoActivate

In this state, the HBA sets the interrupt status bit for this port on the controller.

5.2.11.4 DmaSet:GenIntr

DmaSet:GenIntr	HBA generates an interrupt.

1.	Unconditional	→	DmaSet:AutoActivate

This state is entered to generate an interrupt. See section 10.6 for information on how an HBA generates an interrupt

5.2.11.5 DmaSet:AutoActivate

DmaSet:AutoActivate	

1.	CH(hbaDataTag).W is set to 1 and A bit in the FIS is set to 1	→	DX:Entry
2.	Else	→	H:Idle

This state is entered to determine whether a Data FIS should immediately be transmitted to the device.

5.2.12 Set Device Bits States

5.2.12.1 SDB:Entry

SDB:Entry	HBA performs the following actions: 1. Copies Set Device Bits FIS to system memory at offset PxFB[SDFIS] 2. Updates PxTFD.STS with non-reserved bits in Status field of FIS 3. Updates PxTFD.ERR with Error field of FIS 4. Clears bits in PxSACT that have corresponding bits set in the SActive field of the SDB FIS. Sets hbaSActive to value of SActive field in received FIS. 5. Clears hbaDmaXferCnt to '0'

1.	PxTFD.STS.ERR = '1'	→	ERR:Fatal
2.	I bit is set in the SDB FIS	→	SDB:SetIntr
3.	hbaSActive = '0'	→	H:Idle
4.	Else	→	AggrPM:Entry

This state receives a Set Device Bits FIS and updates the appropriate state based on the Set Device Bits FIS.

5.2.12.2 SDB:SetIntr

SDB:SetIntr	HBA sets PxIS.SDBS to '1'.		
	1. PxIE.SDBE = '1'	→	SDB:SetIS
	2. hbaSActive = '0'	→	H:Idle
	3. Else	→	AggrPM:Entry

This state is entered when a Set Device Bits FIS with the interrupt bit set was received.

5.2.12.3 SDB:SetIS

SDB:SetIS	HBA sets IS.IPS[x] bit.		
	1. GHC.IE bit set	→	SDB:GenIntr
	2. hbaSActive = '0'	→	H:Idle
	3. Else	→	AggrPM:Entry

This state is entered to set status bits in the global interrupt status register. Prior to entering this state, the HBA has set the appropriate PxIS bit based on the interrupt cause.

5.2.12.4 SDB:GenIntr

SDB:GenIntr	HBA generates an interrupt.		
	1. hbaSActive = '0'	→	H:Idle
	2. Unconditional	→	AggrPM:Entry

This state is entered to generate an interrupt for the port. See section 10.6 for information on how an HBA generates an interrupt.

5.2.13 Unknown FIS Receive States

5.2.13.1 UFIS:Entry

UFIS:Entry	HBA performs the following actions in order: 1. Copies the Unknown FIS to system memory at PxFB[UFIS] 2. Sets PxIS.UFS to '1'		
	1. PxIE.UFE is set to 1	→	UFIS:SetIS
	2. Else	→	H:Idle

This state is entered when an Unknown FIS is received and the CRC for that FIS is correct. When this state is entered, PxSERR.DIAG.F has already been set to '1'. Note that PxSERR.DIAG.F is set to '1' as soon as the HBA recognizes that an unknown FIS has been received.

5.2.13.2 UFIS:SetIS

UFIS:SetIS	HBA sets IS.IPS[x] bit.		
	1. GHC.IE bit set		UFIS:GenIntr
	2. Else	→	H:Idle

This state is entered to set status bits in the global interrupt status register.

5.2.13.3 UFIS:GenIntr

UFIS:GenIntr	HBA generates an interrupt.		
	1. Unconditional	→	H:Idle

This state is entered to generate an interrupt for the port. See section 10.6 for information on how an HBA generates an interrupt.

5.2.14 BIST States

5.2.14.1 BIST:FarEndLoopback

BIST:FarEndLoopback	HBA sets PxSSTS.DET to 4h.		
	1. Unconditional	→	BIST:TestOngoing

This state is entered when a BIST Activate FIS with far-end loopback mode enabled is received from the device.

5.2.14.2 BIST:TestOngoing

BIST:TestOngoing			
	1. Unconditional	→	BIST:TestOngoing

This state is entered when a BIST Activate FIS is sent by the HBA to the device or received from the device. In this state, testing is performed that is outside the scope of this specification.

Software exits the HBA from this state by clearing PxCMD.ST to '0'.

5.2.15 Error States

5.2.15.1 ERR:SyncEscapeRecv

ERR:SyncEscapeRecv	HBA performs the following actions in the order specified: 1. Clears both PxTFD.STS.BSY and PxTFD.STS.DRQ to '0'. 2. Sets PxIS.IFS to '1'.		
	1. Unconditional	→	ERR:Fatal

This state is entered when a SYNC escape is received for an H2D Register FIS or an H2D Data FIS. When a SYNC escape is received, the HBA shall not retransmit the FIS in accordance with Serial ATA 1.0a. In order to enable timely recovery without a COMRESET, the PxTFD.STS.BSY bit is cleared to '0' and PxIS.IFS is set to '1'. The HBA shall not clear any of the PxCI bits to ensure that software can determine the command that failed.

5.2.15.2 ERR:Fatal

This state is entered when the HBA has encountered a condition it cannot recover from. Appropriate error bits shall be set by the HBA, and interrupts may optionally be generated. See section 6 for fatal conditions. When a fatal condition occurs, the HBA requires PxCMD.ST to be cleared in order to recover.

5.2.15.3 ERR:Non-Fatal

This state is entered when the HBA has encountered an error it can recover from. Appropriate error bits shall be set by the HBA, and interrupts may optionally be generated. See section 6 for non-fatal conditions. When a non-fatal condition occurs, the HBA shall return to H:Idle after processing the error.

5.3 HBA Rules (Normative)

5.3.1 PRD Byte Count Updates

Each command has a PRD byte count field that is initialized to '0' by software prior to starting a transfer, and is updated by hardware as transfers occur. The purpose of this field is to let software know how many bytes transferred for a given operation in order to determine if underflow occurred.

For non-native queued commands, the PRD byte count field shall contain an accurate count of the number of bytes transferred for the command before the PxCI bit is cleared to '0' for the command. The byte count shall only reflect transmitted data that an R_OK has been received for and received data that has a valid CRC. Hardware may update the byte count after each Data FIS is transferred if desired.

The field should not be used and is not required to be valid when issuing native command queuing commands; in this case underflow is always illegal.

If an overflow occurs, the byte count is not required to reflect the total number of bytes transferred. The PxIS.OFS bit should be used to detect overflow.

5.3.1.1 Data FIS Odd Word Transfers

If the final Data FIS transfer in a command is for an odd number of 16-bit words, the transmitter's Transport layer is responsible for padding the final Dword of a FIS with zeros. If the HBA receives one more word than is indicated in the PRD table due to this padding requirement, the HBA shall not signal this as an overflow condition. In addition, if the HBA inserts padding as required in a FIS it is transmitting,

an overflow error shall not be indicated. The PRD Byte Count field shall be updated based on the number of words specified in the PRD table, ignoring any additional padding.

5.3.2 PRD Interrupt

When a PRD is exhausted, the HBA may be told to generate an interrupt via the 'I' bit in the PRD entry. Note, though, that a PRD is not considered exhausted until the Data FIS is complete. For example, if the Data FIS is 8 KB, and this is covered by 3 PRD entries, the data is not considered valid at the end of the first or second PRD, since CRC has not yet been checked, even though the data has been copied to memory or the device.

Therefore, if the 'I' bit is set in the PRD entry, the HBA must hold onto it internally and not set PxIS.DPS until the Data FIS is complete and CRC is correct. Once correct, PxIS.DPS can be set, and if PxIE.IE and GHC.IE are set, the HBA shall generate an interrupt.

The PRD Interrupt is an opportunistic interrupt. The PRD Interrupt should not be used to definitively indicate the end of a transfer. Two PRD interrupts could happen close in time together such that the second interrupt is missed when the first PRD interrupt is being cleared.

5.4 System Software Rules (Normative)

5.4.1 Basic Steps when Building a Command

When software builds a command for the HBA to execute, it first finds an empty command slot by reading the PxCI register for the port. After a free slot (slot z), is found:

1. Software builds a command FIS in system memory at location PxCLB[CH(z)]:CFIS with the command type.
2. If it is an ATAPI command, the ACMD field shall be filled in with the ATAPI command
3. Software builds a command header at PxCLB[CH(z)] with:
   a. PRDTL containing the number of entries in the PRD table
   b. CFL set to the length of the command in the CFIS area
   c. A bit set if it is an ATAPI command
   d. W (Write) bit set if data is going to the device
   e. P (Prefetch) bit optionally set (see rules in section 5.4.2)
   f. If a Port Multiplier is attached, the PMP field set to the correct Port Multiplier port.
4. If it is a queued command, software shall first set PxSACT.DS(z).
5. Software shall set PxCI.CI(z) to indicate to the HBA that a command is active.

5.4.2 Setting CH(z).P

When software builds commands for the HBA to execute, it may optionally set CH(z).P to enable prefetching of PRDs and data. The HBA is not required to prefetch, but may use this bit to do so. To avoid potential problems where the HBA may prefetch items it cannot use, software must obey the following rules:

- Software shall not set CH(z).P when building queued ATA commands. The S-ATA device may run the commands in a different order than what is sent.
- Software shall not set CH(z).P when building commands to several devices behind a Port Multiplier when FIS based switching is enabled. FISes may be received from a Port Multiplier for a different device than what was just sent by the HBA.

If software does not obey these rules, indeterminate HBA behavior may result.

5.4.3 Processing Completed Commands

Software processes the interrupt generated by the device for command completion. In the interrupt service routine, software checks IS.IPS to determine which ports have an interrupt pending.

For each port that has an interrupt pending:

1. Software determines the cause of the interrupt by reading the PxIS register. It is possible for multiple bits to be set

2. Software clears appropriate bits in the PxIS register corresponding to the cause of the interrupt.
3. Software clears the interrupt bit in IS.IPS corresponding to the port.
4. Software reads the PxCI register, and compares the current value to a previously read value. It completes with success any commands whose bit has been cleared since the last value was read.
5. If there were errors, noted either in the PxIS register or PxTFD.STS.ERR, software performs error recovery actions (see section 6.2.2).

5.5 Transfer Examples (Informative)

In all the following examples, it is assumed that PxCMD.ST has been set, and the HBA is only waiting for a command to be placed in the command list. If PxCMD.ST is not set, software must do so at some point. Additionally, software must ensure that the device has not changed connection status since the last command was added by checking PxIS.PCS.

At any point, the Serial ATA link may be in a Partial or Slumber interface power management state. The HBA shall ensure that the link is active before transmitting a FIS on the Serial ATA link (refer to section 8.3.1.3).

5.5.1 Macro States

In the following examples, the HBA traverses a series of states in the HBA state machine when performing data transfers. To simplify the text in the examples, state sequences that are repeated are abbreviated here in what are called macro-states. Each of these macro-states assumes no errors were encountered.

