
# Serial ATA:

# High Speed Serialized AT Attachment

Revision 1.0a
7-January-2003

**APT Technologies, Inc.**

**Dell Computer Corporation**

**Intel Corporation**

**Maxtor Corporation**

**Seagate Technology**

This 1.0a revision of the Serial ATA / High Speed Serialized AT Attachment specification consists of the 1.0 revision of the specification with the following errata incorporated:

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 29, 30, 31, 32, 33, 34, 35, 36, 38, 40

The following errata were withdrawn and were not incorporated into this revision of the specification:

1, 27, 28

Details on individual errata can be downloaded from the Serial ATA Working Group website at www.serialata.org.

This 1.0a revision of the Serial ATA / High Speed Serialized AT Attachment specification ("Final Specification") is available for product design. Product implementations should ensure compliance with this specification.

SPECIFICATION DISCLAIMER

THIS SPECIFICATION IS PROVIDED TO YOU "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE AUTHORS OF THIS SPECIFICATION DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO USE OR IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. THE AUTHORS DO NOT WARRANT OR REPRESENT THAT SUCH USE WILL NOT INFRINGE SUCH RIGHTS. THE PROVISION OF THIS SPECIFICATION TO YOU DOES NOT PROVIDE YOU WITH ANY LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS.



For more information about Serial ATA, refer to the Serial ATA Working Group website at www.serialata.org.



Serial ATA Workgroup Technical Editor:

Klaus-Peter Deyring
APT Technologies, Inc.
1347 Pacific Avenue
Suite 205
Santa Cruz, CA 95060 USA
Tel: 831-429-7262
Fax: 831-429-7272
Email: pete@apt-tech.com

## Serial ATA Workgroup Promoters

Klaus-Peter Deyring
(831) 429-7262
pete@apt-tech.com

APT Technologies, Inc.
1347 Pacific Avenue
Suite 205
Santa Cruz, CA 95060

Tom Pratt
(512) 723-8549
tom_pratt@dell.com

Dell Computer Corp.
One Dell Way
Round Rock, TX 78682

Knut Grimsrud
(503) 264-8419
knut.s.grimsrud@intel.com

Intel Corp.
M/S JF2-53
2111 NE 25th Ave
Hillsboro, OR 97124

Farbod Falakfarsa
(408) 894-4066
Farbod.Falakfarsa@maxtor.com

PTD Architecture Group
500 McCarthy Blvd
Milpitas, CA 95035

Marc Noblitt
I/O Planning manager
(720) 684-1333 phone
(720) 684-1031 fax
marc_noblitt@seagate.com

Seagate Technology
389 Disc Drive
Longmont, CO 80503

## TABLE OF CONTENTS

# TABLE OF FIGURES

# TABLE OF TABLES

# 1 Scope

The purpose of this document is to provide a technical specification of a high-speed serialized ATA data link interface. Normative information is provided to allow a technical design team to construct a device that will operate correctly with another device designed to this specification. Informative information is provided to help illustrate the normative material.

# 2 Goals, objectives and migration considerations

## 2.1 Goals and objectives

Serial ATA is defined with the following goals and requirements listed in no particular order:

- Primary inside-the-box storage connection (no outside the box)
- Completely SW transparent w/ ATA (easy transition)
- Low pin count for both host and devices (2 pairs)
- Favorable (low) voltages
- Supports lower cost device architectures
- Higher performance than equivalent ATA (data rate, queuing, overlap) w/ scalability to higher
- Much better cabling/connectors (thin, flexible)
- Includes efficient power delivery
- No software dependency. Relatively easy transition (price, IHV NRE and capital inventory risk, wide variety of devices at intro, etc.)
- Power management and power consumption suitable for mobile use
- Allows roadmap spanning ~10 years
- Cable length comparable to ATA (<1 m)
- Transfer rate exceeding best ATA (~150 MB/s) with scalability to higher rates
- Light protocol allowing overhead latencies to be minimized
- Asynchronous only (no isochronous requirements)
- No Peer-peer transfer support (to/from host only)
- Provides support for 1st party DMA access to host
- Cost competitive with equivalent parallel ATA solution at introduction (host + device + cable)
- Storage device centric (no cameras/scanners/printers)
- Easy installation/configuration (plug/play, no jumpers, no external terminators)
- Single host (no multi-initiators or host/host networking)

## 2.2 Migration Considerations

The following table illustrates the migration for various aspects of the Serial ATA signaling growth path:

| | Generation 1 | Generation 2 | Generation 3 |
|---|---|---|---|
| Approximate speed ( 8b side ) | 1.2 Gbits/s ( 150 Mbytes/sec ) | 2.4 Gbits/s | 4.8[1] Gbits/s |
| Approximate speed ( 10b side ) | 1.5 Gbits/sec | 3.0 Gbits/sec | 6.0[1] Gbits/sec |
| Estimated introduction date | mid 2001 | mid 2004 | mid 2007 |
| Connector | | Same as Gen 1 | May be upgraded |
| Cable | | Same as Gen 1 | May be upgraded |
| Signaling compatibility | | Compatible with Generation 1 | Compatible with Generation 2 - may be compatible with Generation 1 |

NOTE –

1. These speed specifications and schedules are subject to change

# 3 Normative references

The following standards contain provisions that, through reference in the text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Copies of the following documents can be obtained from ANSI: Approved ANSI standards, approved and draft international and regional standards (ISO, IEC, CEN/CENELEC, ITUT), and approved and draft foreign standards (including BSI, JIS, and DIN). For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org.

Additional availability contact information is provided below as needed.

## 3.1 Approved references

The following approved ANSI standards, approved international and regional standards (ISO, IEC, CEN/CENELEC, ITUT), may be obtained from the international and regional organizations who control them.

AT Attachment with Packet Interface – 5 (ATA/ATAPI-5) [NCITS 340:2000]

To obtain copies of these documents, contact Global Engineering or NCITS.

## 3.2 Other references

The 8b/10b code used in Serial ATA is based on the following published references

[1] A.X. Widmer and P.A. Franaszek, “A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code.” *IBM Journal of Research and Development*, 27, no. 5: 440-451 (September, 1983)

[2] U.S. Patent 4,486,739. Peter A. Franaszek and Albert X. Widmer. *Byte Oriented DC Balanced (0,4) 8B/10B Partitioned Block Transmission Code*. (Dec. 4, 1984)

Additional documentation relating to the AT Attachment (ATA) storage interface can be retrieved via the World Wide Web from the Technical Committee T13 at http://www.t13.org.

# 4 Definitions, abbreviations, and conventions

## 4.1 Definitions and abbreviations

### 4.1.1 ATA (AT Attachment)

ATA defines the physical, electrical, transport, and command protocols for the internal attachment of storage devices.

### 4.1.2 ATAPI (AT Attachment Packet Interface) device

A device implementing the Packet Command feature set.

### 4.1.3 backchannel

A term used to describe or refer to the transmit same-side SATA interface, when the scope of the paragraph is addressing the receive interface. For example, when discussing the receive serial ATA interface on the device side, the term “backchannel” would be used to describe the transmit interface on the device side.

### 4.1.4 bit synchronization

The state in which a receiver has synchronized the internal receiver clock to the external transmitter and is delivering retimed serial data.

### 4.1.5 byte

A byte is 8 bits of data. The least significant bit is bit 0 and the most significant bit is bit 7. The most significant bit is shown on the left. In the encoding process the bits in a byte are referred to as HGFEDCBA, or “A,B,C,D,E,F,G,H” where A corresponds to bit 0 and H corresponds to bit 7. Case is significant.

### 4.1.6 character

A character is a representation of a data byte or control code. There are two types of characters: data characters and control characters.

### 4.1.7 character alignment

Character alignment is a receiver action that resets the character boundary to that of the comma sequence found in the K28.5 control character of the ALIGN primitive, and establishes Dword synchronization of the incoming serial data stream.

### 4.1.8 character slipping

Character slipping is the receiver action that realigns the receiver’s clock to the received bit stream by adding or removing bit times within the characters of the ALIGN primitive.

### 4.1.9 code violation

A code violation is an error that occurs in the reception process as a result of (1) a running disparity violation or (2) an encoded character that does not translate to a valid data or control character or (3) an encoded character that translates to a control character other than K28.5 or K28.3 in byte 0 of a Dword or (4) an encoded character that translates to any control character (valid or invalid) in bytes 1-3 of a Dword.

### 4.1.10 comma character

A comma character is a control character, that when encoded, contains the comma sequence. In Serial ATA the only comma character used is K28.5, and only the ALIGN primitive contains the comma character. The comma sequence is the first seven bits of the encoded character.

### 4.1.11 comma sequence

The comma sequence is a seven-bit sequence of 0011111 or 1100000 in an encoded stream. The comma sequence is unique in that it appears only in a single encoded character, and furthermore, cannot appear in any subset of bits in adjacent encoded characters. This unique property allows the comma sequence to be used for determining alignment of the received data stream.

### 4.1.12 command aborted

Command aborted is command completion with ABRT set to one in the Error register, and ERR set to one in the Status register.

### 4.1.13 command completion

Command completion describes the completion of an action requested by command, applicable to the device. Command completion also applies to the case where the command has terminated with an error, and the following actions occurred:

a) the appropriate bits of the Status Register have been updated.
b) BSY & DRQ have been cleared to zero
c) assertion of INTRQ (if nIEN is active-low), assuming that the command protocol specifies INTRQ to be asserted.

In Serial ATA, the register contents are transferred to the host using a Register - Device-to-Host FIS.

### 4.1.14 command packet

A command packet is a data structure transmitted to the device by a PACKET command that includes the command and command parameters.

### 4.1.15 Control Block registers

Control Block registers are interface registers used for device control and to post alternate status.

### 4.1.16 control character

A control character is a combination of a byte value with the control variable equal to K.

### 4.1.17 control variable

The control variable, Z, is a flag that determines the code set to be used to interpret a data byte. The control variable has the value D (for data characters) or K (for control characters).

### 4.1.18 CRC

In Serial ATA a 32-bit CRC is calculated over the contents of a Frame Information Structure. The Serial ATA CRC is the Dword in a frame that immediately precedes the EOF primitive.

### 4.1.19 data character

A data character is a combination of a byte value with the control variable equal to D.

### 4.1.20 device

Device is a storage peripheral. Traditionally, a device on the interface has been a hard disk drive, but any form of storage device may be placed on the interface provided the device adheres to this standard.

### 4.1.21 DMA (direct memory access)

Direct memory access is a means of data transfer between device and host memory without host processor intervention.

### 4.1.22 Dword

A Dword is thirty-two (32) bits of data. A Dword may be represented as 32 bits, as two adjacent words, or as four adjacent bytes. When shown as bits the least significant bit is bit 0 and most significant bit is bit 31. The most significant bit is shown on the left. When shown as words the least significant word (lower) is word 0 and the most significant (upper) word is word 1. When shown as bytes the least significant byte is byte 0 and the most significant byte is byte 3. See Figure 1 for a description of the relationship between bytes, words, and Dwords. Dwords are aligned on four byte boundaries to a zero reference defined by a comma character.

### 4.1.23 Dword synchronization

The state in which a receiver has recognized the comma sequence and is producing an aligned data stream of Dwords (four contiguous bytes) from the zero-reference of the comma character.

### 4.1.24 encoded character

An encoded character is the output of the 8b/10b encoder – the result of encoding a character. An encoded character consists of 10 bits, where bit 0 is the most significant bit and bit 9 is the least significant. The bits in an encoded character are symbolically referred to as “abcdeifghj” where “a” corresponds to bit 0 and “j” corresponds to bit 9. Case is significant. Note the out-of-order representation. See Figure 1 for a description of the relationship between bytes, characters and encoded characters.

### 4.1.25 elasticity buffer

The elasticity buffer is a portion of the receiver where character slipping and/or character alignment is performed.

### 4.1.26 First-party DMA access

First-party DMA access is a method by which a device accesses host memory.

### 4.1.27 First-party DMA mode

A device which is operating in First-party DMA mode uses First-party DMA as a primary communications method between the host and the device. A software driver uses legacy mode commands to place the device into First-party DMA mode of operation. The legacy-mode command to place the device into the First-party DMA mode of operation and the command protocol used between a device and host when in First-party DMA mode are not specified by this specification.

### 4.1.28 FIS

See Frame Information Structure.

### 4.1.29 Frame Information Structure

The user payload of a frame, does not include the SOF, CRC, and EOF delimiters.

### 4.1.30 frame

A frame is an indivisible unit of information exchanged between a host and device. A frame consists of a SOF primitive, a Frame Information Structure, a CRC calculated over the contents of the FIS, and an EOF primitive.

### 4.1.31 interrupt pending

Interrupt pending is an internal state of the device that exists when the device protocol requires the device to notify the host of an event by asserting INTRQ, given the condition where nIEN is asserted active-low to zero.

### 4.1.32 legacy device

A legacy device is a device which operates in legacy mode.

### 4.1.33 legacy mode

Legacy mode is the mode of operation which provides software-transparent communication of commands and status between a host and device using the ATA Command Block and Control Block registers.

### 4.1.34 legal character

A legal character is one for which there exists a valid decoding, either into the data character or control character fields. Due to running disparity constraints not all 10-bit combinations result in a legal character. Additional usage restrictions in Serial ATA result in a further reduction in the SATA defined control character space.

### 4.1.35 LFSR

See Section 7.4.5

### 4.1.36 PIO (programmed input/output)

PIO is a means of accessing device registers. PIO is also used to describe one form of data transfers. PIO data transfers are performed by the host processor utilizing PIO register accesses to the Data register.

### 4.1.37 primitive

A primitive is a single Dword of information that consists of a control character in byte 0 followed by three additional data characters in bytes 1-3.

### 4.1.38 sector

A sector is a uniquely addressable set of predetermined size, usually 256 words (512 bytes).

### 4.1.39 Shadow Register Block registers

Shadow Register Block registers are interface registers used for delivering commands to the device or posting status from the device.

### 4.1.40 unrecoverable error

An unrecoverable error is defined as having occurred at any point when the device sets either the ERR bit or the DF bit to one in the Status register at command completion.

### 4.1.41 word

A word is sixteen (16) bits of data. A word may be represented as 16 bits or as two adjacent bytes. When shown as bits the least significant bit is bit 0 and most significant bit is bit 15. The most significant bit is shown on the left. When shown as bytes the least significant byte (lower) byte is byte 0 and the most significant byte (upper) byte is byte 1. See Figure 1 for a description of the

relationship between bytes, words and Dwords. The definition of a word in Serial ATA is the same as the definition of a word in ATA.

## 4.2 Conventions

Lowercase is used for words having the normal English meaning. Certain words and terms used in this document have a specific meaning beyond the normal English meaning. These words and terms are defined either in clause 4.1 or in the text where they first appear.

The names of abbreviations, commands, fields, and acronyms used as signal names are in all uppercase (e.g., IDENTIFY DEVICE). Fields containing only one bit are usually referred to as the "name" bit instead of the "name" field.

Names of device registers begin with a capital letter (e.g., Cylinder Low register).

### 4.2.1 Precedence

If there is a conflict between text, figures, and tables, the precedence shall be tables, figures, and then text.

### 4.2.2 Keywords

Several keywords are used to differentiate between different levels of requirements and optionality.

#### 4.2.2.1 expected

A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

#### 4.2.2.2 mandatory

A keyword indicating items to be implemented as defined by this standard.

#### 4.2.2.3 may

A keyword that indicates flexibility of choice with no implied preference.

#### 4.2.2.4 obsolete:

A keyword used to describe bits, bytes, fields, and code values that no longer have consistent meaning or functionality from one implementation to another. However, some degree of functionality may be required for items designated as “obsolete” to provide for backward compatibility. An obsolete bit, byte, field, or command shall never be reclaimed for any other use in any future standard. Bits, bytes, fields, and code values that had been designated as “obsolete” in previous standards may have been reclassified as “retired” in this standard based on the definitions herein for “obsolete” and “retired”.

#### 4.2.2.5 optional

A keyword that describes features that is not required by this standard. However, if any optional feature defined by the standard is implemented, the feature shall be implemented in the way defined by the standard.

#### 4.2.2.6 retired

A keyword indicating that the designated bits, bytes, fields, and code values that had been defined in previous standards are not defined in this standard and may be reclaimed for other uses in future standards. If retired bits, bytes, fields, or code values are utilized before they are reclaimed, they shall have the meaning or functionality as described in previous standards.

#### 4.2.2.7 reserved

A keyword indicating reserved bits, bytes, words, fields, and code values that are set-aside for future standardization. Their use and interpretation may be specified by future extensions to this or other standards. A reserved bit, byte, word, or field shall be set to zero, or in accordance with a future extension to this standard. The recipient shall not check reserved bits, bytes, words, or fields. Receipt of reserved code values in defined fields shall be treated as a command parameter error and reported by returning command aborted.

#### 4.2.2.8 shall

A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other standard conformant products.

#### 4.2.2.9 should

A keyword indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “it is recommended”.

### 4.2.3 Numbering

Numbers that are not immediately followed by a lowercase "b" or "h" are decimal values. Numbers that are immediately followed by a lowercase "b" (e.g., 01b) are binary values. Numbers that are immediately followed by a lowercase "h" (e.g., 3Ah) are hexadecimal values.

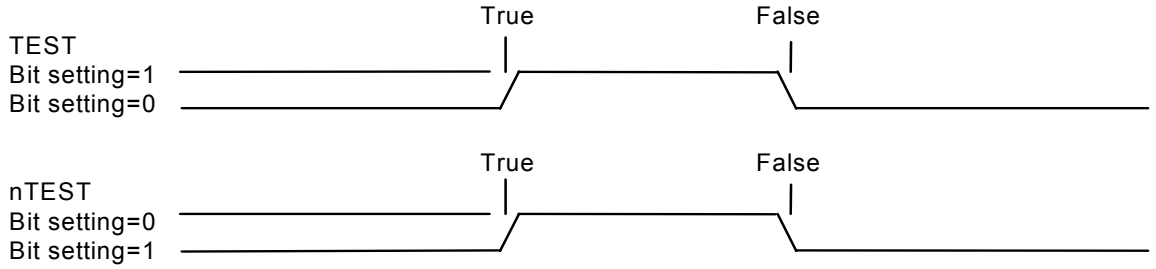
### 4.2.4 Signal conventions

Signal names are shown in all uppercase letters.

All signals are either high active or low active signals. A Number Sign character (#) at the end of a signal name indicates the signal is a low active signal.

### 4.2.5 Bit conventions

Bit names are shown in all uppercase letters except where a lowercase n precedes a bit name. If there is no preceding n, then when BIT is set to one the meaning of the bit is true, and when BIT is cleared to zero the meaning of the bit is false. If there is a preceding n, then when nBIT is cleared to zero the meaning of the bit is true and when nBIT is set to one the meaning of the bit is false.



### 4.2.6 State diagram conventions

For each function to be completed a state machine approach is used to describe the sequence requirements. Each function is composed of several states to accomplish a set goal. Each state of the set is described by an individual state table. Table 1 below shows the general layout for each of the state tables that comprise the set of states for the function.

**Table 1 – State Table Cell Description**

| **State name or identifier** | Action list[P \| W] | | |
|---|---|---|---|
| | Branch condition 0 | → | **Next state 0** |
| | Branch condition 1 | → | **Next state 1** |

Each state is identified by a state designator and a state name. The state designator is unique among all states in all state diagrams in this document. The state designator consists of a set of letters that are capitalized in the title of the figure containing the state diagram followed by a unique number. The state name is a brief description of the primary action taken during the state, and the same state name may appear in other state diagrams. If the same primary function occurs in other states in the same state diagram, they are designated with a unique letter at the end of the name. Additional actions may be taken while in a state and these actions are described in the state description text.

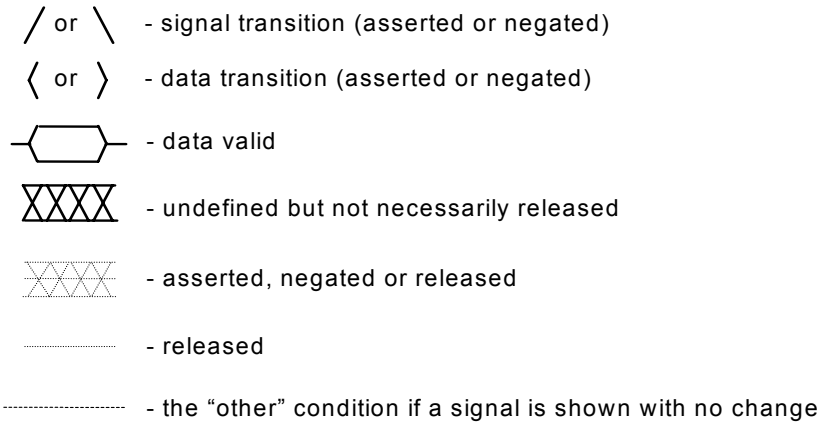
Each transition is identified by a transition label and a transition condition. The transition label consists of the state designator of the state from which the transition is being made followed by the state designator of the state to which the transition is being made. In some cases, the transition to enter or exit a state diagram may come from or go to a number of state diagrams, depending on the command being executed. In this case, the state designator is labeled xx. The transition condition is a brief description of the event or condition that causes the transition to occur and may include a transition action that is taken when the transition occurs. This action is described fully in the transition description text.

Upon entry to a state, all actions to be executed in that state are executed. If a state is re-entered from itself, all actions to be executed in the state are executed again.

It is assumed that all actions are executed within a state and that transitions from state to state are instantaneous.

### 4.2.7 Timing conventions

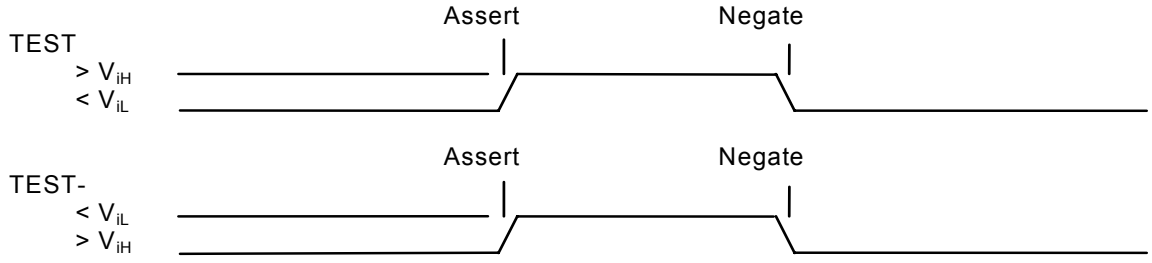
Certain symbols are used in the timing diagrams. These symbols and their respective definitions are listed below.




All signals are shown with the asserted condition facing to the top of the page. The negated condition is shown towards the bottom of the page relative to the asserted condition.

The interface uses a mixture of negative and positive signals for control and data. The terms asserted and negated are used for consistency and are independent of electrical characteristics.

In all timing diagrams, the lower line indicates negated, and the upper line indicates asserted. The following illustrates the representation of a signal named TEST going from negated to asserted and back to negated, based on the polarity of the signal.




In the case of the Serial ATA interface itself, all signals are differential. Each signal is actually transmitted or received on two lines and the state of the signal is determined by the relationship of these lines to each other rather than to a reference ground. These two line are named the same except the names end in either “+” or “-“. The above symbols are used when representing these signals however, assertion of a differential signal is assumed to mean that the “+” signal line is positive in relation to the “-“ line. In other words, if the signal is “TX”, assertion means that the voltage on TX+ minus the voltage on TX- results in a positive number. Conversely, negation means that the result is a negative number.

### 4.2.8 Byte, word and Dword Relationships

Figure 1 – Byte, word and Dword relationships illustrates the relationship between bytes, words and Dwords.

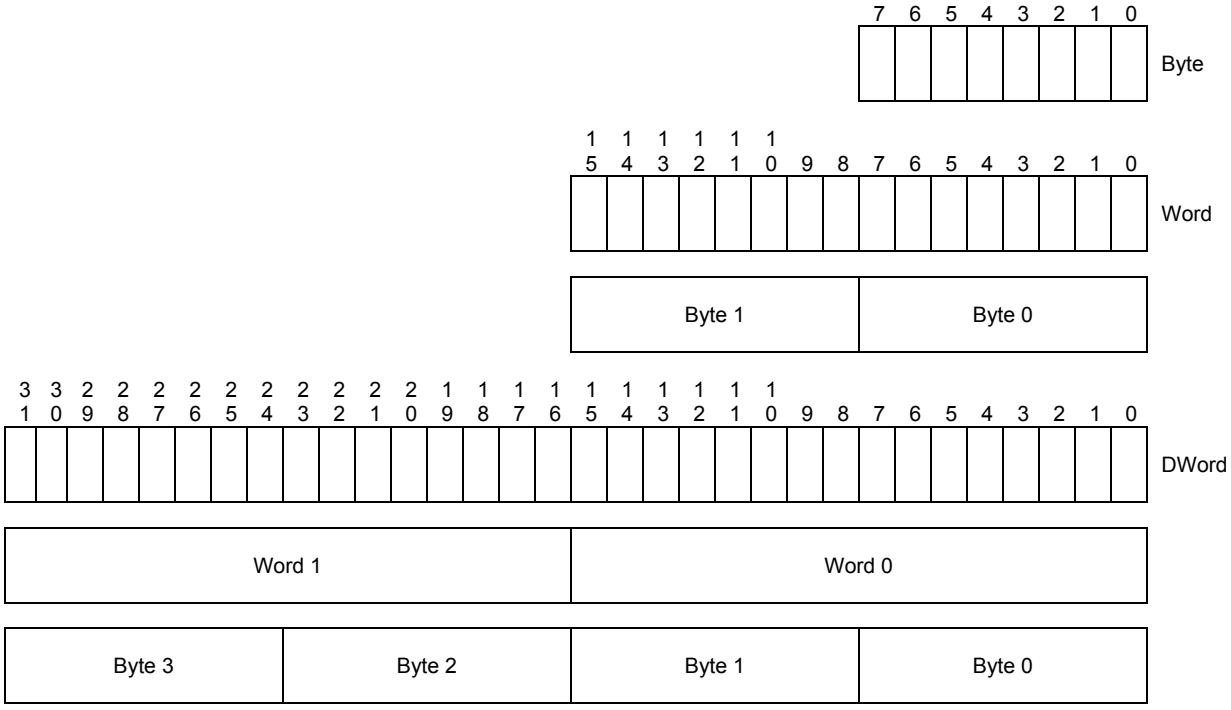



**Figure 1 – Byte, word and Dword relationships**

# 5 General overview

Serial ATA is a high-speed serial link replacement for the parallel ATA attachment of mass storage devices. The serial link employed is a high-speed differential layer that utilizes Gigabit technology and 8b/10b encoding.

Figure 2 – Standard ATA device connectivity illustrates how two devices are connected to a standard ATA host adapter. This method allows up to two devices to be connected to a single port using a Master/Slave communication technique. Each device is connected via a ribbon cable that “daisy chains” the devices.