Macro State	HBA State Machine States
Exam:Fetch	H:Idle → H:SelectCmd → H:FetchCmd → H:Idle
Exam:Transmit	H:Idle → CFIS:Xmit → CFIS:Success → H:Idle
Exam:AcceptNonData	H:Idle → NDR:Entry → NDR:Accept
Exam:DMAReceive	DR:Entry → DR:Receive → DR:UpdateByteCount → H:Idle
Exam:DMATransmit	DX:Entry → DX:Transmit → DX:UpdateByteCount → H:Idle
Exam:PIOTransmit	PIO:Entry → DX:Entry → DX:Transmit → DX:UpdateByteCount → PIO:Update → PIO:ClearCI → PIO:SetIntr → PIO:GenIntr → ChkAggrPM:Entry → H:Idle
Exam:PIOReceive	DR:Entry → DR:Receive → DR:UpdateByteCount → PIO:Update → PIO:ClearCI → PIO:SetIntr → PIO:GenIntr → ChkAggrPM:Entry → H:Idle
Exam:D2HIntr	RegFIS:Entry → RegFIS:ClearCI → RegFIS:SetIntr → RegFIS:SetIS → RegFIS:GenIntr → RegFIS:UpdateSig → ChkAggrPM:Entry → H:Idle
Exam:D2HNoIntr	RegFIS:Entry → RegFIS:ClearCI → RegFIS:UpdateSig → H:ChkAggrPM → H:Idle
Exam:DMASetup	DmaSet:Entry → DmaSet:SetIntr → DmaSet:SetIS → DmaSet:GenIntr → DmaSet:AutoActivate → H:Idle

5.5.2 DMA Data Transfers

5.5.2.1 ATA DMA Write

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a DMA write (data to device), therefore CH(z).W (Write) shall be set to '1', and CH(z).P (Prefetch) may optionally be set to '1' per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set to '1', the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → CFIS:PrefetchData → H:Idle.

As this was a DMA write command, the response from the device shall be a DMA Activate FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state, and shall send a Data FIS by traversing the **Exam:DMATransmit** macro state.

If the Data FIS did not satisfy the transfer count, another DMA Activate FIS shall be sent from the device, and the HBA shall traverse the **Exam:AcceptNonData** and **Exam:DMATransmit** macro states again to send another Data FIS. This process continues until the transfer count is satisfied. When the Data FIS that completes the transfer count finishes, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set to '1', the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set to '1', the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the Partial or Slumber interface power management state after the AggrPM:Entry state.

5.5.2.2 ATAPI Packet DMA Write

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is ATAPI, and is not queued. It is a DMA write (data to device), therefore CH(z).W (Write) shall be set to '1', and CH(z).P (Prefetch) may optionally be set per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set, the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchACMD → CFIS:PrefetchPRD → CFIS:PrefetchData → H:Idle.

Since this was an ATAPI command, the next FIS from the device shall be a PIO Setup FIS. The HBA traverses the **Exam:AcceptNonData** macro state to accept this FIS, and then traverses the following states to transfer the ACMD field to the device: PIO:Entry → ATAPI:Entry → ATAPI:Success → H:Idle

As this command results in a DMA write, the response from the device shall be a DMA Activate FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state, and shall send a Data FIS by traversing the **Exam:DMATransmit** macro state.

If the Data FIS did not satisfy the transfer count, another DMA Activate FIS shall be sent from the device, and the HBA shall traverse the **Exam:AcceptNonData** and **Exam:DMATransmit** macro states again to send another Data FIS. This process continues until the transfer count is satisfied. When the Data FIS that completes the transfer count finishes, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set to '1', the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set to '1', the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the Partial or Slumber interface power management state after the AggrPM:Entry state.

5.5.2.3 ATA DMA Read

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a DMA read (data to memory), therefore CH(z).W (Write) shall be cleared to '0', and CH(z).P (Prefetch) may optionally be set to '1' per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set to '1', the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → H:Idle

As this was a DMA read command, the response from the device shall be a Data FIS. When this arrives, the HBA shall traverse the **Exam:DMAReceive** macro state.

If the Data FIS did not satisfy the transfer count, another Data FIS shall be sent from the device, and the HBA shall traverse the **Exam:DMAReceive** macro states again. This process continues until the transfer count is satisfied. When the Data FIS that completes the transfer count finishes, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set to '1', the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set to '1', the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the Partial or Slumber interface power management state after the AggrPM:Entry state.

5.5.2.4 ATAPI Packet DMA Read

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a DMA read (data to memory), therefore CH(z).W (Write) shall be cleared to '0', and CH(z).P (Prefetch) may optionally be set to '1' per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set to '1', the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchACMD → CFIS:PrefetchPRD → H:Idle.

Since this was an ATAPI command, the next FIS from the device shall be a PIO Setup FIS. The HBA traverses the **Exam:AcceptNonData** macro state to accept this FIS, and then traverses the following states to transfer the ACMD field to the device: PIO:Entry → ATAPI:Entry → ATAPI:Success → H:Idle.

As this command results in a DMA read, the response from the device shall be a Data FIS. When this arrives, the HBA shall traverse the **Exam:DMAReceive** macro state.

If the Data FIS did not satisfy the transfer count, another Data FIS shall be sent from the device, and the HBA shall traverse the **Exam:DMAReceive** macro states again. This process continues until the transfer count is satisfied. When the Data FIS that completes the transfer count finishes, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set to '1', the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set to '1', the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the Partial or Slumber interface power management state after the AggrPM:Entry state.

5.5.3 PIO Data Transfers

The following sections describe data transfers to and from a device using PIO as the command protocol type. PIO data transfers are highly discouraged because of the inadequate error handling coverage for PIO read commands. Some AHCI implementations may choose to only implement support to transfer a single DRQ block in a PIO command. If CAP.PMD is cleared to '0', then the HBA only supports single DRQ block PIO transfers and software must ensure commands are not issued to the device that are for more than one DRQ block of data.

From the HBA's point of view, PIO data transfers look like a DMA transfer. A command table is set up, and the data is bus mastered from or to system memory by the HBA.

5.5.3.1 ATA PIO Write

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a PIO Write (data to device), therefore CH(z).W (Write) shall be set to '1', and CH(z).P (Prefetch) may optionally be set to '1' per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set, the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → CFIS:PrefetchData → H:Idle.

As this was a PIO write command, the response from the device shall be a PIO Setup FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state. It shall then traverse the **Exam:PIOTransmit** macro state to send a Data FIS.

Since this was PIO write command, the device shall next send a D2H Register FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set to '1', the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set to '1', the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the Partial or Slumber interface power management state after the AggrPM:Entry state.

5.5.3.2 ATAPI Packet PIO Write

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is ATAPI, and is not queued. It is a PIO Write (data to device), therefore CH(z).W (Write) shall be set to '1, and CH(z).P (Prefetch) may optionally be set to '1' per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set to '1', the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchACMD → CFIS:PrefetchPRD → CFIS:PrefetchData → H:Idle.

Since this was an ATAPI command, the next FIS from the device shall be a PIO Setup FIS. The HBA traverses the **Exam:AcceptNonData** macro state to accept this FIS, and then traverses the following states to transfer the ACMD field to the device: PIO:Entry → ATAPI:Entry → ATAPI:Success → H:Idle

As this was a PIO Write command, the response from the device shall be a PIO Setup FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state. It shall then traverse the **Exam:PIOTransmit** macro state to send a Data FIS to the device.

Since this was an PIO ATAPI write command, the device shall send a D2H Register FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set to '1', the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set to '1', the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the Partial or Slumber interface power management state after the AggrPM:Entry state.

5.5.3.3 ATA PIO Read

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a PIO Read (data to memory), therefore CH(z).W (Write) shall be cleared to '0', and CH(z).P (Prefetch) may optionally be set to '1' per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set to '1', the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → H:Idle

As this was a PIO read command, the response from the device shall be a PIO Setup FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state. It shall then traverse the states PIO:Entry → H:Idle, and await a Data FIS from the device.

When the Data FIS arrives, the HBA traverses the **Exam:PIOReceive** macro state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the Partial or Slumber interface power management state after the AggrPM:Entry state.

5.5.3.4 ATAPI Packet PIO Read

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is ATAPI, and is not queued. It is a PIO Read (data to memory), therefore CH(z).W (Write) shall be cleared to '0', and CH(z).P (Prefetch) may optionally be set to '1' per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set to '1', the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchACMD → CFIS:PrefetchPRD → H:Idle.

Since this was an ATAPI command, the next FIS from the device shall be a PIO Setup FIS. The HBA traverses the **Exam:AcceptNonData** macro state to accept this FIS, and then traverses the following states to transfer the ACMD field to the device: PIO:Entry → ATAPI:Entry → ATAPI:Success → H:Idle

As this was a PIO read command, the response from the device shall be a PIO Setup FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state. It shall then traverse the states PIO:Entry → H:Idle, and await a Data FIS from the device.

When the Data FIS arrives, the HBA traverses the **Exam:PIOReceive** macro state.

Since this was an ATAPI command, the device shall next send a D2H Register FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the Exam:D2HIntr macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the Partial or Slumber interface power management state after the AggrPM:Entry state.

5.5.4 HBA Assisted Queued DMA Transfers

5.5.4.1 Introduction

Legacy ATA queued DMA, using the DRQ, SERV, and REL bits, is not supported by AHCI. Queued operations are sent to an SATA device using the READ FPDMA QUEUED and WRITE FPDMA QUEUED commands.

To allow a simple mechanism for the HBA to map command list slots to queue entries, software must match the tag number it uses to the slot it is placing the command in. For example, if a queued command is placed in slot 5, the tag for that command must be 5.

System software must determine the maximum tag allowed by the device and the HBA and it must use the lower bound of the two. For example, if the HBA has 8 entries in its command list, and the SATA device only has 4, only tags 0 – 3 in the device may be used, and only command list entries 0 – 3 may be used in the HBA.

Data transfers are activated with the flow described below via the DMA Setup FIS, and command completion is performed via the Set Device Bits FIS.

5.5.4.2 Example

In the following example, the following occurs:

- System software places 4 commands in system memory:
  - Slot 0 contains a queued DMA read
  - Slot 2 contains a queued DMA write
  - Slot 5 contains a queued DMA read
  - Slot 8 contains a queued DMA write
- The HBA fetches the first 3 commands, transfers them to the device, and receives a successful completion.
- Before the HBA can send the 4th command to the device, the device sends a DMA Setup FIS to transfer data for the command in slot 2.
- A data transfer occurs to slot 2.

- The HBA transfers slot 8 to the device.
- A data transfer occurs to slot 5.
- The device sends an SDB FIS to clear slots 2 and 5.
- A data transfer occurs to slot 8.
- The device sends an SDB FIS to clear slot 8.
- A data transfer occurs to slot 0.
- The device sends an SDB FIS to clear slot 0.

Other items to note in this data flow:

- In both queued DMA read commands, the auto-activate bit is not set in the FIS.
- Every SDB FIS received has the 'I' bit set to '1'.

Following is a text description for the flow.

System Software Places 4 Commands in System Memory

Software builds a command into slot 0 as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is of type ReadFPDMAQueued. CH(z).W (Write) shall be cleared to '0', and CH(z).P (Prefetch) shall be cleared.

Software builds a command into slot 2 as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is of type WriteFPDMAQueued. CH(z).W (Write) shall be set to '1' and CH(z).P (Prefetch) shall be cleared to '0'.

Software builds a command into slot 5 as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is of type WriteFPDMAQueued. Both CH(z).W (Write) and CH(z).P (Prefetch) shall be cleared.

Software builds a command into slot 8 as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is of type ReadFPDMAQueued. CH(z).W (Write) shall be set to '1', and CH(z).P (Prefetch) shall be cleared.

At this point, the PxCI and PxSACT register values are "00000125h" (bits 0, 2, 5, and 8 set).

The HBA transmits the First 3 Commands to the Device

The HBA shall transfer the command from slot 0 to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. As this was a queued DMA command, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the Exam:D2HIntr macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state. PxCI is now equal to "00000124h".

The HBA shall transfer the command from slot 2 to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. As this was a queued DMA command, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the Exam:D2HIntr macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state. PxCI is now equal to "00000120h".

The HBA shall transfer the command from slot 5 to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. As this was a queued DMA command, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the Exam:D2HIntr macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state. PxCI is now equal to "00000100h".

DMA Setup FIS Arrives for slot 2

The HBA is now in an idle state, but before it can fetch a new command, a short FIS arrives from the device. This FIS is the DMA Setup FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the **Exam:DMASetup** macro state to process the DMA Setup FIS. The tag indicated in the FIS was for slot 2.

Data Transfer for slot 2

As this was a DMA write command, and the auto-activate bit was not set in the FIS, the next FIS from the device shall be a DMA Activate FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state, and shall send a Data FIS by traversing the **Exam:DMATransmit** macro state.

If the Data FIS did not satisfy the transfer count, another DMA Activate FIS shall be sent from the device, and the HBA shall traverse the **Exam:AcceptNonData** and **Exam:DMATransmit** macro states again to send another Data FIS. This process continues until the transfer count is satisfied. The state machine is now in H:Idle

HBA transfers slot 8 to the device

Since PxCI is not all 0h, and no other FIS is coming in from the device, the HBA shall transfer the command from slot 8 to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. As this was a queued DMA command, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state. PxCI is now equal to "00000000h".

Data transfer to slot 5

A short FIS arrives from the device of type DMA Setup FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the **Exam:DMASetup** macro state to process the DMA Setup FIS. The tag indicated in the FIS was for slot 5.

As this was a DMA read command, the next FIS from the device shall be a Data FIS. When this arrives, the HBA shall traverse the **Exam:DMAReceive** macro state.

If the Data FIS did not satisfy the transfer count, another Data FIS shall be sent from the device, and the HBA shall traverse the **Exam:DMAReceive** macro states again. This process continues until the transfer count is satisfied. The HBA state machine is now in H:Idle.

Device sends SDB FIS to clear slots 2 and 5

At this point, the device sends an SDB FIS to indicate slots 2 and 5 are complete. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the SDB:Entry, and, since the received FIS had its I bit set, the SDB:SetIntr → SDB:SetIS → SDB:GenIntr states, and returns to H:Idle. The PxSACT register is now equal to "00001001h".

Data transfer occurs to slot 8

A short FIS arrives from the device of type DMA Setup FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the **Exam:DMASetup** macro state to process the DMA Setup FIS. The tag indicated in the FIS was for slot 8.

As this was a DMA write command, and the auto-activate bit was not set in the FIS, the next FIS from the device shall be a DMA Activate FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state, and shall send a Data FIS by traversing the **Exam:DMATransmit** macro state.

If the Data FIS did not satisfy the transfer count, another DMA Activate FIS shall be sent from the device, and the HBA shall traverse the **Exam:AcceptNonData** and **Exam:DMATransmit** macro states again to send another Data FIS. This process continues until the transfer count is satisfied. The HBA state machine is now in H:Idle.

Device sends SDB FIS to clear slot 8

At this point, the device sends an SDB FIS to indicate slot 8 is complete. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the SDB:Entry, and, since the received FIS had its I bit set, the SDB:SetIntr → SDB:SetIS → SDB:GenIntr states, and returns to H:Idle. The PxSACT register is now equal to "00000001h".

Data transfer occurs to slot 0

A short FIS arrives from the device of type DMA Setup FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the **Exam:DMASetup** macro state to process the DMA Setup FIS. The tag indicated in the FIS was for slot 2.

As this was a DMA read command, the next FIS from the device shall be a Data FIS. When this arrives, the HBA shall traverse the **Exam:DMAReceive** macro state.

If the Data FIS did not satisfy the transfer count, another Data FIS shall be sent from the device, and the HBA shall traverse the **Exam:DMAReceive** macro states again. This process continues until the transfer count is satisfied. The HBA state machine is now in H:Idle.

Device sends SDB FIS to clear slot 0

At this point, the device sends an SDB FIS to indicate slot 0 is complete. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the SDB:Entry, and, since the received FIS had its I bit set, the SDB:SetIntr → SDB:SetIS → SDB:GenIntr states, and returns to H:Idle. The PxSACT register is now equal to "00000000h".

Since the PxCI and PxSACT registers are now both equal to "00000000h", if the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the Partial or Slumber power management after the AggrPM:Entry state.

6 Error Reporting and Recovery

All errors within an HBA occur within ports. There are no errors that apply to the entire host controller. There are several sources of errors that could occur during a transfer. Examples of errors are:

- System Memory – Bad system memory pointers cause data fetches and stores to be lost
- Interface / Device - such as CRC problems, illegal state machine transitions, etc.

6.1 Error Types

6.1.1 System Memory Errors

System memory errors such as target abort, master abort, and parity may cause the host to stop processing the currently running command. These are serious errors that cannot be recovered from without software intervention (section 6.2.2).

A master/target abort error occurs when system software has given a pointer to the HBA that does not exist in physical memory. When this occurs, the HBA aborts the transfer (if necessary) as described in section 6.2.1. When this is complete, the HBA sets PxIS.HBFS. If PxIE.HBFE and GHC.IE are set, the HBA shall also generate an interrupt. .

A data error (such as CRC or parity), may or may not be transient. If the error occurred on a fetch of a CFIS, PRD entry or command list, the HBA shall stop. If the error occurred on a data FIS or the ACMD field, the HBA is allowed to stop, but may also continue. When a data error occurs, the HBA aborts the transfer (if necessary) as described in section 6.2.1. When this is complete, the HBA sets PxIS.HBDS. If PxIE.HBDE and GHC.IE are set, the HBA shall also generate an interrupt.

If the HBA continue after a data error on a data or ACMD field, it shall poison the CRC of the Data FIS it transfers to the device.

6.1.2 Interface Errors

Interface errors are errors that occur due to electrical issues on the interface, or protocol miscommunication between the device and HBA. Depending on the type of error, different bits in the PxSERR register are set. When these bits are set, either PxIS.IFS (fatal) or PxIS.INFS (non-fatal) shall be set, and if enabled, the HBA shall generate an interrupt.

Conditions that cause PxIS.IFS/PxIS.INFS to be set are:

- In the PxSERR.ERR field, the P bit is set to ‘1’
- In the PxSERR.DIAG field, the C or H bit is set to ‘1’
- PhyRdy drops unexpectedly

Examples of these types of errors are below, with the corresponding PxSERR bit that is set if appropriate.

The only difference between PxIS.IFS and PxIS.INFS being set is the type of FIS that is being transmitted/received when the error occurs. If the error occurred during a non-Data FIS, the FIS must be retransmitted, so the error is non-fatal and PxIS.INFS is set. If the error occurred during a Data FIS, the transfer shall stop, so the error is fatal and PxIS.IFS is set.

In the case of a non-Data FIS error, between seeing a non-Data FIS fail and the attempt to re-transmit, the HBA may receive other FISes from the device (this will most likely happen when performing native command queuing commands). When this occurs, the HBA must accept the FIS, perform the correct actions, and then retry the failed FIS.

If the HBA was transmitting a Data FIS it does not retry the FIS and waits for software to clear the PxCMD.ST bit to ‘0’. The HBA shall retransmit a non-Data FIS continuously after a failure until either the transfer succeeds or system software stops the controller by clearing PxCMD.ST to ‘0’.

- **Received Disparity Error / Illegal Character (K28.3):** When a disparity error is encountered, the HBA assumes the character is correct and resets the disparity counter. No error bits are set.
- **Received Disparity Error / Illegal Character (D):** When this occurs, the HBA returns R_ERR at the end of the FIS. It sets PxSERR.DIAG.B. Note that PxIS.IFS/PxIS.INFS shall not be set; the CRC error that is likely to occur due to this event will set the appropriate bit. If there is an illegal

character outside the FIS boundaries, the HBA may ignore the event and is not required to set PxSERR.DIAG.B.

- **PhyRdy Dropping Unexpectedly:** When this occurs, the HBA returns to idle. If the PhyRdy signal dropped during the middle of a command, the HBA may have to be restarted.
- **Calculated Different CRC than Received:** When this occurs, the HBA returns R_ERR and returns to idle. It sets PxSERR.DIAG.C
- **Incorrect FIS or FIS with Illegal Length for Corresponding FIS Type Received:** When this occurs, the HBA returns R_ERR at end of the FIS, shall not post the FIS to memory, and returns to idle. It sets PxSERR.ERR.P. This can only be done for supported FIS types. An unknown FIS is not considered an illegal FIS, unless the length received is more than 64 bytes. If an unknown FIS arrives with length <= 64 bytes, it is posted and the HBA continues normal operation.
- **Internal Buffer Overflow:** This occurs when the HBA sends a HOLD, but a HOLDA was not received quickly enough by the HBA, and the HBA's internal data FIFOs overflow. The HBA returns R_ERR at the end of the FIS. It sets PxSERR.ERR.P.
- **HBA Receives R_ERR:** If the HBA receives an R_ERR to a FIS it was transmitting, it sets PxSERR.DIAG.H.
- **FIS received from a device, where both BSY and DRQ are to be updated in the Status register and both PxTFD.STS.BSY and PxTFD.STS.DRQ are cleared:** When this occurs, the HBA returns R_OK, does not set any error bits, and does not update any registers or the received FIS area based on the received FIS (i.e. the FIS is ignored). No error bits are set because this is a valid occurrence when enumerating devices on a Port Multiplier.

It is system software's responsibility to check the PxSERR register periodically to determine if the interface is operating cleanly, and take appropriate actions (such as going down to Generation1 speed if operating at a higher speed or notifying the user) when interface errors occur.

Note that when an error such as a bad CRC or an illegal length occurs, the HBA cannot trust the incoming FIS to be correct. For example, the HBA may have thought the FIS was a D2H Register FIS, but if the CRC is incorrect, it could be because the type specified in the FIS is incorrect.

To address this problem, the HBA shall not update its internal registers, nor update the FIS received area, until it determines that a non-Data FIS it received was valid. If the HBA believes the FIS to be a Data FIS, however, it may copy it to memory at the appropriate PRD location. This is because Data FISes may be as long as 8KB. An HBA may set other error or diagnostic bits based on the FIS contents when there is a CRC error; these bits may not be completely accurate due to the CRC error. It is possible that a fatal error is generated due to a non-fatal CRC error. In this case, software should perform the normal procedure to recover from a fatal error.

6.1.3 Port Multiplier Errors

When a Port Multiplier is connected, if a FIS is received from a device, where the PMP field does not match what is expected, the HBA returns R_OK and sets PxIS.IPMS. The HBA shall discard the packet and not update any registers or memory structures based on the FIS contents. In command-based switching, this indicates that the only active PMP port was not properly returned. In FIS based switching, this indicates that a PMP field was returned that is not in the list of active PMPs.

6.1.4 Device Errors

When a FIS arrives that updates the taskfile, the HBA checks to see if PxTFD.STS.ERR is set. If it is, and PxIE.TFEE is set, the HBA shall generate an interrupt and stop processing any more commands.

6.1.5 Command List Overflow

Command list overflow is defined as software building a command table that has fewer total bytes than the transaction given to the device. On device writes, the HBA will run out of data, and on reads, there will be no room to put the data.

For an overflow on data read, either PIO or DMA, the HBA shall set PxIS.OFS, and if enabled via PxIE.OFE and GHC.IE, generate an interrupt. When this condition occurs on data reads, the HBA shall make a best effort to continue, however the HBA may not be able to recover without software

intervention. Overflow is a serious error, thus software should perform a fatal error recovery procedure to ensure that the HBA is brought back to a known condition before continuing. For an overflow on writes, the HBA may transmit HOLDs to the device since it does not have any data to satisfy the request size; a COMRESET is required by software to clean up from this serious error. For an overflow on data writes with DMA, the HBA does not know there is more data until it receives the next DMA Activate. When this occurs, it may optionally set PxIS.OFS and attempt to terminate the transfer. However, this is a fatal condition, and an HBA is allowed to hang on the transfer. For PIO writes, the HBA receives the PIO Setup FIS and therefore knows the length, and therefore may optionally set PxIS.OFS. However, by not satisfying the length, the transfer shall end in an error, and software must recover. Therefore setting PxIS.OFS is optional for both DMA and PIO data write conditions. Detecting overflow and setting PxIS.OFS on native command queuing commands is optional.

6.1.6 Command List Underflow

Command list underflow is defined as software building a command table that has more total bytes than the transaction given to the device.