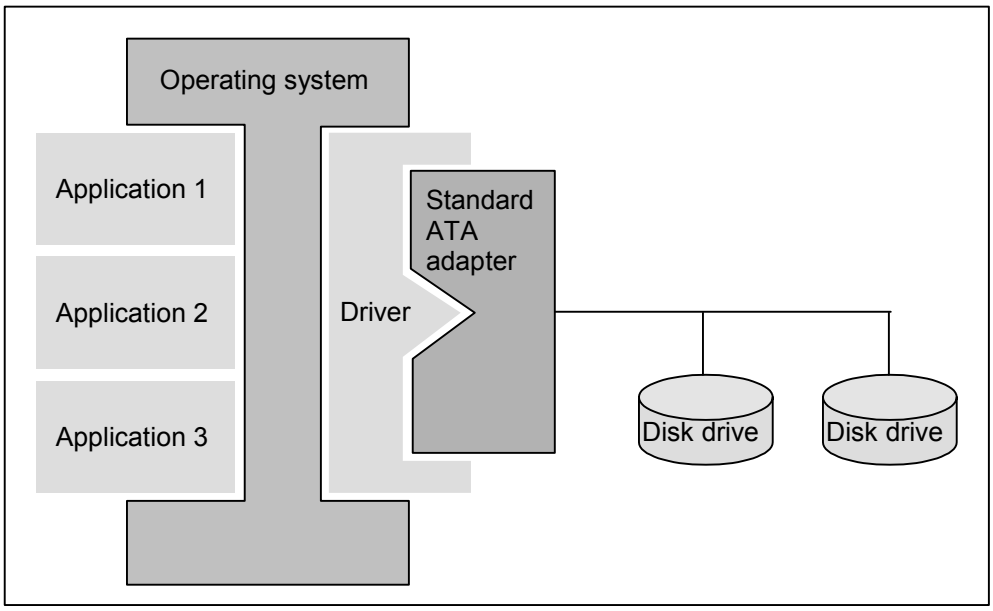



**Figure 2 – Standard ATA device connectivity**

Figure 3 – Serial ATA connectivity illustrates how the same two devices are connected using a Serial ATA host adapter. In this diagram the dark grey portion is functionally identical to the dark grey portion of the previous diagram. Legacy host software that is only ATA aware accesses the Serial ATA subsystem in the exact same manner and will function correctly. In this case, however, the software views the two devices as if they were “masters” on two separate ports. The right hand portion of the host adapter is of a new design that converts the normal operations of the software into a serial data/control stream. The Serial ATA structure connects each of the two drives with their own respective cables in a point-to-point fashion.

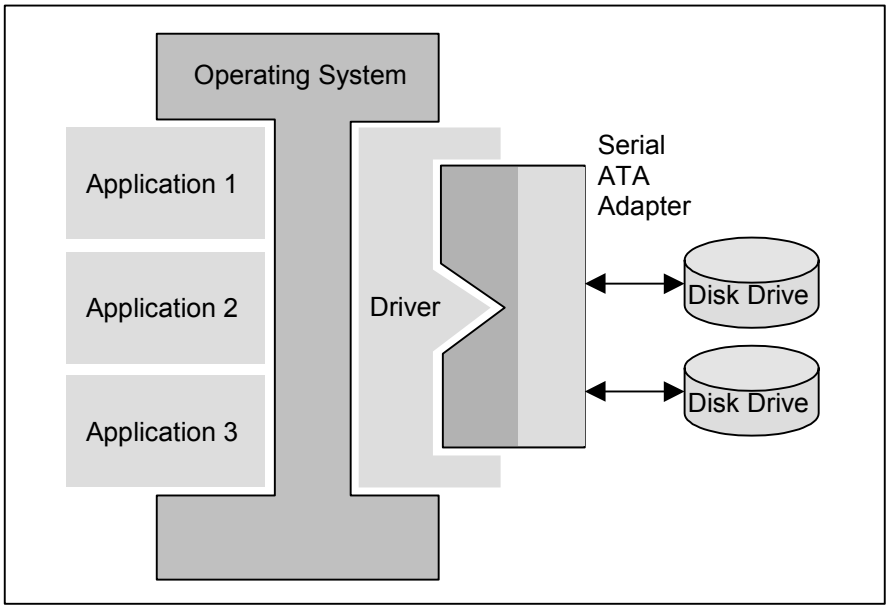



**Figure 3 – Serial ATA connectivity**

## 5.1 Sub-module operation

The Transport control state machine and the Link state machine are the two core sub-modules that control overall operation. The Link state machine controls the operation(s) related to the serial line and the Transport control state machine controls the operation(s) relating to the host platform. The two state machines coordinate their actions and utilize resources to transfer data between a host computer and attached mass storage device. The host Link state machine communicates via the serial line to a corresponding Link state machine located in the device. The host Transport machine also likewise communicates with a corresponding device Transport state machine. The two Link state machines ensure that control sequences between the two Transport control state machines are properly exchanged. Figure 4 shows how the machines communicate their various needed parameters in the traditional layered model. Each layer communicates with its counterpart directly or indirectly

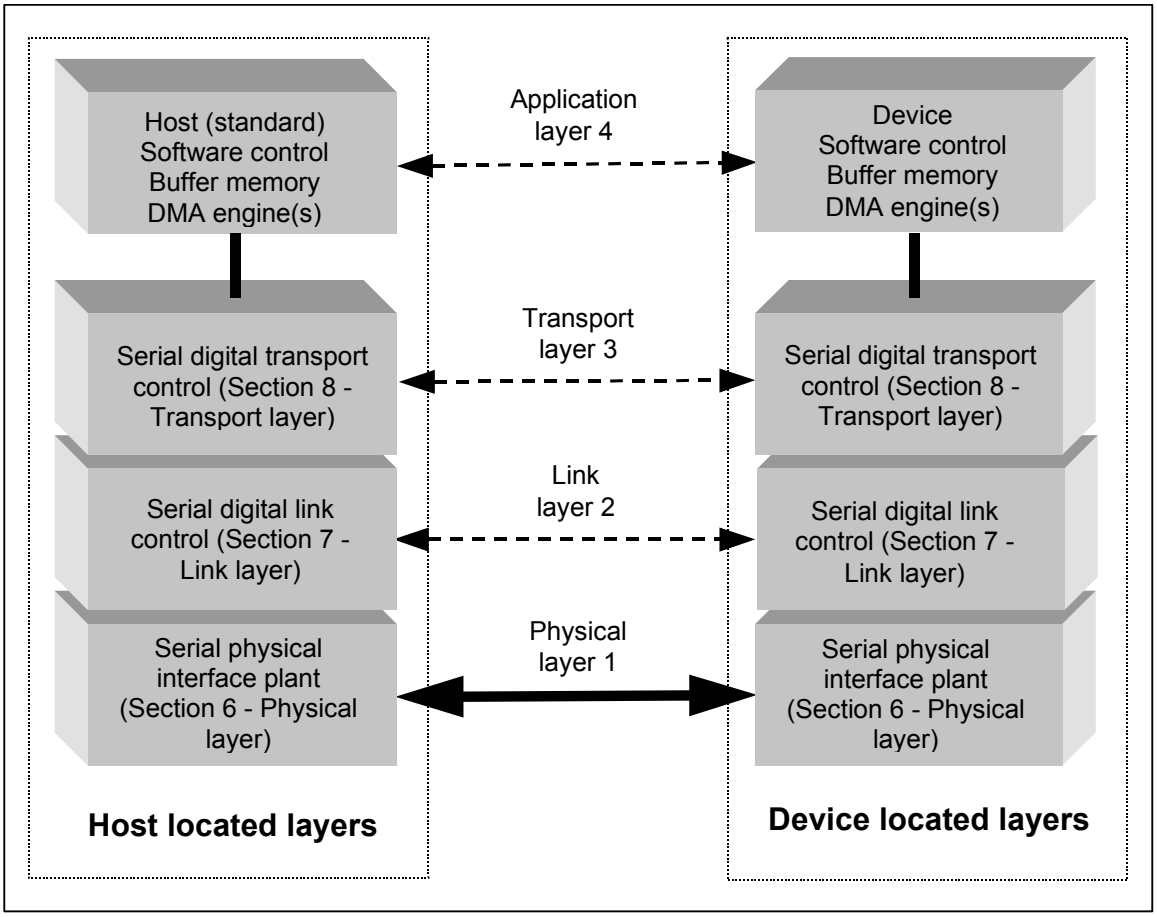



**Figure 4 – Communication layers**

The host interacts with the Transport control state machine through a register interface that is equivalent to that presented by a traditional parallel ATA host adapter. The host software follows existing standards and conventions when accessing the register interface and follows standard command protocol conventions. The Transport control state machine breaks down these operations into a sequence of actions that are exchanged with the Link state machine. The Transport control state machine uses resources located in the ATA interface sub-module to perform its operations.

## 5.2 Standard ATA Emulation

Emulation of parallel ATA device behavior as perceived by the host BIOS or software driver, is a cooperative effort between the device and the Serial ATA host adapter hardware. The behavior of Command and Control Block registers, PIO and DMA data transfers, resets, and interrupts are all emulated.

The host adapter contains a set of registers that shadow the contents of the traditional device registers, referred to as the Shadow Register Block. All Serial ATA devices behave like Device 0 devices. Devices shall ignore the DEV bit in the Device/Head field of received Register FIS's, and it is the responsibility of the host adapter to gate transmission of Register FIS's to devices, as appropriate, based on the value of the DEV bit.

After a reset or power-on, the host bus adapter emulates the behavior of a traditional ATA system during device discovery. Immediately after reset, the host adapter shall place the value 0x7Fh in its Status shadow register and shall place the value 0xFFh in all the other taskfile registers (0xFFFFh in the Data register). In this state the host bus adapter shall not accept writes to the taskfile shadow registers. When the Phy detects presence of an attached device, the host bus adapter shall now set bit 7 in the Status shadow register yielding the value 0xFFh, and the host bus adapter shall now allow writes to the shadow taskfile registers. If a device is present, the Phy shall take no longer than 10 ms to indicate that it has detected the presence of a device, and has set bit 7 in the Status shadow register. When the attached device establishes communication with the host bus adapter, it shall send a register FIS to the host, resulting in the shadow registers being updated with values appropriate for the attached device.

The host adapter may present a Master-only emulation to host software, that is, each device is a Device 0, and each Device 0 is accessed at a different set of host bus addresses. The host adapter may optionally present a Master/Slave emulation to host software, that is, two devices on two separate Serial ATA ports are represented to host software as a Device 0 and a Device 1 accessed at the same set of host bus addresses.

### 5.2.1 Software Reset

According to the parallel ATA specification, issuing a software reset is done by toggling the SRST bit in the Control register. The toggle period is stipulated as being no shorter than 5us. As a result of the SRST bit changing in the Control register, Serial ATA host adapters shall issue at least two Register FISes to the device (one with the SRST bit asserted and a subsequent one with the SRST bit cleared). See section 8.5.2 for a detailed definition of Register FIS. Although host software is required to toggle the SRST bit no faster than 5us, devices may not rely on the inter-arrival time of received Register FISes also meeting this timing. Because of flow control, frame handshaking, and other protocol interlocks, devices may effectively receive the resulting Register FISes back-to-back.

Due to flow control, protocol interlocks, power management state, or other transmission latencies, the subsequent Register frame transmission clearing the SRST bit during a software reset may be triggered prior to the previous Register FIS transmission having been completed. Host adapters are required to allow host software to toggle the SRST with the minimum 5us timing specified in the parallel ATA specification even if frame transmission latencies result in the first Register FIS transmission taking longer than 5us. Host adapters are required to ensure transmission of the two resulting Register FISes to the device regardless of the transmission latency of each individual FIS.

### 5.2.2 Master-only emulation

A native Serial ATA host adapter behaves the same as if a legacy master only device were attached with no slave present. It is the responsibility of the host adapter to properly interact with host software and present the correct behavior for this type of configuration. All Serial ATA devices, therefore, need not be aware of master / slave issues and ignore legacy task file information that deal with a

secondary device. When the DEV bit in the Device/Head register is set to one, selecting the non-existent Device 1, the host adapter shall respond to register reads and writes as specified for a Device 0 with no Device 1 present, as defined in the ATA standard. This includes not setting the BSY bit in the shadow Status register when Device 1 is selected, as described in the ATA standard. When Device 0 is selected, the host adapter shall execute the Serial ATA protocols for managing the shadow register contents as defined in later sections.

When Device 0 is selected and the Command register is written in the Shadow Register Block, the host adapter sets the BSY bit in its shadow Status register. The host adapter then transmits a frame to the device containing the new register contents. When the Device Control register is written in the Shadow Register Block with a change of state of the SRST bit, the host adapter sets the BSY bit in its shadow Status register and transmit a frame to the device containing the new register contents. Transmission of register contents when the Device Control register is written with any value that is not a change of state of the SRST bit shall not set the BSY bit in the shadow Status register, and transmission of a frame to the device containing new register contents is optional. Similarly, the host adapter sets the BSY bit in its shadow Status register to one when a COMRESET is requested or the SRST bit is set to one in the Device Control register. The expected timing for BSY bit assertion is thereby preserved.

The device updates the contents of the host adapter Shadow Register Block by transmitting a register frame to the host. This allows the device to set the proper ending status in the host adapter Shadow Register Block at the completion of a command or control request. Specific support is added to ensure proper timing of the DRQ and BSY bits in the Status register for PIO transfers.

Finally the host adapter cooperates with the device by providing an interrupt pending flag in the host adapter. This flag is set by the host adapter when the device sends a serial bus frame including the request to set the interrupt pending flag. The host adapter asserts the interrupt to the host processor any time the interrupt pending flag is set, the DEV bit is cleared to zero in the shadow Device/Head register, and the nIEN bit in the shadow Device Control register is cleared to zero. The host adapter clears the interrupt pending flag any time a COMRESET is requested, the SRST bit in the shadow Device Control register is set to one, the shadow Command register is written with DEV cleared to zero, or the shadow Status register is read with DEV cleared to zero. This allows the emulation of the host interrupt and its proper timing.

### 5.2.3 Master/Slave emulation (optional)

All devices behave as if they are Device 0 devices. However, the host adapter may optionally implement emulation of the behavior of a Master/Slave configuration by pairing two Serial ATA ports, and managing their associated shadow registers accordingly, as though they were a Device 0 and Device 1 at the same set of host bus addresses.

A host adapter that emulates Master/Slave behavior shall manage the two sets of shadow registers (one set for each of the two devices) based on the value of the DEV bit in the shadow Device/Head register. Based on the value of the DEV bit, the host adapter shall direct accesses to the Shadow Register Block Registers, and accesses to the Control Block Registers, to the appropriate set of shadow registers in the correct device. It is the responsibility of the host adapter to ensure that communication with one or both of the attached devices is handled properly, and that information gets routed to the devices correctly. Each device shall process any communication with the host adapter as if it is targeted for the device regardless of the value of the DEV bit.

If a host adapter is emulating Master/Slave behavior, and there is no device attached to the cable designated as the Device 1 cable, the host adapter shall emulate Device 0 behavior with no Device 1 present as described in the ATA standard.

#### 5.2.3.1 Software reset

Host adapters that emulate Master/Slave behavior shall emulate proper behavior for software reset. Based on the Phy initialization status, the host adapter knows whether a device is attached to each of the two ports used in a Master/Slave emulation configuration. Device Control Register writes, that have the SRST bit set to one, shall result in the associated shadow register being written for each port to which a device is attached. The frame transmission protocol for each associated port executed, results in a Register FIS being transmitted to each attached device. Similarly, the subsequent write to the Device Control register that clears the SRST bit shall result in a Register FIS being sent to each attached device. The host adapter shall then await a response from each attached device (or timeout), and shall merge the contents of the Error and Status registers for the attached devices, in accordance with the ATA standard, to produce the Error and Status register values visible to host software.

#### 5.2.3.2 EXECUTE DEVICE DIAGNOSTICS

Host adapters that emulate Master/Slave behavior shall emulate proper behavior for EXECUTE DEVICE DIAGNOSTICS. The host adapter shall detect the EXECUTE DEVICE DIAGNOSTIC command being written to the Command register. Detecting the EXECUTE DEVICE DIAGNOSTICS command shall result in the associated shadow register being written for each port to which a device is attached. The frame transmission protocol for each associated port executed, results in a Register FIS being transmitted to each attached device. The host adapter shall then await response from each attached device (or timeout), and shall merge the contents of the Error and Status registers for the attached devices, in accordance with the ATA standard to produce the Error and Status register values visible to host software.

#### 5.2.3.3 Restrictions and limitations

Superset capabilities that are unique to Serial ATA and not supported by legacy parallel ATA are not required to be supported in Master/Slave emulation mode. Such capabilities include support for first-party DMA, dynamic plug/unplug, fine-grained interface power management, and superset status and control information. Master/Slave emulation is recommended only in configurations where legacy software drivers are used and the number of attached devices exceeds the number of taskfile interfaces the legacy software supports.

### 5.2.4 Standard ATA interoperability state diagrams

This state diagram defines the protocol of the host adapter to emulate Master only legacy ATA devices as seen from the host BIOS or software driver. The interrupt pending flag (IPF) is an internal state bit in the host adapter that reflects whether or not the device has an interrupt pending to the host.

| HA0: HA_SEL/NOINTRQ | The interrupt signal is not asserted to the host. | | |
|---|---|---|---|
| 1. | COMRESET requested by the host. | → | HA_SEL/NOINTRQ |
| 2. | Device Control register written by the host with SRST=0 and (nIEN=1 or IPF=0). | → | HA_SEL/NOINTRQ |
| 3. | Device Control register written by the host with SRST=0 and (nIEN=0 and IPF=1). | → | HA_SEL/INTRQ |
| 4. | Device Control register written by the host with SRST=1. | → | HA_SEL/NOINTRQ |
| 5. | Command register written by the host. | → | HA_SEL/NOINTRQ |
| 6. | Device/Head register written by the host with DEV = 1. | → | HA_NOTSEL |
| 7. | Any other register read or write by the host | → | HA_SEL/NOINTRQ |
| 8. | Transport layer indicates new register content from the device with no set IPF flag. | → | HA_SEL/NOINTRQ |
| 9. | Transport layer indicates new register content from the device with set IPF flag and nIEN=1. | → | HA_SEL/NOINTRQ |
| 10. | Transport layer indicates new register content from the device with set IPF flag and nIEN=0. | → | HA_SEL/INTRQ |

| HA1: HA_SEL/INTRQ | The interrupt signal is asserted to the host. | | |
|---|---|---|---|
| 1. | COMRESET requested by the host. | → | HA_SEL/NOINTRQ |
| 2. | Device Control register written by the host with SRST=0 and (nIEN=1 or IPF=0). | → | HA_SEL/NOINTRQ |
| 3. | Device Control register written by the host with SRST=0 and (nIEN=0 and IPF=1). | → | HA_SEL/INTRQ |
| 4. | Device Control register written by the host with SRST=1. | → | HA_SEL/NOINTRQ |
| 5. | Command register written by the host. | → | HA_SEL/NOINTRQ |
| 6. | Device/Head register written by the host with DEV = 1. | → | HA_NOTSEL |
| 7. | Status register read by the host | → | HA_SEL/NOINTRQ |
| 8. | Any other register write by the host | → | HA_SEL/INTRQ |
| 9. | Any other register read by the host | → | HA_SEL/INTRQ |
| 10. | Transport layer indicates new register content from the device. | → | HA_SEL/INTRQ |

| HA2: HA_NOTSEL | The interrupt signal is not asserted to the host. | |
|---|---|---|
| 1. COMRESET requested by the host. | → | HA_SEL/NOINTRQ |
| 2. Device Control register written by the host with SRST = 0. | → | HA_NOTSEL |
| 3. Device Control register written by the host with SRST = 1. | → | HA_SEL/NOINTRQ |
| 4. Command register written by the host with command other than EXECUTE DEVICE DIAGNOSTIC. | → | HA_NOTSEL |
| 5. Command register written by the host with command EXECUTE DEVICE DIAGNOSTIC. | → | HA_SEL/NOINTRQ |
| 6. Device/Head register written by the host with DEV = 0 and (nIEN=1 or IPF=0). | → | HA_SEL/NOINTRQ |
| 7. Device/Head register written by the host with DEV = 0 and (nIEN=0 and IPF=1). | → | HA_SEL/INTRQ |
| 8. Any other register write by the host | → | HA_NOTSEL |
| 9. Read of Status or Alternate Status register by the host. | → | HA_NOTSEL |
| 10. Read of any other register by the host. | → | HA_NOTSEL |
| 11. Transport layer indicates new register content from the device. | → | HA_NOTSEL |

**HA0: HA_SEL/NOINTRQ state:** This state is entered when Device 0 is selected and either IPF is cleared to zero or nIEN is set to one.

When in this state, the interrupt signal to the host shall not be asserted.

**Transition HA0:1:** When a COMRESET is requested by the host, the host adapter shall set BSY to one in the shadow Status register, clear IPF to zero, notify the Link layer to have a bus COMRESET asserted, and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA0:2:** When the Device Control register is written by the host with SRST cleared to zero and either nIEN set to one or IPF is cleared to zero, the host adapter shall notify the Transport layer to send a ContrlFIS with the current content of the shadow registers, and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA0:3:** When the Device Control register is written by the host with SRST cleared to zero, nIEN cleared to zero, and IPF is set to one, the host adapter shall notify the Transport layer to send a ContrlFIS with the current content of the shadow registers, and make a transition to the HA1: HA_SEL/INTRQ state.

**Transition HA0:4:** When the Device Control register is written by the host with SRST set to one, the host adapter shall set BSY to one in the shadow Status register, clear IPF to zero, notify the Transport layer to send a ContrlFIS with the current content of the shadow registers, and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA0:5:** When the Command register is written by the host, the host adapter shall set BSY to one in the shadow Status register, clear IPF to zero, notify the Transport layer to send a CmdFIS with the current content of the shadow registers, and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA0:6:** When the Device/Head register is written by the host with DEV set to one, the host adapter shall make a transition to the HA2: HA_NOTSEL state.

**Transition HA0:7:** When any register is written by the host other than those described above, the host adapter shall make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA0:8:** When the Transport layer indicates new register content from the device with no set IPF flag, the host adapter shall place the new register content into the shadow registers and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA0:9:** When the Transport layer indicates new register content from the device with set IPF flag and nIEN is set to one, the host adapter shall set IPF to one, place the new register content into the shadow registers, and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA0:10:** When the Transport layer indicates new register content from the device with set IPF flag and nIEN is cleared to zero, the host adapter shall set IPF to one, place the new register content into the shadow registers, and make a transition to the HA1: HA_SEL/INTRQ state.

**HA1: HA_SEL/INTRQ state:** This state is when DEVICE 0 is selected, nIEN is cleared to zero, and IPF is set to one.

When in this state, the interrupt signal to the host shall be asserted.

**Transition HA1:1:** When a COMRESET is requested by the host, the host adapter shall set BSY to one in the shadow Status register, clear IPF to zero, notify the Link layer to have a bus COMRESET asserted, and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA1:2:** When the Device Control register is written by the host with SRST cleared to zero, with nIEN set to one or IPF cleared to zero, the host adapter shall notify the Transport layer to send a ContrlFIS with the current content of the shadow registers, and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA1:3:** When the Device Control register is written by the host with SRST cleared to zero with nIEN cleared to zero and IPF set to one, the host adapter shall notify the Transport layer to send a ContrlFIS with the current content of the shadow registers, and make a transition to the HA1: HA_SEL/INTRQ state.

**Transition HA1:4:** When the Device Control register is written by the host with SRST set to one, the host adapter shall set BSY to one in the shadow Status register, clear IPF to zero, notify the Transport layer to send a ContrlFIS with the current content of the shadow registers, and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA1:5:** When the Command register is written by the host, the host adapter shall set BSY to one in the shadow Status register, clear IPF to zero, notify the Transport layer to send a CmdFIS with the current content of the shadow registers, and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA1:6:** When the Device/Head register is written by the host with DEV set to one, the host adapter shall make a transition to the HA2: HA_NOTSEL state.

**Transition HA1:7:** When the Status register is read by the host, the host adapter shall clear IPF to zero and make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA1:8:** When any register is written by the host other than those described above, the host adapter shall make a transition to the HA1: HA_SEL/INTRQ state.

**Transition HA1:9:** When any register is read by the host other than that described above, the host adapter shall make a transition to the HA1: HA_SEL/INTRQ state.

**Transition HA1:10:** When the Transport layer indicates new register content from the device, the host adapter shall place the new register content into the shadow registers and make a transition to the HA1: HA_SEL/INTRQ state.

**HA2: HA_NOTSEL state:** This state is entered when DEVICE 1 is selected.

When in this state, the interrupt signal to the host shall not be asserted.

**Transition HA2:1:** When a COMRESET is requested by the host, the host adapter shall set BSY to one in the shadow Status register, clear DEV to zero in the shadow Device/Head register, clear IPF to zero, notify the Link layer to have a bus COMRESET asserted, and make a transition to the HA0: SEL/NOINTRQ state.

**Transition HA2:2:** When the Device Control register is written by the host with SRST cleared to zero, the host adapter shall notify the Transport layer to send a ContrlFIS with the current content of the shadow registers, and make a transition to the HA2: HA_NOTSEL state.

**Transition HA2:3:** When the Device Control register is written by the host with SRST set to one, the host adapter shall set BSY to one in the shadow Status register, clear DEV to zero in the shadow Device/Head register, clear IPF to zero, notify the Transport layer to send a ContrlFIS with the current content of the shadow registers, and make a transition to the HA0: HA_ SEL/NOINTRQ state.

**Transition HA2:4:** When the Command register is written by the host with any command code other than EXECUTE DEVICE DIAGNOSTIC, the host adapter shall not set BSY in the shadow Status register and shall make a transition to the HA2: HA_NOTSEL state.

**Transition HA2:5:** When the Command register is written by the host with the EXECUTE DEVICE DIAGNOSTIC command code, the host adapter shall set BSY to one in the shadow Status register, clear DEV to zero in the Device/Head register, clear IPF to zero, notify the Transport layer to send a CmdFIS with the current content of the shadow registers, and make a transition to the HA0: HA_SEL/INTRQ state.

**Transition HA2:6:** When the Device/Head register is written by the host with DEV cleared to zero and either nIEN is set to one or IPF is cleared to zero, the host adapter shall make a transition to the HA0: HA_SEL/NOINTRQ state.

**Transition HA2:7:** When the Device/Head register is written by the host with DEV cleared to zero, nIEN cleared to zero, and IPF is set to one, the host adapter shall make a transition to the HA1: HA_SEL/INTRQ state.

**Transition HA2:8:** When any register is written by the host other than those described above, the host adapter shall make a transition to the HA2: HA_NOTSEL state.

**Transition HA2:9:** When the Status or Alternate Status register is read by the host, the host shall return register content 00h to the host and make a transition to the HA2: HA_NOTSEL state.

**Transition HA2:10:** When any register is read by the host other than that described above, the host adapter shall return the current shadow register content to the host and make a transition to the HA2: HA_NOTSEL state.

**Transition HA2:11:** When the Transport layer indicates new register content from the device, the host adapter shall place the new register content into the shadow registers and make a transition to the HA2: HA_NOTSEL state.

### 5.2.5 IDENTIFY DEVICE command

In the IDENTIFY DEVICE command various parameters are communicated to the host from the device. The following sections define those words that are different from and additions to the ATA/ATAPI-5 standard definition of the data contents.