For data writes, both PIO and DMA, the device shall detect an error and end the transfer. These errors are most likely going to be fatal errors that will cause the port to be restarted. For data reads, the HBA shall update its PRD byte count with the total number of bytes received from the last FIS, and may be able to continue normally, but is not required to.

The HBA is not required to detect underflow conditions for native command queuing commands.

6.1.7 Native Command Queuing Tag Errors

The HBA does not actively check incoming DMA Setup FISes to ensure that the PxSACT register bit for that slot is set.

The reason for this is if the device gives an incorrect tag, it could just as likely be for a tag that is active. In this case, the HBA would see no error, although the data transfer that occurs is incorrect. Therefore, there is little benefit in the HBA checking for inactive tags. Just as in the wrong active tag case, the data transfer that occurs will be incorrect.

Existing error mechanisms, such as host bus failure, or bad protocol, are used to recover from this case.

6.1.8 PIO Data Transfer Errors

In accordance with Serial ATA 1.0a, Data FISes prior to the final Data FIS must be an integral number of Dwords. If the HBA receives an intermediate Data FIS transfer request that is not an integral number of Dwords, the HBA shall set PxSERR.ERR.P to ‘1’, set PxIS.IFS to ‘1’ and stop running until software restarts the port.

The HBA shall ensure that the size of the Data FIS received during a PIO command matches the size in the Transfer Count field of the preceding PIO Setup FIS. If the Data FIS size does not match the Transfer Count field in the preceding PIO Setup, the HBA shall respond with R_ERR to the Data FIS, set PxSERR.ERR.P to ‘1’, set PxIS.IFS to ‘1’, and then stop running until software restarts the port.

6.2 Error Recovery

6.2.1 HBA Aborting a Transfer

When the HBA detects an error that it cannot recover from, it may need to end the transfer on the SATA interface.

To do this, the HBA asserts SYNC Escape to stop the bad FIS, and when the device is quiescent, returns to idle. The SATA device should send a D2H Register FIS at this point, with the ERR bit set to indicate an error in the transfer.

When aborting a transfer, the HBA does not wait for the D2H Register FIS before proceeding with error recovery (such as setting interrupt status bits and generating interrupts). This is because a device may be in a hung condition and cannot generate the D2H Register FIS.

6.2.2 Software Error Recovery

When an interrupt is generated due to an error condition, software will attempt to recover. Fatal errors (signified by the setting of PxIS.HBFS, PxIS.HBDS, PxIS.IFS, or PxIS.TFES) will cause the HBA to enter the ERR:Fatal state, and clear PxCMD.CR. In this state, the HBA shall not issue any new commands nor acknowledge DMA Setup FISes to process any native command queuing commands. To recover, the port must be restarted; the port is restarted by clearing PxCMD.ST to '0' and then setting PxCMD.ST to '1'. For non-fatal errors (signified by the setting of PxIS.INFS or PxIS.OFS) the HBA continues to operate.

If the transfer was aborted (see section 6.2.1), the device is expected to send a D2H Register FIS with PxTFD.STS.ERR set to '1' and both PxTFD.STS.BSY and PxTFD.STS.DRQ cleared to '0'. Under this scenario, system software knows that the device is in a stable state and transfers may be restarted without issuing a COMRESET to the device.

For fatal errors, software must determine which commands were not processed and either re-issue them or notify higher level software that the command failed. The steps involved are listed in the following sections.

To detect an error that requires software recovery actions to be performed, software should check whether any of the following status bits are set on an interrupt: PxIS.HBFS, PxIS.HBDS, PxIS.IFS, and PxIS.TFES. If any of these bits are set, software should perform the appropriate error recovery actions based on whether non-queued commands were being issued or native command queuing commands were being issued.

6.2.2.1 Non-Queued Error Recovery

The flow for system software to recover from an error when non-queued commands are issued is as follows:

- Reads PxCI to see which commands are still outstanding
- Reads PxCMD.CCS to determine the slot that the HBA was processing when the error occurred
- Clears PxCMD.ST to '0' to reset the PxCI register, waits for PxCMD.CR to clear to '0'
- Clears any error bits in PxSERR to enable capturing new errors.
- Clears status bits in PxIS as appropriate
- If PxTFD.STS.BSY or PxTFD.STS.DRQ is set to '1', issue a COMRESET to the device to put it in an idle state
- Sets PxCMD.ST to '1' to enable issuing new commands
- Optionally issue a command to gather information about the error, for example READ LOG EXT, if software did not have to perform a device reset (COMRESET or software reset) as part of the error recovery.

Software then either completes the command that had the error and commands still outstanding with error to higher level software, or re-issues these commands to the device.

6.2.2.2 Native Command Queuing Error Recovery

The flow for system software to recover from an error when native command queuing commands are issued is as follows:

- Reads PxSACT to see which commands have not yet completed
- Clears PxCMD.ST to '0' to reset the PxCI and PxSACT registers, waits for PxCMD.CR to clear to '0'
- Clears any error bits in PxSERR to enable capturing new errors.
- Clears status bits in PxIS as appropriate
- If PxTFD.STS.BSY or PxTFD.STS.DRQ is set to '1', issue a COMRESET to the device to put it in an idle state
- Sets PxCMD.ST to '1' to enable issuing new commands
- Issue READ LOG EXT to determine the cause of the error condition if software did not have to perform a device reset (COMRESET or software reset) as part of the error recovery

Software then either completes commands that did not finish with error to higher level software, or re-issues them to the device.

6.2.2.3 Recovery of Unsolicited COMINIT

An unsolicited COMINIT is a COMINIT that is not received as a consequence of issuing a COMRESET to the device (refer to Serial ATA II: Extensions to Serial ATA 1.0a revision 1.1). If the HBA receives an unsolicited COMINIT during normal operation, the HBA shall perform the following actions:

- Respond to the device with a COMRESET
- Halt execution until PxIS.PCS is cleared to '0' by software

To detect this condition, software should check whether PxIS.PCS is set to '1' on an interrupt. The HBA cannot guarantee that a device received a COMRESET because a COMINIT may appear to be solicited to the HBA if it happens to occur closely to an issued COMRESET. Therefore, when software detects that PxIS.PCS is set, software should first issue a COMRESET to ensure that the device receives a COMRESET. Then software should perform the appropriate actions to clear PxIS.PCS to '0'. To recover, software should perform error recovery actions for a fatal error condition (including restarting the controller). Then software should perform a re-enumeration to check whether a new device has been inserted.

7 Hot Plug Operation

7.1 Platforms that Support Cold Presence Detect

For platforms that support cold-presence detect, additional logic is required on the board that implements the SATA ports. How this logic is implemented is beyond the scope of this specification.

7.1.1 Device Hot Unplugged

When a powered-down device is removed, the HBA shall be notified through an external pin for that port going to a logical '0'. The HBA set PxIS.CPDS to indicate the device changed state. If PxIE.CPDE and GHC.IE are set, the HBA shall also generate an interrupt. Software should then check PxCMD.CPS to determine whether a device is present.

7.1.2 Device Hot Plugged

When a device is added, the HBA shall be notified through an external pin for that port going to a logical '1'. The HBA shall report that the device status changed by setting PxIS.CPDS to indicate the device changed state. If PxIE.CPDE and GHC.IE are set, the HBA shall also generate an interrupt. Software should then check PxCMD.CPS to determine whether a device is present.

7.2 Platforms that Support Interlock Switches.

For platforms that support interlock switches, additional logic is required on the board that implements the SATA ports. How this logic is implemented is beyond the scope of this specification. The net result of the logic, though, is that whenever the logic changes state, a high logic level will be sent to the HBA, which shall set PxIS.DIS.

Setting of this bit in and of itself does not imply a hot plug or unplug event, but is a notification to software that the device state may have changed.

7.3 Native Hot Plug Support

The HBA must support native hot plug. Hot plug insertion is detected by reception of a COMINIT signal from the device. Hot plug removal is detected by a change in the state of the HBA's internal PhyRdy signal.

On reception of an unsolicited COMINIT, the HBA shall generate a COMRESET. If the COMINIT is in response to a COMRESET, then the HBA shall begin the normal communication negotiation sequence as outlined in the Serial ATA 1.0a specification. When a COMRESET is sent to the device the PxSSTS.DET field shall be cleared to 0h. When a COMINIT is received, the PxSSTS.DET field shall be set to 1h. When the communication negotiation sequence is complete and PhyRdy is true the PxSSTS.DET field shall be set to 3h.

The HBA shall set the PxSERR.DIAG.X bit to '1' when a COMINIT is received from the device. Hot plug insertions are detected via the PxIS.PCS bit that directly reflects the PxSERR.DIAG.X bit. The HBA shall set the PxSERR.DIAG.N bit to '1' when the HBA's internal PhyRdy signal changes state. Hot plug removals are detected via the PxIS.PRCS bit that directly reflects the PxSERR.DIAG.N bit. Note that PxSERR.DIAG.N is also set to '1' on insertions and during interface power management entry/exit.

When PhyRdy transitions from '1' to '0', the HBA may update PxSSTS.DET to 1h since the reason for the loss of communications is not fully known.

7.3.1 Hot Plug Removal Detection and Power Management Interaction (Informative)

PhyRdy changes state when a device is inserted, when a device is disconnected, and when the link enters/exits a Partial or Slumber interface power management state. Thus, if interface power management is enabled for a port, the PxSERR.DIAG.N bit may be set due to the link entering the Partial or Slumber power management state, rather than due to a hot plug insertion or removal event.

To reliably get removal detection notification via the PxSERR.DIAG.N bit, interface power management must be disabled for the port. When the Serial ATA link is in a Partial or Slumber interface power

management state, device removals cannot be detected and reported via the PxSERR.DIAG.N bit because there is no active signal on the link.

7.3.1.1 Software Flow for Hot Plug Removal Detection (Informative)

To reliably detect hot plug removals, software must disable interface power management. Software should perform the following initialization on a port after a device is attached:

- Set PxSCTL.IPM to 3h to disable interface power management state transitions.
- Set PxCMD.ALPE to '0' to disable aggressive power management.
- Ensure PxIE.PRCE is set to '1' to enable interrupts on hot plug removals.
- Disable device initiated interface power management by issuing the appropriate SET FEATURES command.

During normal operation, software should check PxIS.PRCS to determine if a hot plug removal has occurred.

7.3.1.2 Software Flow for Power Management (Informative)

To utilize power management on a port and avoid spurious interrupts due to PhyRdy transitions, software should disable the interrupt for hot plug removal. Software should perform the following initialization on a port after a device is attached:

- Clear PxIE.PRCE to '0' to disable interrupts on PhyRdy state changes.
- Set PxSCTL.IPM to 0h to enable interface power management state transitions.
- Set PxCMD.ALPE and PxCMD.ASP appropriately based on the aggressive power management policy.

During normal operation, software should check that PxSSTS.DET is set to 3h to ensure that a hot plug removal has not occurred.

7.4 Interaction of the Command List and Port Change Status

Software may have one or several commands in the command list at the time a device is unplugged. A new device may also be inserted before software has had an opportunity to see the change.

When a hot plug event occurs, software will normally stop outstanding commands to the device that was removed and begin issuing commands to the new device. To accomplish this, an HBA shall stop transferring data, return to an idle condition, and clear PxCMD.CR whenever PxIS.PCS is set. This allows software to see what commands are outstanding by checking PxCI, PxSACT, and PxCMD.CCS. Once software has determined which commands need to be re-issued, it shall clear PxCMD.ST and restart the controller by setting PxCMD.ST and taking any necessary actions to enable the SATA device.

8 Power Management Operation

8.1 Introduction

This section covers power management of the HBA and the Serial ATA interface. This specification does not cover any power management that a Serial ATA device may do internally that is transparent to the interface.