**Table 2 – IDENTIFY DEVICE information**

| Word | F/V | | |
|---|---|---|---|
| 0-46 | | Set as indicated in ATA/ATAPI-5 | |
| 47 | | | |
| | F | 15-8 | 80h |
| | R | 7-0 | 00h = Reserved<br>01h-FFh = Maximum number of sectors that shall be transferred per interrupt on READ/WRITE MULTIPLE commands |
| 48 | | Set as indicated in ATA/ATAPI-5 | |
| 49 | | Capabilities | |
| | R | 15-14 | Reserved for the IDENTIFY PACKET DEVICE command. |
| | F | 13 | 1=Standby timer values as specified in this standard are supported<br>0=Standby timer values shall be managed by the device |
| | R | 12 | Reserved for the IDENTIFY PACKET DEVICE command. |
| | F | 11 | 1=IORDY supported<br>0=IORDY may be supported |
| | F | 10 | 1=IORDY may be disabled |
| | R | 9 | Shall be set to one. Utilized by IDENTIFY PACKET DEVICE command. |
| | R | 8 | Shall be set to one. Utilized by IDENTIFY PACKET DEVICE command. |
| | X | 7-0 | Retired |
| 50-52 | | Set as indicated in ATA/ATAPI-5 | |
| 53 | R | 15-3 | Reserved |
| | F | 2 | 1=the fields reported in word 88 are valid<br>0=the fields reported in word 88 are not valid |
| | F | 1 | 1=the fields reported in words 64-70 are valid<br>0=the fields reported in words 64-70 are not valid |
| | V | 0 | 1=the fields reported in words 54-58 are valid<br>0=the fields reported in words 54-58 may be valid |
| 54-62 | | Set as indicated in ATA/ATAPI-5 | |
| 63 | | 15-3 | Set as indicated in ATA/ATAPI-5 |
| | F | 2 | 1= Multiword DMA mode 2 and below are supported |
| | F | 1 | 1= Multiword DMA mode 1 and below are supported |
| | F | 0 | 1= Multiword DMA mode 0 is supported |
| 64 | | 15-2 | Set as indicated in ATA/ATAPI-5 |
| | F | 1-0 | PIO modes 3 and 4 supported |
| 65 | | Minimum Multiword DMA transfer cycle time per word | |
| | F | 15-0 | Cycle time in nanoseconds |
| 66 | | Manufacturer's recommended Multiword DMA transfer cycle time | |
| | F | 15-0 | Cycle time in nanoseconds |
| 67 | | Minimum PIO transfer cycle time without flow control | |
| | F | 15-0 | Cycle time in nanoseconds |
| 68 | | Minimum PIO transfer cycle time with IORDY flow control | |
| | F | 15-0 | Cycle time in nanoseconds |
| 69-87 | | Set as indicated in ATA/ATAPI-5 | |
| 88 | | 15-5 | Set as indicated in ATA/ATAPI-5 |
| | F | 4 | 1=Ultra DMA mode 4 and below are supported |
| | F | 3 | 1=Ultra DMA mode 3 and below are supported |
| | F | 2 | 1=Ultra DMA mode 2 and below are supported |

| | | |
|---|---|---|
| | F 1<br>F 0 | 1=Ultra DMA mode 1 and below are supported<br>1=Ultra DMA mode 0 is supported |
| 89-92 | | Set as indicated in ATA/ATAPI-5 |
| 93 | V | COMRESET result. The contents of this word shall be set to zero. |
| 94-255 | | Set as indicated in ATA/ATAPI-5 |

Key:
F = the content of the word is fixed and does not change. For removable media devices, these values may change when media is removed or changed.
V = the contents of the word is variable and may change depending on the state of the device or the commands executed by the device.
X = the content of the word is vendor specific and may be fixed or variable.
R = the content of the word is reserved and shall be zero.

#### 5.2.5.1 Word 0 - 46: Set as indicated in ATA/ATAPI-5

#### 5.2.5.2 Word 47: Multiword PIO transfer

Bits 15 through 8 of word 47 shall be set as indicated in ATA/ATAPI-5.
Bits 7 through 0 are used to indicate the maximum number of sectors that shall be transferred per interrupt on READ/WRITE MULTIPLE commands. This field should be set to 16 or less. See section 8.5.8.1.

#### 5.2.5.3 Word 48: Set as indicated in ATA/ATAPI-5

#### 5.2.5.4 Word 49: Capabilities

Bits 15 through 12 of word 49 shall be set as indicated in ATA/ATAPI-5.

Bit 11 of word 49 is used to determine whether a device supports IORDY. This bit shall be set to one, indicating the device supports IORDY operation.

Bit 10 of word 49 is used to indicate a device's ability to enable or disable the use of IORDY. This bit shall be set to one, indicating the device supports the disabling of IORDY. Disabling and enabling of IORDY is accomplished using the SET FEATURES command.

Bits 9 through 0 of word 49 shall be set as indicated in ATA/ATAPI-5.

#### 5.2.5.5 Words 50 - 52: Set as indicated in ATA/ATAPI-5

#### 5.2.5.6 Word 53: Field validity

Bit 0 of word 53 shall be set to one, indicating the values reported in words 54 through 58 are valid. Bit 1 of word 53 shall be set to one, the values reported in words 64 through 70 are valid. Any device that supports PIO mode 3 or above, or supports Multiword DMA mode 1 or above, shall set bit 1 of word 53 to one and support the fields contained in words 64 through 70. Bit 2 of word 53 shall be set to one indicating the device supports Ultra DMA and the values reported in word 88 are valid. Bits 15-3 are reserved.

#### 5.2.5.7 Word 54 - 62: Set as indicated in ATA/ATAPI-5

#### 5.2.5.8 Word 63: Multiword DMA transfer

Bits 2 - 0 of word 63 shall be set to one indicating that the device supports Multiword DMA modes 0, 1, and 2. Bits 15 – 3 shall be set as indicated in ATA/ATAPI-5.

#### 5.2.5.9 Word 64: PIO transfer modes supported

Bits 1 - 0 of word 64 shall be set to one indicating that the device supports PIO modes 3 and 4. Bits 15 – 2 shall be set as indicated in ATA/ATAPI-5.

#### 5.2.5.10 Word 65: Minimum Multiword DMA transfer cycle time per word

Shall be set to indicate 120 ns.

#### 5.2.5.11 Word 66: Device recommended Multiword DMA cycle time

Shall be set to indicate 120 ns.

#### 5.2.5.12 Word 67: Minimum PIO transfer cycle time without flow control

Shall be set to indicate 120 ns.

#### 5.2.5.13 Word 68: Minimum PIO transfer cycle time with IORDY

Shall be set to indicate 120 ns.

#### 5.2.5.14 Words 69-87: Set as indicated in ATA/ATAPI-5

#### 5.2.5.15 Word 88: Ultra DMA modes

Bits 4 - 0 of word 88 shall be set to one indicating that the device supports Ultra DMA modes 0, 1, 2, 3, and 4. Bits 15 – 5 shall be set as indicated in ATA/ATAPI-5.

#### 5.2.5.16 Word 89 - 92: Set as indicated in ATA/ATAPI-5

#### 5.2.5.17 Word 93: Hardware configuration test results

Word 93 shall be set to 0000h indicating that the word is not supported.

#### 5.2.5.18 Words 94-255: Set as indicated in ATA/ATAPI-5

# 6 Physical layer

## 6.1 Overview

This section describes the physical layer of Serial ATA. The information that is provided is comprised of two types – informative and normative. Unless otherwise described, the information should be considered normative in nature and is included in this document as a necessary requirement in order to properly allow a piece of equipment to attach to another piece of equipment. The normative information is deliberately structured to constrain and define areas only to the degree that is required for compatibility. The information that is provided and marked informative is provided only to help the reader better understand the normative sections and should be taken as examples only. Exact implementations may vary.

## 6.2 List of services

- Transmit a 1.5 Gb/sec differential NRZ serial stream at specified voltage levels
- Provide a 100 Ohm matched termination (differential) at the transmitter
- Serialize a 10, 20, 40, or other width parallel input from the Link for transmission
- Receive a 1.5 Gb/sec +350/- 2650 ppm differential NRZ serial stream
- Provide a 100 Ohm matched termination (differential) at the receiver
- Extract data (and, optionally, clock) from the serial stream
- Deserialize the serial stream
- Detect the K28.5 comma character and provide a bit and word aligned 10, 20, 40, or other width parallel output
- Provide specified OOB signal detection and transmission
- Perform proper power-on sequencing and speed negotiation
- Provide device status to Link layer
  - device present
  - device absent
  - device present but failed to negotiate communications
- Optionally support power management modes (note: if not supported, power management requests must be properly rejected by the Link layer)
- Optionally perform transmitter and receiver impedance calibration (if manufacturing tolerances require)
- Handle the input data rate frequency variation due to a spread spectrum transmitter clock (Nominal +0/-0.5% slow frequency variation, over a 33.33 us up/down triangular wave period).
- Accommodate request to go into Far-End retimed loopback test mode of operation when commanded (exit this mode on OOB COMRESET/COMINIT)

## 6.3 Cables and connectors specifications

### 6.3.1 Overview

This section covers Serial ATA connectors and cable assemblies. It defines the

- Connector mating interfaces
- Connector location on the device
- Electrical, mechanical and reliability requirements of the connectors and cable assemblies
- Connector and cable testing procedures

It, however, does not define how the connector and cable assembly should be implemented, such as

- The mounting feature of the connectors
- The cabling and cable terminations
- The methods on how the PCB connects to other components of the system

### 6.3.2 Objectives

The basic guidelines for Serial ATA connectors and cables are

- Supporting 1.5 Gbps date rate with headroom for 3.0 Gbps speed
- Cost competitive to Ultra ATA connector and cable
- Facilitating smooth transition from Ultra ATA to Serial ATA
- Common connector interface for both 2.5” and 3.5” devices

From those guidelines, the following connector and cable objectives are derived:

- Minimal discontinuity at connectors
- Good impedance control for cable
- 0 to 1 meter cable length
- Low profile, fitting in the 2.5” drive
- Blind-mateable
- Hot-pluggable by means of staggered contacts
- Supporting power delivery with 12.0 V, 5.0 V, and 3.3 V voltages

### 6.3.3 General descriptions

A Serial ATA device may be either directly connected to a host or connected to a host through a cable.

For direct connection, the device plug connector, shown as (a) and (b) in Figure 5, is inserted directly into a host receptacle connector, illustrated as (g) in Figure 5. The device plug connector and the host receptacle connector incorporate features that enable the direct connection to be hot pluggable and blind mateable.

For connection via cable, the device signal plug connector, shown as (a) in Figure 5, mates with the signal cable receptacle connector on one end of the cable, illustrated as (c) in Figure 5. The signal cable receptacle connector on the other end of the cable is inserted into a host signal plug connector, shown as (f) in Figure 5. The signal cable wire consists of two twinax sections in a common outer sheath.

Besides the signal cable, there will be also a separate power cable for the cabled connection. A Serial ATA power cable includes a power cable receptacle connector, shown as (d) in Figure 5, on one end and may be directly connected to the host power supply on the other end or may include a power cable receptacle on the other end. The power cable receptacle connector on one end of the

power cable mates with the device power plug connector, shown as (b) in Figure 5. The other end of the power cable is attached to the host as and however necessary.

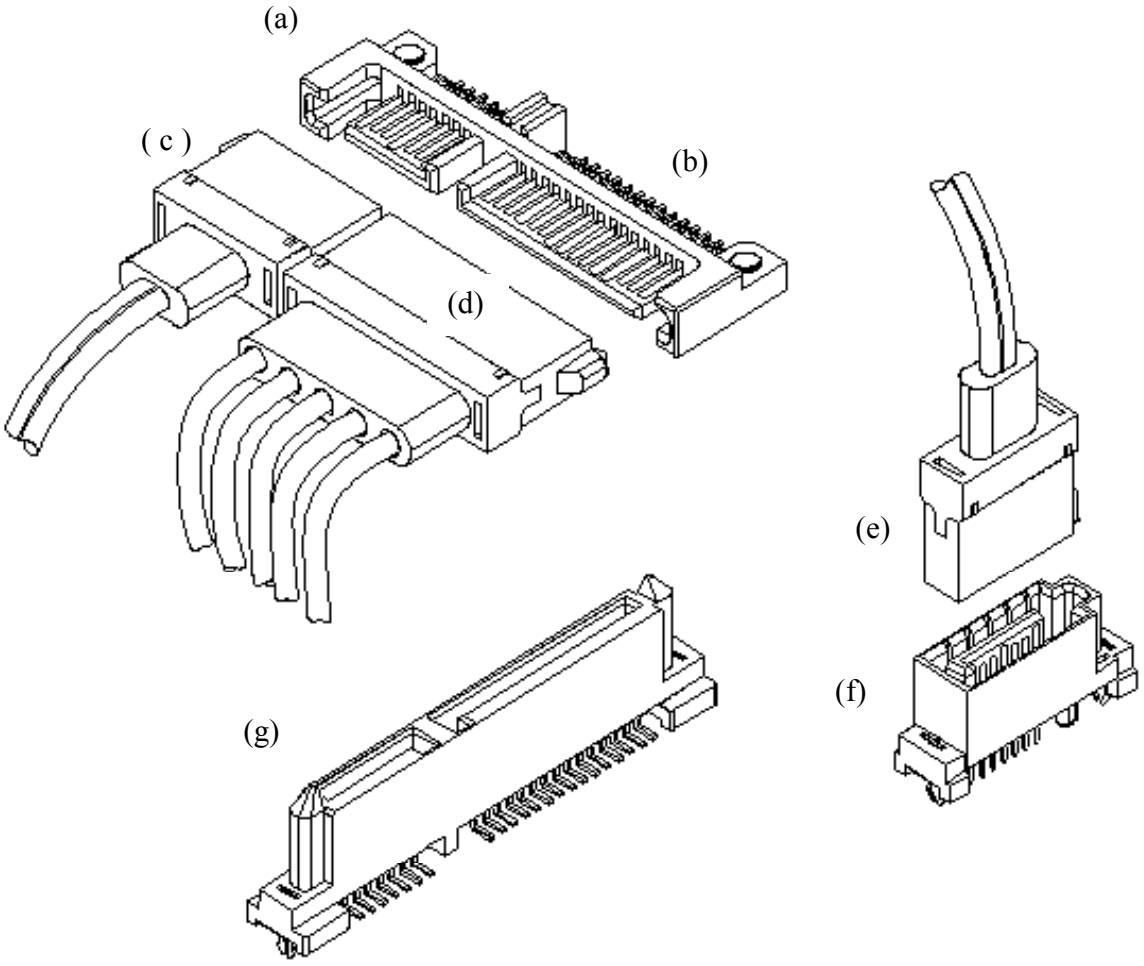



**Figure 5 – Serial ATA connector examples**

Illustrated above: (a) device signal plug segment or connector; (b) device power plug segment or connector; (c) signal cable receptacle connector, to be mated with (a); (d) power cable receptacle connector, to be mated with (b); (e) signal cable receptacle connector, to be mated with (f), the host signal plug connector; (g) host receptacle connector mating directly with device plug connector (a) & (b).

Figure 6 illustrates how signals and grounds are assigned in direct connect and cabled configurations.

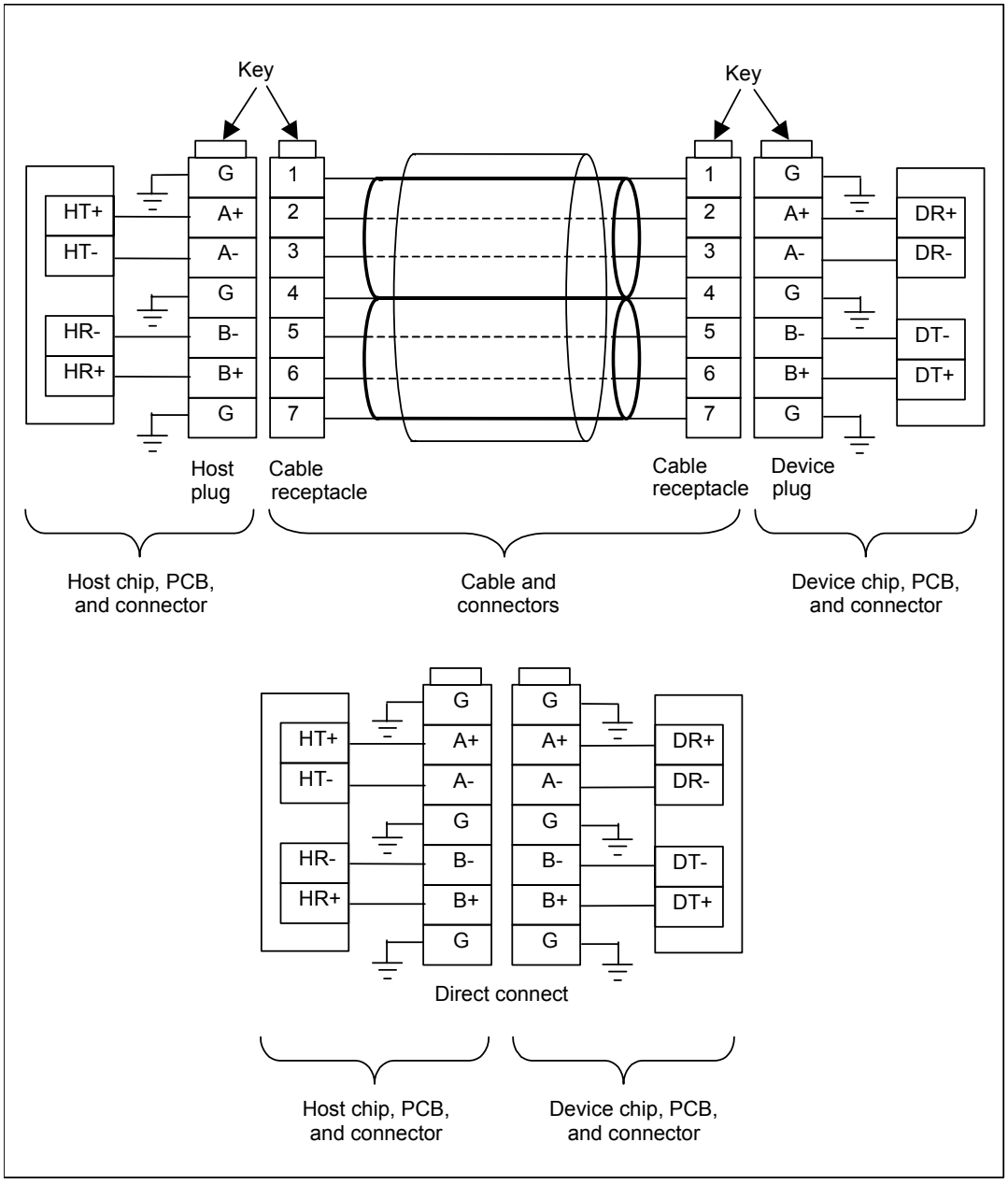


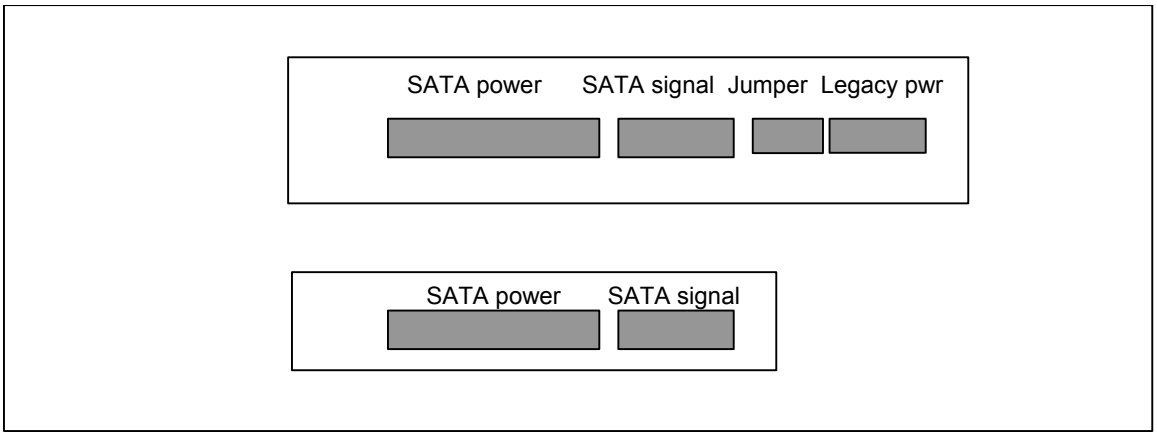

**Figure 6 - Signals and grounds assigned in direct connect and cabled configurations**

Illustrated above: G: ground; A+/A- and B+/B-: connector differential pairs; HT+/HT-: host transmitter signal pair; HR+/HR-: host receiver signal pair; DT+/DT-: device transmitter signal pair; DR+/DR-: device receiver signal pair.

### 6.3.4 Connector configurations and locations

#### 6.3.4.1 Configurations

Serial ATA defines several connector configurations on devices. The figures below illustrate the reference connector configurations for 3.5” and 2.5” form factor devices. Refer to Appendix C for all the defined connector configurations on devices.




The reference connector configuration for the 3.5” device includes the Serial ATA connector with both signal and power segments, and the legacy jumper and power connectors for smooth transition from Parallel ATA.

The reference configuration for the 2.5” device is the only configuration allowed for the 2.5" form factor.

#### 6.3.4.2 Locations

The device connector location is defined to facilitate blind mating.

Figure 7 and Figure 8 define the connector locations on 3.5” and 2.5” devices, respectively.

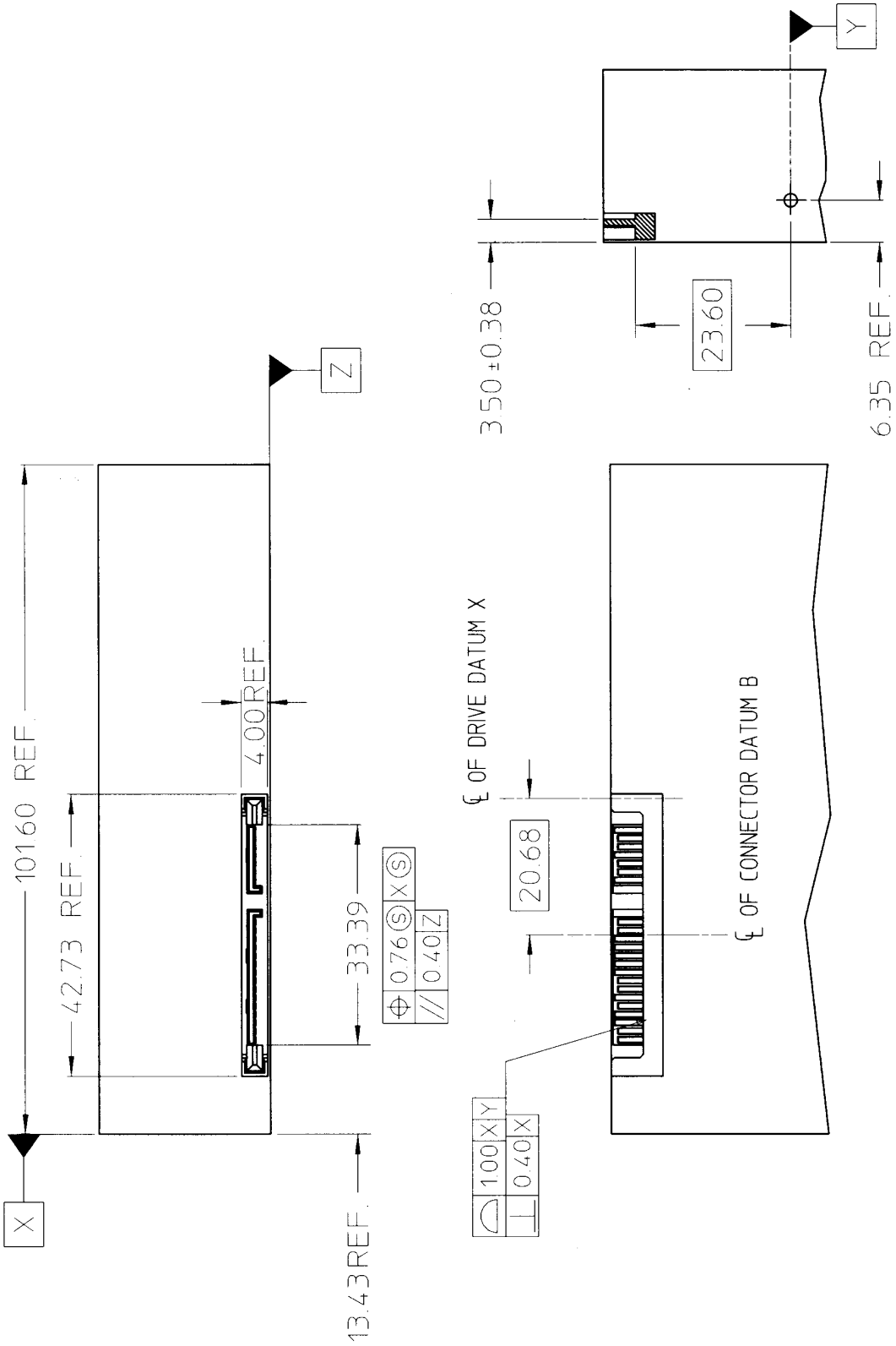



**Figure 7 – Device plug connector location on 3.5” device**

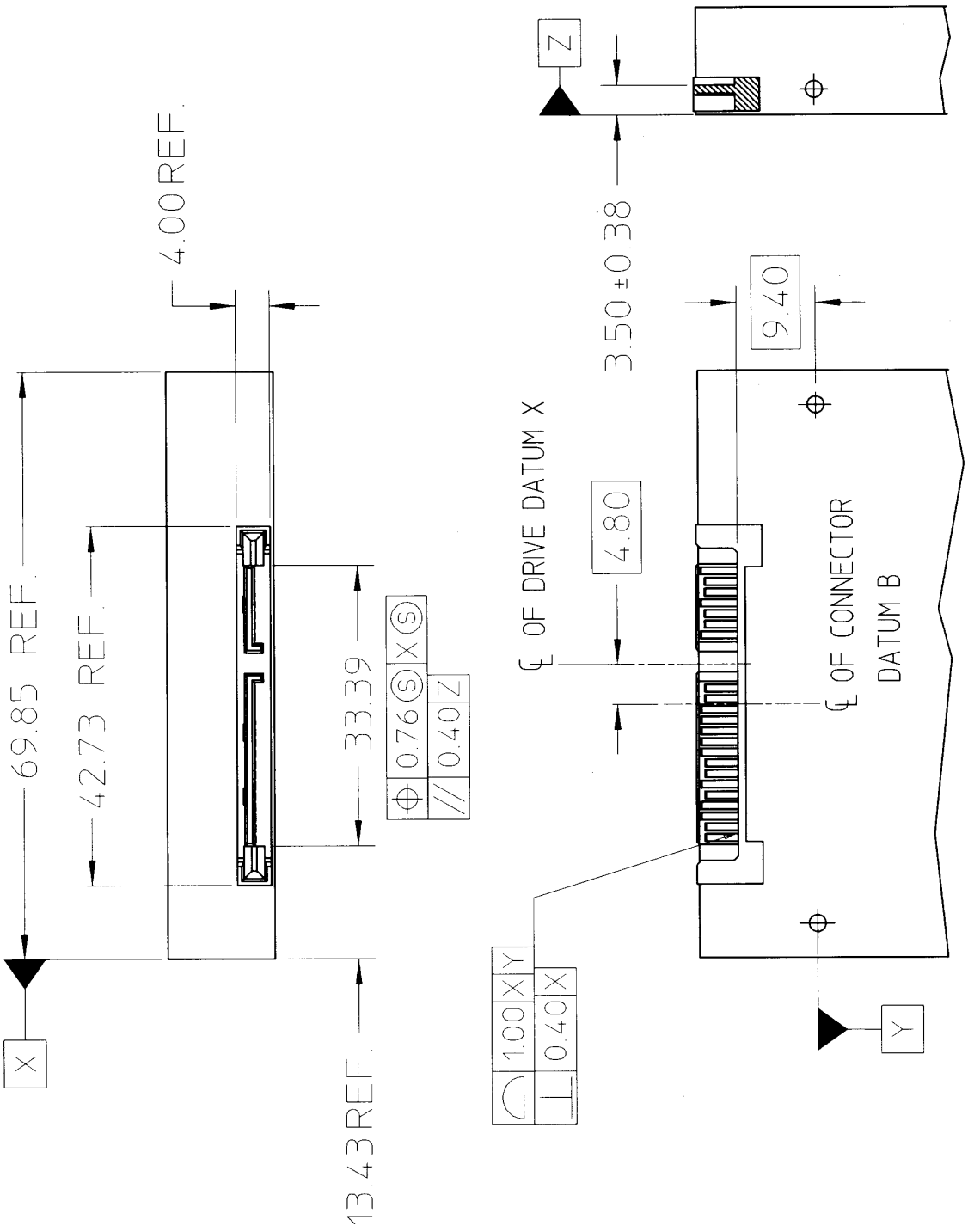



**Figure 8 – Device plug connector location on 2.5” device**

### 6.3.5 Mating interfaces

Serial ATA connectors are defined in terms of their mating interface or front end characteristics only. Connector back end characteristics including PCB mounting features and cable termination features are not defined.