8.2 Power State Mappings

The PCI specification defines power management states for devices, which shall be applied to the HBA. They are:

- D0 – Working (required)
- D1 – Not defined for storage HBAs
- D2 – Not defined for storage HBAs.
- D3 – Very deep sleep (required). This state is split into two sub-states, $D3_{HOT}$ (can respond to PCI configuration accesses) and $D3_{COLD}$ (cannot respond to PCI configuration accesses). These two sub-states are considered the same, where $D3_{HOT}$ has V_{CC}, but $D3_{COLD}$ does not. PxCMD.ST must be cleared to '0' before entering the D3 power state.

Serial ATA devices may also have multiple power states. Each of these device states are subsets of the HBA's D0 state. They are:

- D0 – Device is working and instantly available.
- D1 – Device enters when it receives a STANDBY IMMEDIATE command. Exit latency from this state is in seconds.
- D2 – Not currently defined for Serial ATA devices.
- D3 – Device enters when it receives a SLEEP command. Exit latency from this state is in seconds.

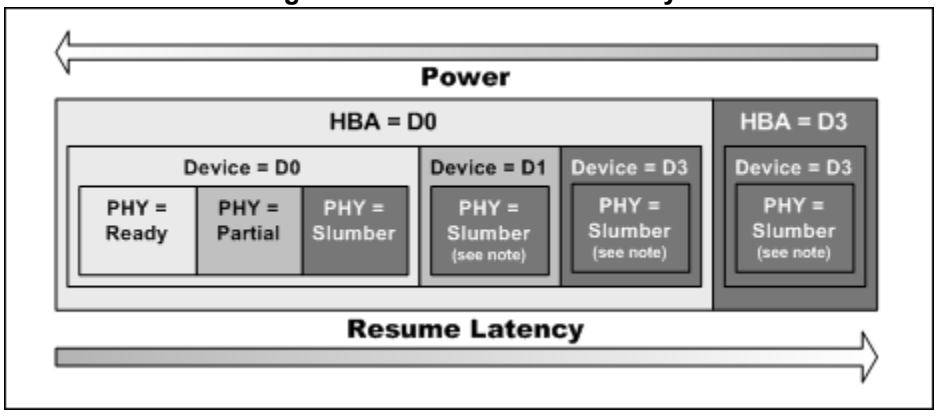
Finally, Serial ATA defines three Phy layer (or interface) power states. They are:

- Phy Ready – Phy logic and PLL are both on and active
- Partial – Phy logic is powered, but in a reduced state. Exit latency is no longer than 10µs
- Slumber – Phy logic is powered, but in a reduced state. Exit latency can be up to 10ms.

Since these states have much lower exit latency than the D1 and D3 states, AHCI defines these states as sub-states of the device D0 state.

The following picture gives a hierarchical view of power states of Serial ATA.

Figure 17: Power State Hierarchy



Note: The Phy is not required to be in a Slumber state when the device is in a D1 or D3 state, nor is it required to be in a Slumber state when the HBA is in a D3 state. While this may be the likely condition of the interface when the devices connected to the interface are in a low power state, it is not a requirement, and the interface shall break out of these states on a power management event.

8.3 Power State Transitions

8.3.1 Interface Power Management

The Serial ATA 1.0a specification defines two lower power interface power management states, Partial and Slumber, in order to save power on the Serial ATA link in power sensitive systems. The Partial and Slumber interface power management states can be initiated by software, the HBA itself, or by the device. The interface power management state is negotiated between the host and the device on the interface using Serial ATA primitives. Any request can be accepted (using the PMACK primitive) or rejected (using PMNACK primitives) based upon current conditions and settings in the device and HBA. The current interface power management state is reflected to software in PxSSTS.IPM.

8.3.1.1 Device Initiated

By default, a device that supports initiating interface power management states has the capability disabled. To enable this feature, the appropriate SET FEATURES command may be issued to the device. The HBA shall respond to device initiated power management requests as specified by PxSCTL.IPM. A request from the device to enter an interface power management state may be rejected by the HBA if the HBA needs to transmit a FIS to the device.

8.3.1.2 System Software Initiated

PxCMD.ICC is used by system software to initiate interface power management state transitions. The request to transition to a different interface power management state shall only be acted on by the HBA if the Link layer is currently in the L_IDLE state. If the HBA's Link layer is not in the L_IDLE state when the PxCMD.ICC field is written, the request shall be ignored. The HBA shall not perform a transition directly from Partial to Slumber or from Slumber to Partial based on a new value being written to PxCMD.ICC. If the link is currently in a Partial or Slumber interface power management state, it is software's responsibility to bring the link to the active state before requesting a transition to a different interface power management state. The time from the request written to PxCMD.ICC until the link is active is bounded by the maximum recovery times from Partial or Slumber as outlined in the Serial ATA 1.0a specification.

8.3.1.3 HBA Initiated

The HBA may implement aggressive power management, as indicated in CAP.SALP. Aggressive power management allows the HBA to initiate an interface power management state as soon as there are no commands outstanding to the device. This enables immediate entry into the low power interface state without waiting for software in power sensitive systems. The PxCMD.ALPE bit defines whether the feature is enabled and the PxCMD.ASP field controls whether Partial or Slumber is initiated by the HBA when enabled.

When PxCMD.ALPE is set to '1', if the HBA recognizes that there are no commands to process, the HBA shall initiate a transition to Partial or Slumber interface power management state based upon the setting of PxCMD.ASP. The HBA recognizes no commands to transmit as either:

- PxSACT is set to 0h, and the HBA updates PxCI from a non-zero value to 0h.
- PxCI is set to 0h, and a Set Device Bits FIS is received that updates PxSACT from a non-zero value to 0h.

If the PxSACT and PxCI registers are both cleared to 0h, and the interface is in an active state, the HBA shall not initiate placing the interface into a lower power state, unless PxCMD.ICC is written with an appropriate value.

Before performing a FIS transmission, the HBA must ensure the link is in the active state. If the link is in the Partial or Slumber interface power management state, a COMWAKE must be issued, and the HBA must wait until the link is active before proceeding with transmission of the FIS.

8.3.1.4 Software Requirements and Precedence

Software must check CAP.SSC (Slumber capable) and CAP.PSC (Partial capable) to determine if the HBA supports interface power management transitions as an initiator or a target. If an interface power management state is not supported, then software shall not write the PxCMD.ICC field nor set the

aggressive power management capability to initiate a transition to that state. Software must set the PxSCTL.IPM field to disable transition to any unsupported interface power management state. If CAP.SSC or CAP.PSC is cleared to '0', software should disable device-initiated power management by issuing the appropriate SET FEATURES command to the device.

The Serial ATA 1.0a specification defines the proper behavior of the link layer when a host initiated and device initiated power state transition occur concurrently. HBA initiated interface power management requests are higher priority than software initiated requests. Thus if the HBA and software request transitions to different states at the same time, the HBA's request shall take precedence over the software request.

8.3.2 Device D1, D3 States

The D1 and D3 device states are entered when system software has determined that no commands will be sent to the device for some time. To enter these states, software may perform two actions. The first is to issue a command to the device to enter the low power state (STANDY IMMEDIATE for D1, SLEEP for D3), and the second step is to put the interface into a Slumber power management state (by setting PxCMD.ICC to 6h).

Note: It is recommended that the device initiate a Slumber power management state when it receives a command to enter the D1 or D3 state.

8.3.3 HBA D3 state

After the interface and device have been put into a low power state, the HBA may be put into a low power state. This is performed via the PCI power management registers in configuration space. AHCI only supports the D3 state.

There are two very important aspects to note when using PCI power management.

- When the power state is D3, only accesses to configuration space are allowed. Any attempt to access the register memory space must result in master abort.
- When the power state is D3, no interrupts may be generated, even if they are enabled. If an interrupt status bit is pending when the controller transitions to D0, an interrupt may be generated.

Software must disable interrupts (GHC.IE must be cleared to '0') prior to requesting a transition of the HBA to the D3 state. This precaution by software avoids an interrupt storm if an interrupt occurs during the transition to the D3 state.

PxCMD.ST must be cleared to '0' before entry into the D3 power state.

8.4 PME

When the HBA is in the D3 state, it may optionally wake based on a change in the device state.

PME must be generated when the HBA is in the D3 state under the following conditions:

- PxIS.PCS is set to '1' due to a native hot plug insertion
- PxIS.DIS set, indicating an interlock switch has been opened or closed
- PxIS.CPDS set, indicating cold presence detect state change
- Set Device Bits FIS received on the interface with the 'I' bit set to '1'

If any of these bits are set, regardless of the setting of the enables in PxIE and GHC.IE, the HBA shall generate PME#.

9 Port Multiplier Support

Port Multiplier support for HBAs is optional, and its support is indicated to system software via CAP.SPM. If Port Multipliers are supported in the HBA, it must support command-based switching. This section describes the hardware and software differences necessary to make a Port Multiplier work.

9.1 Command Based Switching

In this mode of operation, a communication path is opened between the HBA and a device through the Port Multiplier. Since Port Multipliers are meant to be simple, the burden of making a connection is on AHCI software, to ensure that multiple commands are not outstanding to different devices behind the Port Multiplier.

9.1.1 Non-Queued Operation

When running non-queued commands, the command list may be filled with any combination of ports, and each command list entry can target a different port. There is no fixed relationship between number of commands allocated to the device and the number of devices behind a Port Multiplier. For example, 29 commands could be allocated to the device behind PM port #0,and 3 commands for the remaining devices, if that is desired by system software.

Since the commands are non-queued, the HBA shall execute each command entry in its entirety before moving to the next entry in the command list, which may include a command to a different port.

This places special burden on system software for building commands. Since the HBA operates in order in its command list, system software should not fill the list in sequential order with commands to a single device; otherwise another device may get starved. Take the following example:

- 4 devices attached to a PM behind a port
- The operating system presents several commands
    - 4 to the device behind PM port #0
    - 2 to the device behind PM port #1
    - 1 to the device behind PM port #2

If the AHCI software placed these commands in sequential order, starting at command slot 0, the list would look as follows:

- Slot #0: PM Port #0, Command 1
- Slot #1: PM Port #0, Command 2
- Slot #2: PM Port #0, Command 3
- Slot #3: PM Port #0, Command 4
- Slot #4: PM Port #1, Command 1
- Slot #5: PM Port #1, Command 2
- Slot #6: PM Port #2, Command 1

This would cause the device behind port #2 to not execute its command for a very long time. In the worst case scenario, 31 commands to other devices behind different PM ports may execute before this command is allowed to execute.

A better placement of commands to result in fairer execution would be as follows:

- Slot #0: PM Port #0, Command 1
- Slot #1: PM Port #1, Command 1
- Slot #2: PM Port #2, Command 1
- Slot #8: PM Port #0, Command 2
- Slot #9: PM Port #1, Command 2
- Slot #16: PM Port #0, Command 3
- Slot #24: PM Port #0, Command 4

Mapping commands in this fashion helps ensure that the device behind PM port #2 has a fairer chance of executing. The above algorithm results in round-robin execution of commands. Many algorithms for fairness exist, and their implementations are beyond the scope of this specification.

In order to find the best slot for placing the next command, software should use PxCMD.CCS to find the current slot the HBA is executing, and place the command in an appropriate slot for the fairness algorithm that is desired.

9.1.2 Queued Operation

Since queued commands result in two different operations (command issue, clear of BSY, then data transfer), if commands were sent to different ports, the Port Multiplier may issue FISes back to the HBA in an interleaved manner from different ports. This will break an HBA that only supports command-based switching. Therefore, when executing native command queuing commands, system software must only add commands to the command list that target a single port behind the Port Multiplier, wait for the commands to finish (PxSACT bits all cleared), then add commands for a different port. Additionally, the tags used must match the command slot entries.

9.2 Port Multiplier Enumeration

In order to enumerate a Port Multiplier, a software reset is issued to port 0Fh (control port) on the Port Multiplier. If the signature returned corresponds to a Port Multiplier, then a Port Multiplier is attached. If the signature returned corresponds to another device type, then a Port Multiplier is not attached.

To reliably enumerate the Port Multiplier, regardless of the presence of a device on Port Multiplier device port 0, the PxCMD.CLO (command list override) bit should be used if this feature is supported by the HBA (indicated by CAP.SCLO being set to ‘1’). Software should ensure that the PxCMD.ST bit is ‘0’. Then software should construct the two Register FISes required for a software reset in the command list, where the PM Port field value in the Register FIS is set to 0Fh. After constructing the FISes in the command list, software should set PxCMD.CLO to ‘1’ to force the BSY and DRQ bits in the Status register to be cleared. Then software should set the PxCMD.ST bit to ‘1’ and set appropriate PxCI bits in order to begin execution of the software reset command.

If the CAP.SCLO bit is cleared to ‘0’, a Port Multiplier can only be enumerated after a device on Port Multiplier device port 0 sends a Register FIS to the host that clears PxTFD.STS.BSY and PxTFD.STS.DRQ to ‘0’.

10 Platform Communication

10.1 Software Initialization of HBA

Before any useful work can be performed, the HBA must be initialized. Initialization consists of two independent phases: a firmware phase (platform BIOS) and a system software phase. Initialization of the HBA's PCI configuration space is beyond the scope of this specification.

10.1.1 Firmware Specific Initialization

To aid system software during runtime, the BIOS shall ensure that the following registers are initialized to values that are reflective of the capabilities supported by the platform. Firmware shall always initialize the following registers and values:

- CAP.SSS (support for staggered spin-up)
- CAP.SIS (support for interlocked switches)
- PI (ports implemented)
- PxCMD.HPCP (whether port is hot plug capable). Firmware shall initialize the HPCP bit for each port implemented on the platform (as defined by the PI register). The PxCMD.HPCP should be set to '1' if PxCMD.ISP or PxCMD.CPD is set to '1' for the port.
- PxCMD.ISP (whether interlocked switch is attached to the port). Firmware shall initialize the ISP bit for each port implemented on the platform (as defined by the PI register).
- PxCMD.CPD (whether cold presence detect logic is attached to the port). Firmware shall initialize the CPD bit for each port implemented on the platform (as defined by the PI register).

After firmware has initialized the aforementioned registers, it shall then perform the following steps to complete the staggered spin-up process (if applicable to the platform) on each port implemented by the AHCI HBA (as indicated by the PI register):

1. Indicate that system software is AHCI aware by setting GHC.AE to '1'.
2. Ensure that PxCMD.ST = '0', PxCMD.CR = '0', PxCMD.FRE = '0', PxCMD.FR = '0', and PxSCTL.DET = '0h'.
3. Allocate memory for the command list and the FIS receive area. Set PxCLB and PxCLBU to the physical address of the allocated command list. Set PxFB and PxFBU to the physical address of the allocated FIS receive area. Then set PxCMD.FRE to '1'.
4. Initiate a spin up of the SATA drive attached to the port; i.e. set PxCMD.SUD to '1'.
5. Wait for a positive indication that a device is attached to the port (the maximum amount of time to wait for presence indication is specified in the Serial ATA 1.0a specification). This is done by polling PxSSTS.DET. If PxSSTS.DET returns a value of 1h or 3h when read, then system software shall continue to the next step, otherwise if the polling process times out system software moves to the next implemented port and returns to step 1.
6. Clear the PxSERR register, by writing '1s' to each implemented bit location.
7. Wait for indication that SATA drive is ready. This is determined via an examination of PxTFD.STS. If PxTFD.STS.BSY, PxTFD.STS.DRQ, and PxTFD.STS.ERR are all '0', prior to the maximum allowed time as specified in the ATA/ATAPI-6 specification, the device is ready.

10.1.2 System Software Specific Initialization

This section describes how system software places the AHCI HBA into a *minimally* initialized state. Because the term *system software* is used to collectively reference both the platform BIOS and operating system, this section applies to any software that is AHCI aware. Note that some steps may not be required for system BIOS (e.g. ensuring that the controller is not running) since the host controller will be in a known state following a system reset. Additionally, if system BIOS already allocated memory for and initialized the appropriate registers for the command list and FIS receive area, it may skip this step of the process.

To place the AHCI HBA into a minimally initialized state, system software shall:

1. Indicate that system software is AHCI aware by setting GHC.AE to ‘1’.
2. Determine which ports are implemented by the HBA, by reading the PI register. This bit map value will aid software in determining how many ports are available and which port registers need to be initialized.
3. Ensure that the controller is not in the running state by reading and examining each implemented port’s PxCMD register. If PxCMD.ST, PxCMD.CR, PxCMD.FRE and PxCMD.FR are all cleared, the port is in an idle state. Otherwise, the port is not idle and should be placed in the idle state prior to manipulating HBA and port specific registers. System software places a port into the idle state by clearing PxCMD.ST and waiting for PxCMD.CR to return ‘0’ when read. Software should wait at least 500 milliseconds for this to occur. If PxCMD.FRE is set to ‘1’, software should clear it to ‘0’ and wait at least 500 milliseconds for PxCMD.FR to return ‘0’ when read. If PxCMD.CR or PxCMD.FR do not clear to ‘0’ correctly, then software may attempt a port reset or a full HBA reset to recover.
4. Determine how many command slots the HBA supports, by reading CAP.NCS.
5. For each implemented port, system software shall allocate memory for and program:
   - PxCLB and PxCLBU (if CAP.S64A is set to ‘1’)
   - PxFB and PxFBU (if CAP.S64A is set to ‘1’)

   It is good practice for system software to ‘zero-out’ the memory allocated and referenced by PxCLB and PxFB. After setting PxFB and PxFBU to the physical address of the FIS receive area, system software shall set PxCMD.FRE to ‘1’.
6. For each implemented port, clear the PxSERR register, by writing ‘1s’ to each implemented bit location.
7. Determine which events should cause an interrupt, and set each implemented port’s PxIE register with the appropriate enables. To enable the HBA to generate interrupts, system software must also set GHC.IE to a ‘1’.

   Note: Due to the multi-tiered nature of the AHCI HBA’s interrupt architecture, system software must always ensure that the PxIS (clear this first) and IS.IPS (clear this second) registers are cleared to ‘0’ before programming the PxIE and GHC.IE registers. This will prevent any residual bits set in these registers from causing an interrupt to be asserted.

At this point the HBA is in a minimally initialized state. System software may provide additional programming of the GHC register and port PxCMD and PxSCTL registers based on the policies of the operating system and platform – these details are beyond the scope of this specification.

Software shall not set PxCMD.ST to ‘1’ until it is determined that a functional device is present on the port as determined by PxTFD.STS.BSY = ‘0’, PxTFD.STS.DRQ = ‘0’, and PxSSTS.DET = 3h. Note that to enable the PxTFD register to be updated with the initial Register FIS for a port, the PxSERR.DIAG.X bit must be cleared to ‘0’.

10.2 Hardware Prerequisites to Enable/Disable GHC.AE

When a legacy interface is supported in addition to AHCI (CAP.SAM = ‘0’), software may desire to switch from one mode to the other. Actively switching between AHCI and a legacy mode is strongly discouraged due to OS sensitivity to the class code used by the controller. This section lists hardware requirements only for switching between AHCI mode and a legacy mode, it does not comprehend any software considerations.

To switch from legacy mode to AHCI mode, the legacy interface should be in an idle state and there should be no outstanding commands to any devices connected to the controller. At this point, software can begin the sequence for initializing the AHCI controller outlined in section 10.1.

To switch from AHCI mode to legacy mode, the AHCI controller must be brought to an idle state for all ports and there should be no outstanding commands to any devices connected to the controller. The PxCMD.ST should be cleared to ‘0’ for all ports and software should ensure that the PxCMD.CR bit is cleared to ‘0’ for all ports. The PxCMD.FRE bit should be cleared to ‘0’ for all ports and software should

ensure that PxCMD.FR is cleared to '0' for all ports. Then software should clear the GHC.IE bit to '0' to ensure there are no interrupts that occur on the AHCI controller. At this point, software may set the GHC.AE bit to '0'. Commands may be issued at this point to the legacy controller.

10.3 Software Manipulation of Port DMA Engines

Each port contains two major DMA engines. One DMA engine walks through the command list, and is controlled by PxCMD.ST. The second DMA engine copies received FISes into system memory, and is controlled by PxCMD.FRE.

10.3.1 Start (PxCMD.ST)

When PxCMD.ST is set, software is limited in what actions it is allowed to perform on the port (refer to section 5.2.2.2).

- It shall not manipulate PxCMD.POD to power on or off a device through cold presence detect logic (if supported by the HBA).
- It shall not manipulate PxSCTL.DET to change the Phy state
- It shall not manipulate PxCMD.SUD to spin-up the device (if supported by the HBA)

The above actions are only allowed while the HBA is idle, indicated by both PxCMD.ST and PxCMD.CR being equal to '0'. This is noted by the HBA state machine H:NotRunning state. If software performs any of the above actions while the port is not idle (PxCMD.ST or PxCMD.CR set), indeterminate results may occur.

Software shall not write PxCMD.ST to '1' from a '0' until it sees that PxCMD.CR is '0'. Additionally, software shall not set PxCMD.ST to '1' until it is determined that a functional device is present on the port as determined by PxTFD.STS.BSY = '0', PxTFD.STS.DRQ = '0', and PxSSTS.DET = 3h.

Before the HBA enters the low power D3 state, software shall clear PxCMD.ST to '0'.

10.3.2 FIS Receive Enable (PxCMD.FRE)

When PxCMD.FRE is set (causing PxCMD.FR to be set to '1'), the HBA receives FISes from the device and copies them into system memory. When PxCMD.FRE is cleared (causing PxCMD.FR to be cleared to '0'), received FISes are held internally, and if the HBAs internal FIFO is full, further FIS reception is blocked.

Software is allowed to manipulate PxCMD.FRE so that it may move the FIS receive area to a new location. When this bit is cleared to '0', software must first wait for PxCMD.FR to clear to '0', indicating that the DMA engine for FIS reception is in an idle condition.

Software shall not clear this bit while PxCMD.ST remains set to '1'.

Upon HBA reset, this bit is cleared. The D2H register FIS containing the device signature shall be accepted by the HBA, and the signature field updated.

Note that if the HBA stops running due to an error (for example, PxIS.IFS is set to '1'), FISes may not be posted until the PxCMD.ST bit is cleared to '0' to recover from the error.

10.4 Reset

AHCI supports three levels of reset:

- Device – a single device on one of the ports is reset, but the HBA and physical communication remain intact. This is the least intrusive.
- Port – the physical communication between the HBA and device on a port are disabled. This is more intrusive
- HBA – the entire HBA is reset, and all ports are disabled. This is the most intrusive.

When an HBA or port reset occurs, Phy communication shall be re-established with the device through a COMRESET followed by the normal out-of-band communication sequence defined in Serial ATA. At the end of the reset, the device, if working properly, will send a D2H Register FIS, which contains the device

signature. When the HBA receives this FIS, it updates PxTFD.STS and PxTFD.ERR register fields, and updates the PxSIG register with the signature.

10.4.1 Device Reset

Legacy software contained a standard mechanism for generating a reset to a Serial ATA device – setting the SRST (software reset) bit in the Device Control register. Serial ATA has a more robust mechanism called COMRESET, also referred to as port reset. A port reset is the preferred mechanism for error recovery and should be used in place of device reset.

To issue a software reset in AHCI, software builds two H2D Register FISes in the command list. The first Register FIS has the SRST bit set to '1' in the Control field of the Register FIS, the 'C' bit is set to '0' in the Register FIS, and the command table has the CHz[R] (reset) and CHz[C] (clear BSY on R_OK) bits set to '1'. The CHz[R] (reset) bit causes the HBA to perform a SYNC escape if necessary to put the device into an idle condition before sending the software reset. The CHz[C] (clear BSY on R_OK) bit needs to be set for the first Register FIS to clear the BSY bit and proceed to issue the next Register FIS since the device does not send a response to the first Register FIS in a software reset sequence. The second Register FIS has the SRST bit set to '0' in the Control field of the Register FIS, the 'C' bit is set to '0' in the Register FIS, and the command table has the CHz[R] (reset) and CHz[C] (clear BSY on R_OK) bits cleared to '0'. Refer to the Serial ATA 1.0a specification for more information on the software reset FIS sequence.