#### 6.3.5.1 Device plug connector

Figure 9 and Table 3 show the interface dimensions for the device plug connector with both signal and power segments.

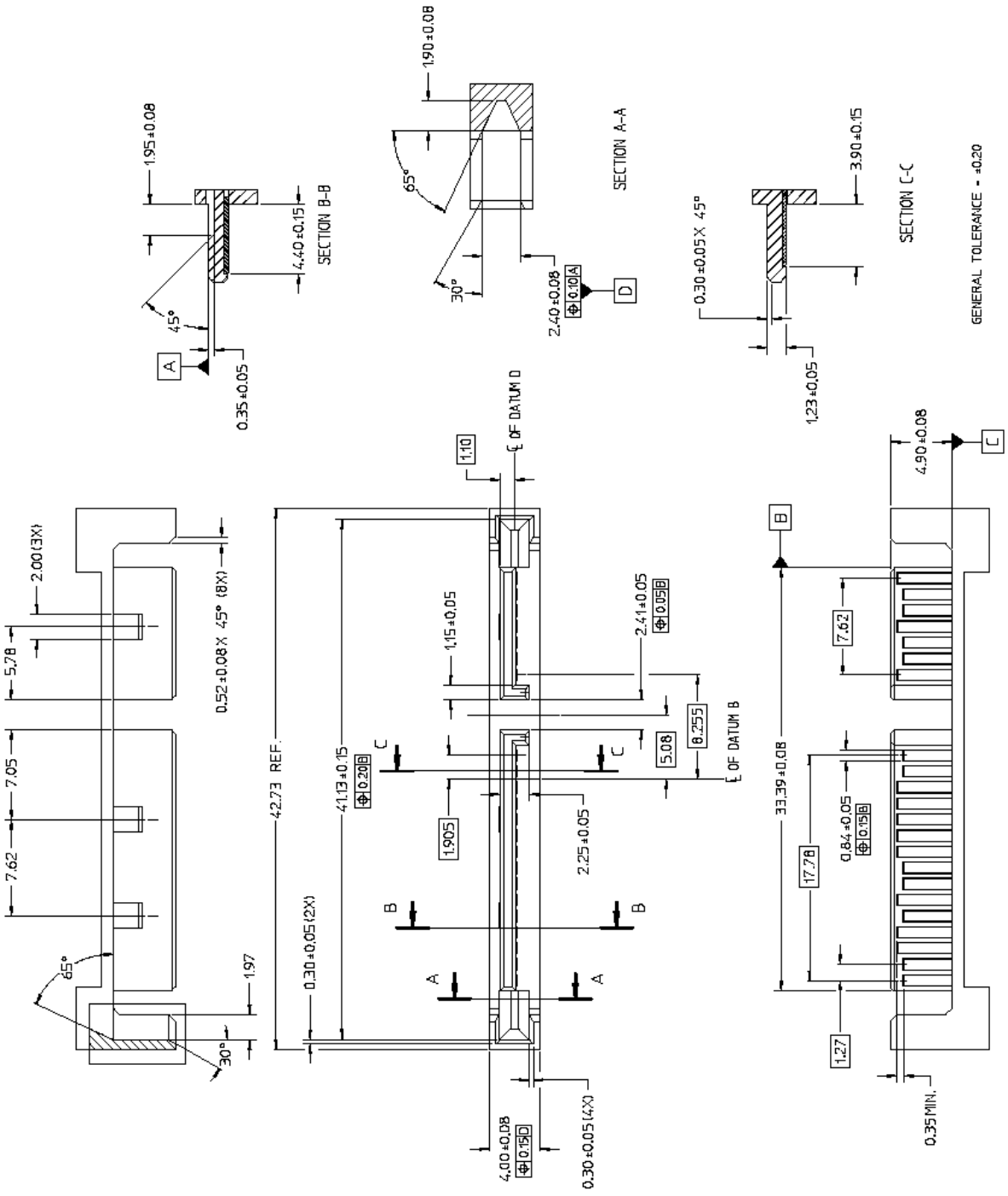



**Figure 9 – Device plug connector interface dimensions**

There are total of 7 pins in the signal segment and 15 pins in the power segment. The pin definitions are shown in Table 3. Note that the pin is numbered from the pin furthest from the power segment.

**Table 3 – Device plug connector pin definition**

| | | | |
|---|---|---|---|
| Signal Segment Key | | | |
| Signal segment | S1 | Gnd | 2nd mate |
| | S2 | A+ | Differential signal pair A from Phy |
| | S3 | A- | |
| | S4 | Gnd | 2nd mate |
| | S5 | B- | Differential signal pair B from Phy |
| | S6 | B+ | |
| | S7 | Gnd | 2nd mate |
| Signal Segment "L" | | | |
| Central Connector Polarizer | | | |
| Power Segment "L" | | | |
| Power segment | P1 | $V_{33}$ | 3.3 V power |
| | P2 | $V_{33}$ | 3.3 V power |
| | P3 | $V_{33}$ | 3.3 V power, pre-charge, 2nd mate |
| | P4 | Gnd | 1st mate |
| | P5 | Gnd | 2nd mate |
| | P6 | Gnd | 2nd mate |
| | P7 | $V_5$ | 5 V power, pre-charge, 2nd mate |
| | P8 | $V_5$ | 5 V power |
| | P9 | $V_5$ | 5 V power |
| | P10 | Gnd | 2nd mate |
| | P11 | Reserved | 1. The pin corresponding to P11 in the backplane receptacle connector is also reserved<br>2. The corresponding pin to be mated with P11 in the power cable receptacle connector shall always be grounded |
| | P12 | Gnd | 1st mate |
| | P13 | $V_{12}$ | 12 V power, pre-charge, 2nd mate |
| | P14 | $V_{12}$ | 12 V power |
| | P15 | $V_{12}$ | 12 V power |
| Power Segment Key | | | |

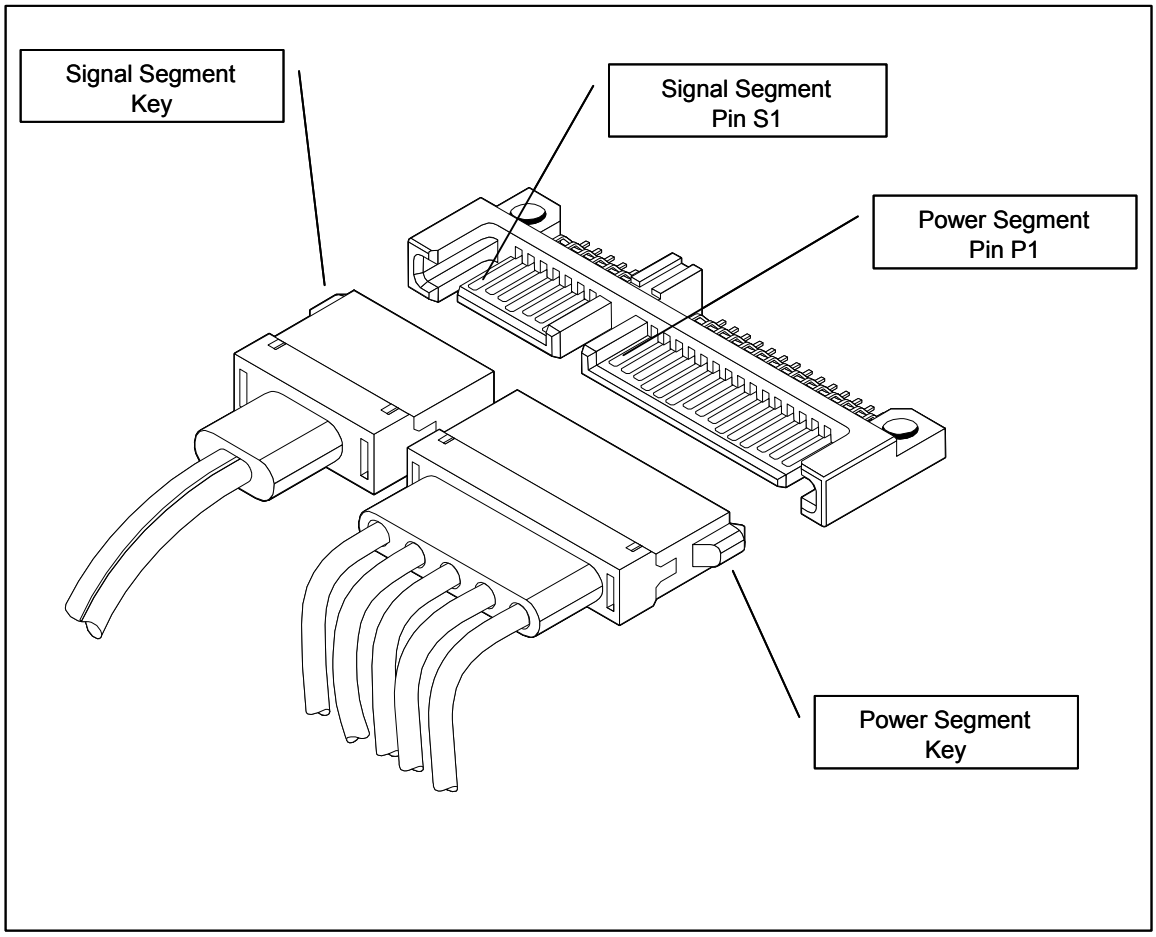



**Figure 10 Connector Pin and Feature Locations**

The following points should be noted:

- All pins are in a single row, with a 1.27 mm (.050”) pitch.
- The comments on the mating sequence in Table 3 apply to the case of backplane blind-mate connector only. In this case, the mating sequences are: (1) the ground pins P4 and P12; (2) the pre-charge power pins and the other ground pins; and (3) the signal pins and the rest of the power pins.
- There are three power pins for each voltage. One pin from each voltage is used for pre-charge in the backplane blind-mate situation.
- If a device uses 3.3 V, then all $V_{33}$ pins must be terminated. Otherwise, it is optional to terminate any of the $V_{33}$ pins.
- If a device uses 5.0 V, then all $V_5$ pins must be terminated. Otherwise, it is optional to terminate any of the $V_5$ pins.
- If a device uses 12.0 V, then all $V_{12}$ pins must be terminated. Otherwise, it is optional to terminate any of the $V_{12}$ pins.

### 6.3.5.2 Signal cable receptacle connector

Figure 11 shows the interface dimensions for the signal cable receptacle connector. There are two identical receptacles at the two ends of the Serial ATA cable assembly. The cable receptacle mates with either the signal segment of the device plug connector on the device, or the host plug connector on the host.

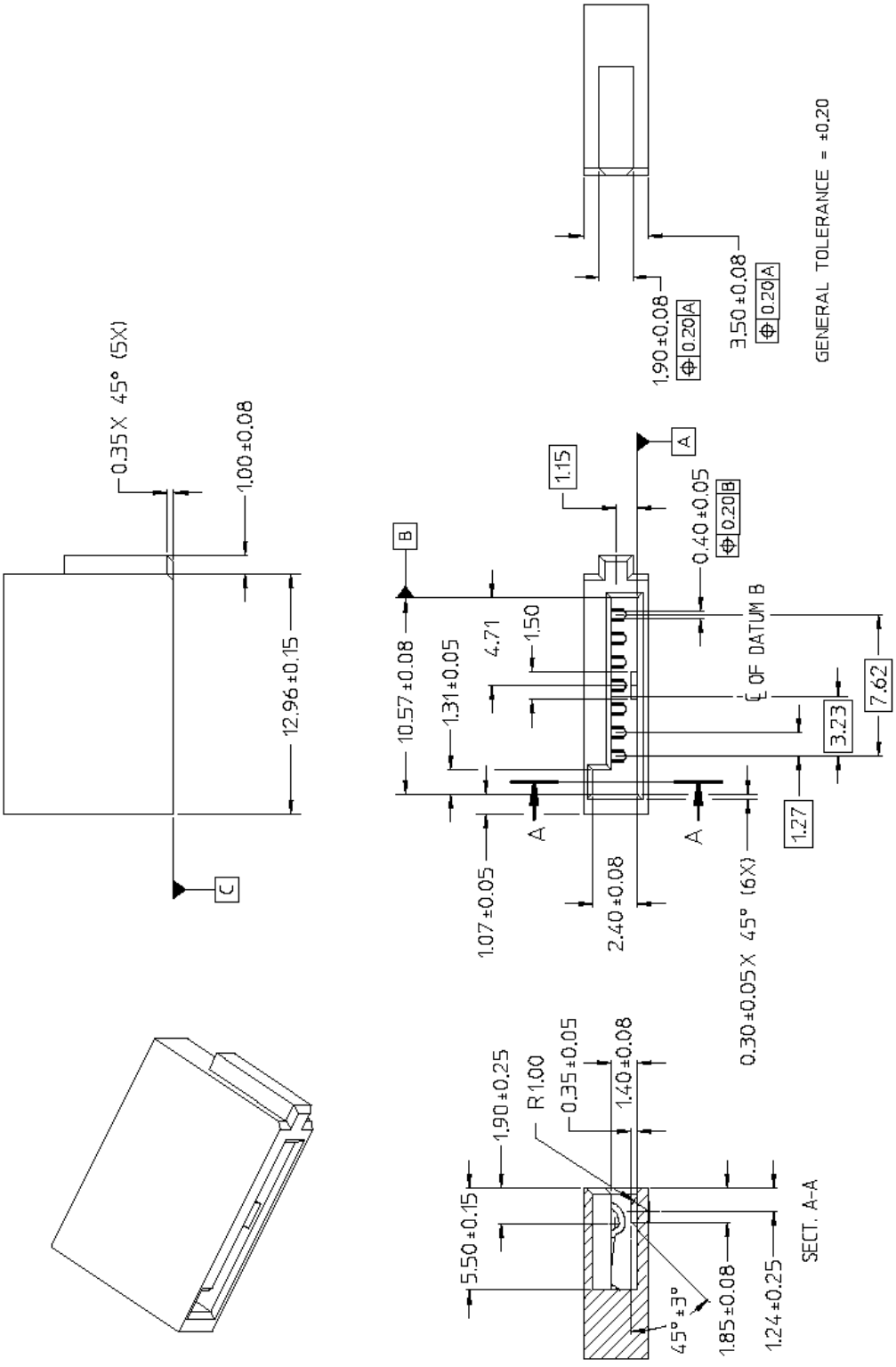



**Figure 11 – Cable receptacle connector interface dimensions**

The pin out of the cable receptacle connector is the mirror image of the signal segment of the device plug connector. Notice that

- The two differential pin pairs are terminated with the corresponding differential cable pairs
- The ground pins are terminated with the cable drain wires, if it applies
- The choice of cable termination methods, such as crimping or soldering is up to each connector vendor

#### 6.3.5.3 Signal host plug connector

The signal host plug connector is to be mated with one end of the Serial ATA cable assembly. So the pinout of the host plug connector is the mirror image of the signal cable receptacle. Figure 12 shows the host plug connector interface definition.