When issuing a software reset, there should not be other commands in the command list. Before issuing the software reset, software must clear PxCMD.ST, wait for the port to be idle (PxCMD.CR = '0'), and then re-set PxCMD.ST. PxTFD.STS.BSY and PxTFD.STS.DRQ must be cleared prior to issuing the reset. If PxTFD.STS.BSY or PxTFD.STS.DRQ is still set based on the failed command, then a port reset should be attempted.

10.4.2 Port Reset

If a port is not functioning properly after a device reset, software may attempt to re-initialize communication with the port via a COMRESET. It must first clear PxCMD.ST, and wait for PxCMD.CR to clear to '0' before re-initializing communication. However, if PxCMD.CR does not clear within a reasonable time (500 milliseconds), it may assume the interface is in a hung condition and may continue with issuing the port reset.

Software causes a port reset (COMRESET) by writing 1h to the PxSCTL.DET field to invoke a COMRESET on the interface and start a re-establishment of Phy layer communications. Software shall wait at least 1 millisecond before clearing PxSCTL.DET to 0h; this ensures that at least one COMRESET signal is sent over the interface. After clearing PxSCTL.DET to 0h, software should wait for communication to be re-established as indicated by bit 0 of PxSSTS.DET being set to '1'. Then software should write all 1s to the PxSERR register to clear any bits that were set as part of the port reset.

When PxSCTL.DET is set to 1h, the HBA shall reset PxTFD.STS to 7Fh and shall reset PxSSTS.DET to 0h. When PxSCTL.DET is set to 0h, upon receiving a COMINIT from the attached device, PxTFD.STS.BSY shall be set to '1' by the HBA.

10.4.3 HBA Reset

If the HBA becomes unusable for multiple ports, and a device reset or port reset does not correct the problem, software may reset the entire HBA by setting GHC.HR to '1'. When software sets the GHC.HR bit to '1', the HBA shall perform an internal reset action. The bit shall be cleared to '0' by the HBA when the reset is complete. A software write of '0' to GHC.HR shall have no effect. To perform the HBA reset, software sets GHC.HR to '1' and may poll until this bit is read to be '0', at which point software knows that the HBA reset has completed.

If the HBA has not cleared GHC.HR to '0' within 1 second of software setting GHC.HR to '1', the HBA is in a hung or locked state.

When GHC.HR is set to '1', GHC.AE, GHC.IE, the IS register, and all port register fields (except PxFB/PxFBU/PxCLB/PxCLBU) that are not HwInit in the HBA's register memory space are reset. The HBA's configuration space and all other global registers/bits are not affected by setting GHC.HR to '1'. Any HwInit bits in the port specific registers are not affected by setting GHC.HR to '1'. The port specific

registers PxFB, PxFBU, PxCLB, and PxCLBU are not affected by setting GHC.HR to '1'. If the HBA supports staggered spin-up, the PxCMD.SUD bit will be reset to '0'; software is responsible for setting the PxCMD.SUD and PxSCTL.DET fields appropriately such that communication can be established on the Serial ATA link. If the HBA does not support staggered spin-up, the HBA reset shall cause a COMRESET to be sent on the port.

10.5 Interface Speed Support

The HBA indicates the maximum speed it can support via the CAP.ISS register. Software can further limit the speed of a port by manipulating each port's PxSCTL.SPD field to a lower value. If software writes a value that is greater than the value in CAP.ISS, the actual speed negotiated will be limited by CAP.ISS, as shown in the following table:

CAP.ISS	PxSCTL.SPD	Maximum Speed Negotiated
1h	0h	1.5 Gbps
1h	1h	1.5 Gbps
1h	2h	1.5 Gbps
2h	0h	3 Gbps
2h	1h	1.5 Gbps
2h	2h	3 Gbps

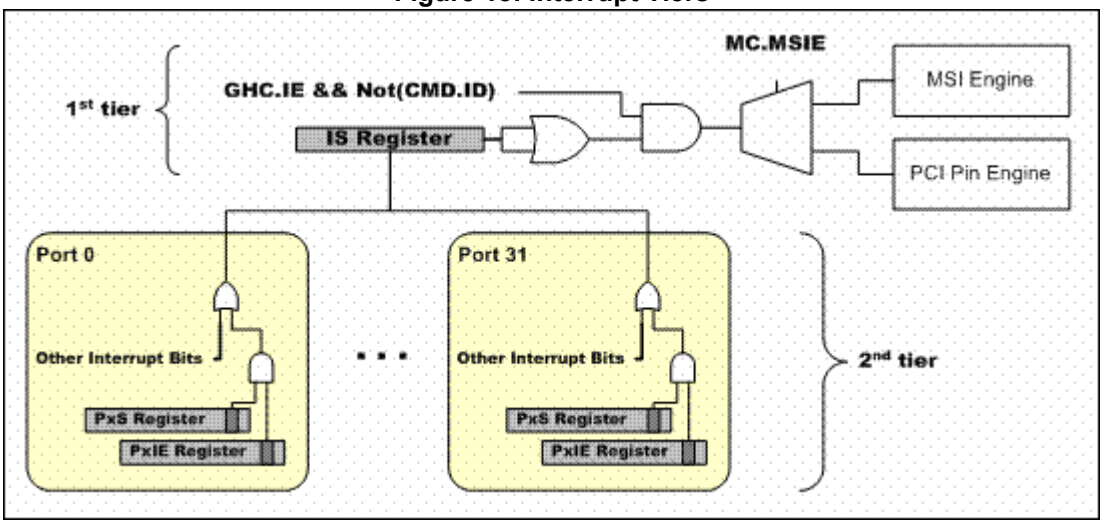
If PxSCTL.SPD is set to a non-zero value when system software loads, platform BIOS (or expansion ROM) decided that the current port required speed limitation. Speed limitation may be set by the platform BIOS to avoid an inordinate number of errors on a port; e.g. a laptop swapbay connector that goes through several intermediate connectors may need to speed limitation for robust operation. If PxSCTL.SPD is set to a non-zero value when system software loads, system software should preserve this setting, including across power management state transitions. It is recommended that platform BIOS (or expansion ROM) not specify a speed limitation unless it is necessary for robust operation. If system software encounters errors and speed could be limited further, system software is allowed to restrict the speed negotiated to a slower rate.

10.6 Interrupts

10.6.1 Tiered Operation

AHCI defines a two-tiered interrupt architecture, which allows for reporting of interrupts to software such that software can handle interrupts through the least amount of programming and status checking. A diagram of the interrupt tiers is shown below:

Figure 18: Interrupt Tiers



10.6.1.1 First Tier (IS Register)

The first tier is identified by the GHC and IS registers.

GHC.IE register enables interrupts for the entire HBA. This is the 'master enable. Until this bit is set, the HBA shall not generate any interrupts. When it is cleared, the HBA may generate interrupts. This bit is only a mask, and does not affect the setting of any of the interrupt status bits in any of the ports. These bits are set regardless of whether or not interrupts are enabled.

The 32-bit IS register reports whether a port has an interrupt pending. This is a bit-mapped register indicating a bit for each of the 32 ports allowed in AHCI. Each bit location can be thought of as reporting a '1' if the virtual "interrupt line" for that port is indicating it wishes to generate an interrupt. That is, if a port has one or more interrupt status bit set, and the enables for those status bits are set, then this bit shall as set. The bits in this register are read/write clear. It is set by the level of the virtual interrupt line being a set, and cleared by a write of '1' from the software.

This register allows software to perform a quick glance of the HBA to see which ports are reporting interrupts. By being read/write clear, it also allows for the implementation of message signaled interrupts.

10.6.1.2 Second Tier (PxIS Registers)

The second tier is identified in each port, through the PxIS (status) and PxIE (interrupt enable) registers. The PxIS register for each port has various interrupt bits

Each one of these interrupts can be individually enabled or disabled for a port, by setting the corresponding bit in PxIE. The status bit in PxIS is always set regardless of the setting of the corresponding PxIE bit.

10.6.2 HBA/SW Interaction

10.6.2.1 Pin Based and Single MSI Message Based Behavior

This is the mode of interrupt operation if any of the following conditions are met:

- MSI is disabled via MSICAP.MC.MSIE
- HBA only supports a single MSI interrupt via the configuration register MSICAP.MC.MMC
- HBA only enabled for a single interrupt via MSICAP.MC.MME
- MSICAP.MC.MME is programmed to a larger value than MSICAP.MC.MMC

In this mode, the IS register determines whether the PCI interrupt line shall be driven active or MSI message shall be sent. The PxIS register contains status information to generate the interrupt. The resulting logical "AND" of the PxIS bit and corresponding PxIE bit results in a "virtual wire" that sets a sticky bit in the IS register. When a level of a virtual wire for a port is '1', the IS register bit for that port shall be set.

The resulting output of the IS register is sent to one of two places. If MSIs are not enabled, the resulting IS bits are logical "ORed" together – any bit being a one causes the PCI interrupt line to be active (electrical '0'). If MSIs are enabled, any change to the IS register that doesn't result in the register being all cleared shall cause an MSI to be sent. Therefore, that in wire mode, a single wire remains active, while in MSI mode, several messages may be sent, as each edge triggered event on a port shall cause a new message, as shown in Table 1.

In order to clear an interrupt, software must first clear the event from the PxIS register, then clear the interrupt event from the IS register. If software clears IS register only, leaving the level of the virtual wire from the PxIS register set, the resulting level of '1' shall cause the IS register bit to be re-set.

Table 1: MSI vs. PCI IRQ Actions

Status of Tier 1 Register	Wire-Mode Action	MSI Action
All bits '0'	Wire inactive	No action
One or more bits set to '1'	Wire active	New message sent
One or more bits set to '1', new bit gets set to '1'	Wire active	New message sent
One or more bits set to '1', software clears some (but not all) bits	Wire active	New message sent
One or more bits set to '1', software clears all bits	Wire inactive	No action

10.6.2.2 Multiple MSI Based Messages

An HBA may optionally support multiple MSI messages for better performance. In this mode, each port has its own interrupt message. To support this mode, the MSICAP.MC.MMC field represents a power-of-2 wrapper on the number of implemented ports in the global memory space PI register. For example, if 3 ports are implemented, then the MSICAP.MC.MMC field must be '010' (4 interrupts).

Multiple-message MSI enables each port to send its own interrupt message, as opposed to a single message for all ports. When enabled for multiple message generation, generation of interrupts is no longer controlled through the IS register. Hardware shall continue to set the IS register just as in the single message case, however the IS register is not used to determine whether to generate an MSI. The IS register may be used by software if fewer than the requested number of messages is granted in order to determine which port had the interrupt. The GHC.IE register shall still be a global interrupt enable/disable for all ports when multiple-message MSI is used.

Each port receives its own interrupt message, up to a maximum of 16 interrupts (16 ports). 16 interrupts is the practical limit of MSI messages in an x86/Windows environment. The mapping of ports to interrupts is done in a 1-1 relationship, up to the maximum number of assigned interrupts. Any ports implemented beyond the maximum allocated messages use the last/maximum interrupt message.

Take the following example in Table 2, where 8 ports are implemented (ports 0 – 7), and 4 messages are allocated (messages 0 – 3). The mapping of ports to interrupts is as follows:

Table 2: Port/MSI Message Mapping, Example 1

Port	Interrupt Message
0	0
1	1
2	2
3	3
4	
5	
6	
7	

Each implemented port maps to an interrupt message of the same value. For example, port 2 maps to message 2, even if port 1 is not implemented. Below is an example, showing 6 ports, where only ports 0, 3, and 5 are implemented:

Table 3: Port/MSI Message Mapping, Example 2

Port	Interrupt Message
0	0
1	1 (unused)
2	2 (unused)
3	3
4	4 (unused)
5	5
-	6 (unused)
-	7 (unused)

In this example, since the HBA had 6 ports, it is required to ask for 8 interrupt messages. In this example, therefore, messages 6 and 7 are also not used.

When generating an MSI message, a port looks at its PxIS register, and uses the following rules to generate a message:

- If a new bit is set in PxIS, and the corresponding bit in PxIE is set, send a message
- If bits are cleared in PxIS, and other bits remain set, if their corresponding bits in PxIE are set, send a message.

If multiple ports share the same MSI message, the rules for generating an MSI must be implemented across all the ports that share that message. For example, in example 1 ports 3-7 share a single message. If all P4IS bits are cleared, but (P3IS & P3IE) is non-zero, a new message is sent.

10.6.3 Disabling Device Interrupts (NIEN Bit in Device Control Register)

The NIEN bit was part of the device control register (legacy I/O addresses 3f6h for primary, 376h for secondary). A legacy HBA must snoop writes to this bit to know whether to mask device interrupts. An AHCI HBA does not snoop this bit. AHCI does not use the NIEN bit; all masking is controlled via the PxIE register. Software should always set the NIEN bit to '0' in all Register FISes transmitted to the device.

10.7 Interlock Switch Operation

In systems that support an interlock switch, the platform contains a mechanical mechanism that acts as an attention indicator or locking mechanism. When this mechanism changes state (such as a button pressed or switch opened/closed), a line changes state. This line is a level driven signal, not edge-triggered.

HBAs that support interlock switches have the following additional features:

- CAP.SIS shall be set to '1'
- PxCMD.ISP shall be set to '1' for each port that has an interlocked switch attached.
- An additional input pin per port on the HBA, the level is reflected in PxCMD.ISS which shows whether the interlock switch is open or closed.

10.8 Cold Presence Detect Operation

In systems that support cold presence detect, the platform board contains voltage comparator logic, as described in the Serial ATA II specification, for recognizing attachments, and FET control to supply power to the device.

HBAs that support cold presence-detect have the following additional features:

- PxCMD.CPD shall be set to '1' for each port that has cold presence detection capability.
- An additional input pin per port on the HBA.
- An additional output pin per port on the HBA to be used as FET control for the power rails to the device.
- For each port, PxCMD.POD shall be read/write.

When an HBA that supports cold presence-detect is first powered-up, no power is supplied to the devices. Initialization software checks the status of the ports, and if a device is connected, sets PxCMD.POD to a '1' to supply power to the port.

10.9 Staggered Spin-up Operation

Staggered spin-up is an optional feature in the Serial ATA II: Extensions to Serial ATA 1.0a revision 1.1 specification. This feature enables an HBA to individually spin-up attached devices. The feature is useful to avoid having a power supply that must handle maximum current draw from all devices at the same time when applying power to the devices. In order for a system to support staggered spin-up, the devices, the HBA, and BIOS/driver must all support staggered spin-up.

In systems that support staggered spin-up, an HBA can individually spin-up devices connected to implemented SATA ports. In systems that do not support staggered spin-up, the HBA spins up all connected SATA devices upon receiving power to the HBA.

Staggered spin-up is only performed when power is applied to the drive. If the drive has been spun-down due to an ATA command (e.g. STANDBY) and the device continues to be powered, the drive will not spin-up again until access to the media is required – the staggered spin-up mechanism is not invoked in this case. Note that when a system transitions to system power management state S3 or greater, the drive will cease having power applied. Since the drive loses power, when the drive is powered back on in the transition back to S0, the drive will undergo another staggered spin-up process.

HBAs that support staggered spin-up shall have the following additional features:

- CAP.SSS shall be set.
- For each port, PxCMD.SUD shall be read/write.

10.9.1 Interaction of PxSCTL.DET and PxCMD.SUD

In systems that support staggered spin-up, there is an interaction between PxSCTL.DET and PxCMD.SUD in controlling the Phy behavior. The two register fields must be set correctly in order to avoid illegal combinations of the two values. In systems that do not support staggered spin-up, PxCMD.SUD is read-only and always set to '1' such that there is no interaction.

The PxSCTL.DET value manipulates the behavior of the Phy. For platforms that support staggered spin-up, PxCMD.SUD also manipulates the behavior of the Phy. The following table describes the interaction.

PxSCTL.DET	PxCMD.SUD	Mode	Behavior
0h	0	Listen	Interface in a reduced power state. • If COMINIT is received then PxSERR.DIAG.X shall be set and no response (OOB signal) shall be sent to the device. • COMWAKE ignored. Software shall only place the port into this state when it believes that no device is connected on the port. In this mode, the HBA forces the Phy into a low power state without requesting a Slumber transition on the link.
0h	0 → 1	Spin-Up	HBA shall send COMRESET, begin initialization sequence
0h	1	Normal	This is the state where the HBA is performing data transfers. • COMINIT received, PxSERR.DIAG.X set, send COMRESET, begin initialization sequence • COMWAKE received, wake the link
1h	0	Illegal	If software programs this condition, indeterminate results may occur.
1h	1	Reset	HBA continuously transmits COMRESET. HBA does not listen for COMINIT. The HBA may optionally set the PxSERR.DIAG.X bit if a COMINIT is received.
1h → 0h	1	Initialize	Stop sending COMRESET, begin initialization sequence.
4h	N/A	Off	Off

Software must only clear PxCMD.SUD if it believes that no device is attached. In Listen Mode (PxSCTL.DET = 0h and PxCMD.SUD = 0), the HBA Phy may enter a reduced power state, equivalent to the power consumed while in the Slumber power management state. The HBA Phy enters this reduced power state without negotiating a transition to Slumber on the link, as asking for a transition to Slumber when no device is attached will fail, and therefore the HBA Phy would stay in a high power state. To avoid this software should ensure that PxSSTS.DET = 0h indicating that no device is present before clearing PxCMD.SUD.

10.9.2 Spin-Up Procedure (Informative)

During initial system power-on in a system that supports staggered spin-up, software needs to spin up each attached device. In addition, when the system transitions from the S3 or greater system power management state back to S0, the attached drives will also need to be spun up. In order to spin up the devices attached to the HBA, software should perform the procedure outlined in section 10.1.1 for staggered spin-up.

10.9.3 Preparing for Low Power System State (Informative)

Before a system transitions to the S3 or greater power management state, the driver should prepare the drives in the system for a smooth transition to the low power state and eventually back to S0. When a system transitions to state S3 or greater, the drive will cease having power applied. Since the drive loses power, when the drive is powered back on in the transition back to S0 the drive will undergo another staggered spin-up process.

Software should follow the procedure outlined below on each implemented port before the power management transition:

1. If a drive is present, issue either the STANDBY or SLEEP command to the drive.

2. If a drive is present, place the interface into Slumber by setting PxCMD.ICC = 6h.
3. After the interface is no longer in the active power state as specified by PxSSTS.IPM, set PxSCTL.DET = 0h and PxCMD.SUD = 0h to enter listen mode. Note that PxSCTL.DET must be set to 0h before setting PxCMD.SUD to 0h.

This process ensures that staggered spin-up can be used on the system transition back to S0.

10.9.4 When to Enter Listen Mode (Informative)

Listen mode should be entered on a particular port when:

- Staggered spin-up is supported on the platform
- Interlocked switch is not supported on this port
- Cold presence detect is not supported on this port
- No device is attached to this port

In this case, listen mode should be entered to save power on the port while still allowing a hot plug insertion event to be detected via PxSERR.DIAG.X. Listen mode should also be entered to save power when a drive is present on a port and the system is about to enter a low power state, refer to section 10.9.3.

Listen mode should not be used if interlocked switch or cold presence detect is supported on the port. In this instance, the Phy for the port should be placed in the offline mode to save maximum power since any hot plug insertion can be detected using the interlocked switch or cold presence detect interrupts.

10.10 Asynchronous Notification

Asynchronous Notification is a feature in Serial ATA II: Extensions to Serial ATA 1.0 revision 1.1. This feature allows an ATAPI device to send a signal to the host when media is inserted or removed and avoids polling the device for media changes. The signal sent to the host is a Set Device Bits FIS with the ‘I’ (interrupt) and ‘N’ (notification) bits set to ‘1’. Refer to the Extension to Serial ATA 1.0a revision 1.1 specification for more information on device behavior for asynchronous notification and how the device shows support for this feature.

The HBA may support receipt of an asynchronous notification event via the Set Device Bits FIS for a directly attached device. To use asynchronous notification, software should set the PxIS.SDBS bit to enable interrupt notification on a Set Device Bits FIS. When accesses to the ATAPI device are idle, software should place the device in a low power state. When the device has a media change, the device will signal this to the host with a Set Device Bits FIS. In response to receiving a PxIS.SDBS interrupt on an idle port, the host should interrogate the device to determine the cause of the interrupt.

10.11 Activity LED

An HBA optionally drives an output pin that can be connected to an external LED based upon activity of the various ports in the HBA. The intended use of this LED is for desktop and mobile systems that contain only a single LED to indicate device activity. An HBA indicates support for this pin via CAP.SAL.

The intent of this LED output is to replace the DASP functionality that is provided in parallel ATA systems. The DASP logic in the device would light the LED under the following scenarios:

- During boot, soft reset, or device reset: LED would be driven by the slave device on a cable (if present) to indicate slave presence. The LED would remain on until a command was written to the slave device, or 30 seconds, whichever comes first.
- During normal operation, while the BSY bit was set in the task file for ATA commands. It may optionally be on for ATAPI commands (A0h and A1h).

In AHCI, slave devices are not supported – every device is a master. Therefore, the first LED mechanism is not supported by an AHCI HBA. An HBA that supports legacy operation will have to consider this support if it supports master/slave emulation.

To support the second mechanism of LED operation, the HBA shall drive the LED pin active if CAP.SAL is set, and:

- If (PxCI != 0h or PxSACT != 0h) and PxCMD.ATAPI = '0'.
- If (PxCI != 0h or PxSACT != 0h) and PxCMD.ATAPI = '1' and PxCMD.DLAE = '1',

When PxCI and PxSACT are both cleared to 0h the LED shall be driven off.

10.12 BIST

BIST is entered when software builds a BIST FIS in the command list and sets CTBAz[B]. Once a BIST command is placed into the list, SW is not allowed to build any more commands until it clears PxCMD.ST.

Details of how an HBA operates in the test mode are outside the scope of this specification. A series of vendor specific tests may be performed using vendor specific registers (such as those in the HBA's PCI configuration space). The details of BIST are not defined in AHCI to allow vendors the freedom of constructing either very elaborate or very light testing schemes.

The test mode is exited when system software clears PxCMD.ST and writes a value of 1h into PxSCTL.DET. Clearing PxCMD.ST shall put the DMA engines into an idle condition, and writing to PxSCTL.DET shall reset the interface.

11 Informative Appendix

11.1 Enclosure Management Services

An HBA supports enclosure management as specified in the Serial ATA II: Extensions to Serial ATA 1.0a revision 1.1 specification. An HBA that integrates enclosure management logic should allocate one of its 32 ports for use as the enclosure management port. It reports a signature of enclosure management as per that specification, and software uses the command list for this port to execute enclosure management commands per that specification. The enclosure management logic can either be internal or external to the HBA.

While allocating a port for enclosure management reduces the available ports on an HBA to 31 from a maximum of 32, constructing enclosure management services in this way reduces software complexity. Software may talk to an enclosure management processor in the same fashion, regardless of whether that processor is integrated in the HBA, or externally, such as through a Port Multiplier.

11.2 Port Selector Support

A Port Selector may be attached to an HBA port. To control the Port Selector with protocol-based port selection, software should issue COMRESET bursts at the appropriate intervals by using the PxSCTL register appropriately. Controlling the Port Selector with side-band port selection is beyond the scope of this specification.