For applications where multiple Serial ATA ports or connectors are stacked together on the host, there is a clearance or spacing requirement to prevent the cable assemblies from interfering with each other. Figure 13 shows the recommended clearance or spacing.

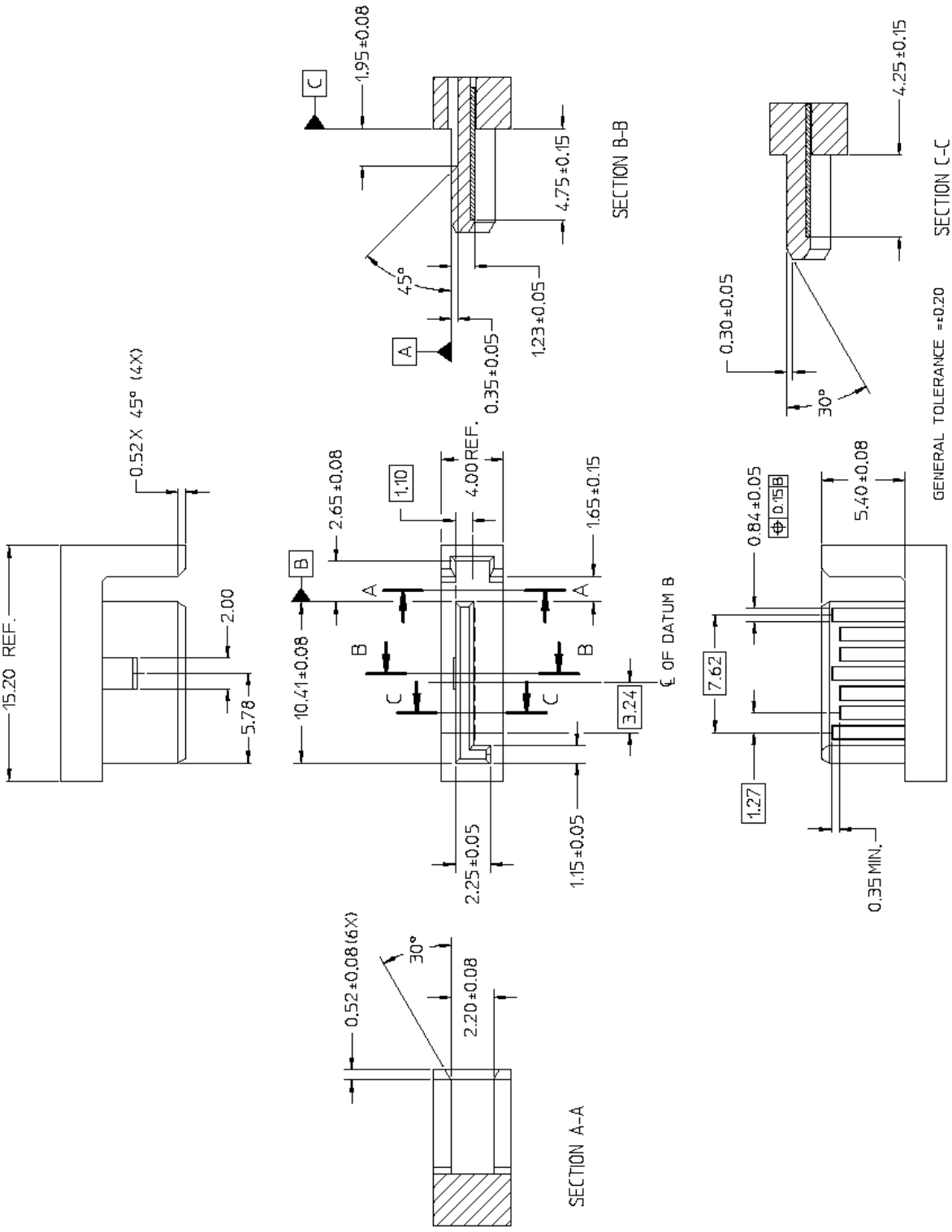



**Figure 12 – Host plug connector interface dimension**

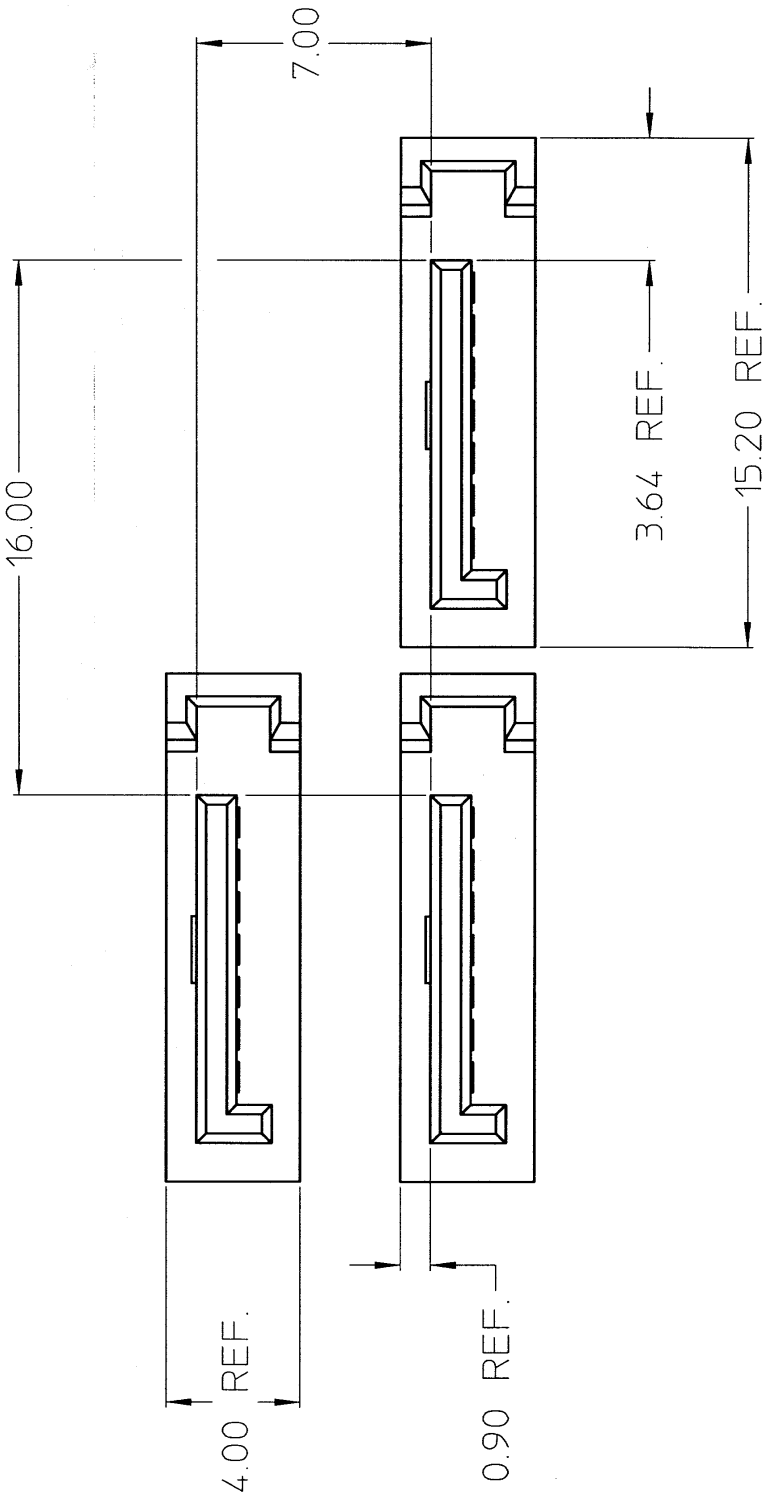



**Figure 13 – Recommended host plug connector clearance or spacing**

#### 6.3.5.4 Host receptacle connector

The host receptacle connector is to be blind-mated directly with the device plug connector. The interface dimensions for the host receptacle connector are shown in Figure 14. Note that dimension B allows two values: 8.15mm and 14.15mm. There are two levels of contacts in the host receptacle connector. The advancing ground contacts P4 and P12 mate first with the corresponding ground pins on the device plug connector, followed by the engaging of the pre-charged power pins. An appropriate external retention mechanism independent of the connector is required to keep the host PCB and the device in place. The host receptacle connector is not designed with any retention mechanism.

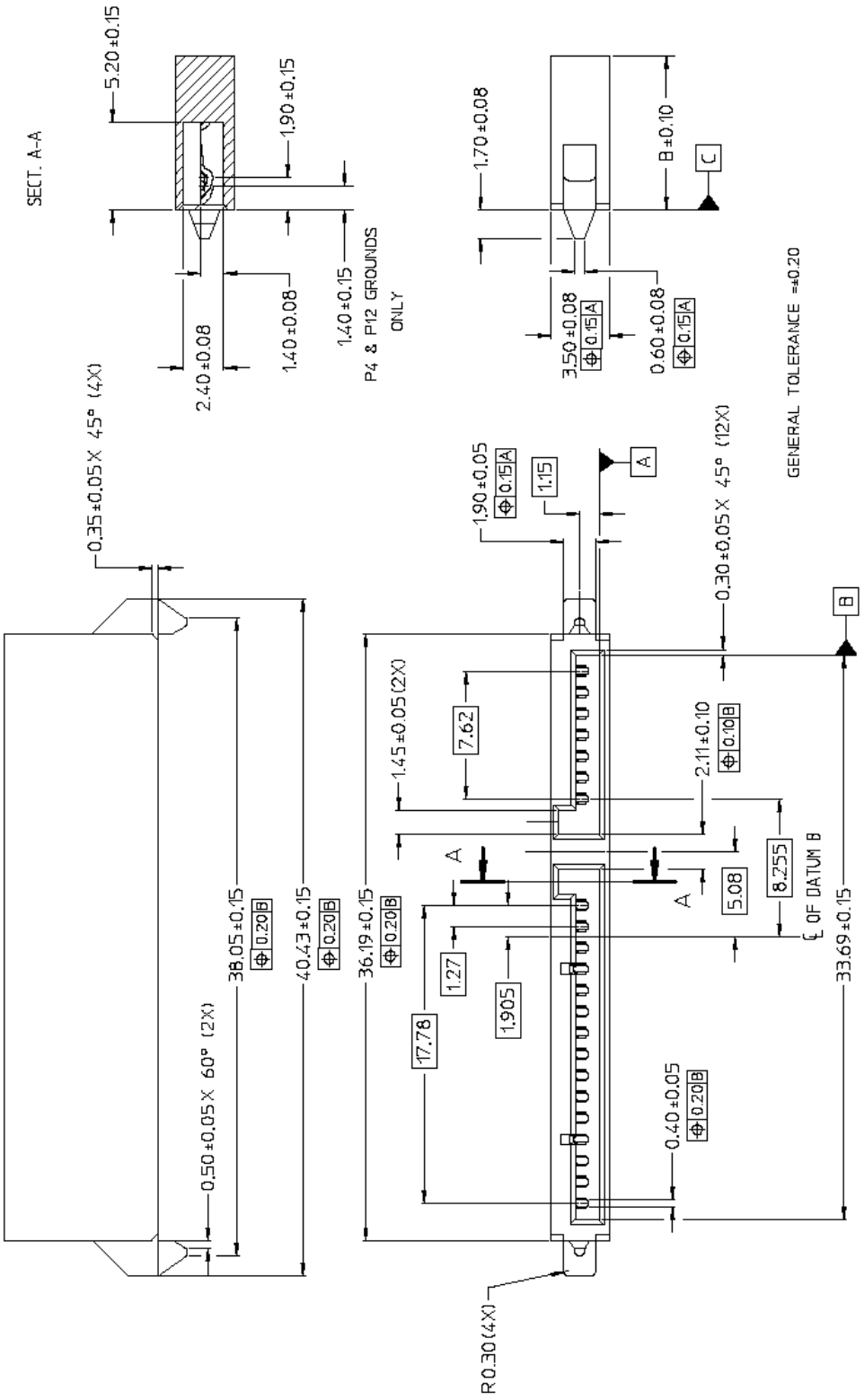



**Figure 14 – Host receptacle connector interface dimensions**

#### 6.3.5.5 Power cable receptacle connector

The power cable receptacle connector mates with the power segment of the device plug, bringing power to the device. Figure 15 shows the interface dimensions of the power receptacle connector. The pinout of the connector is the mirror image of the power segment of the device plug shown in Table 3.

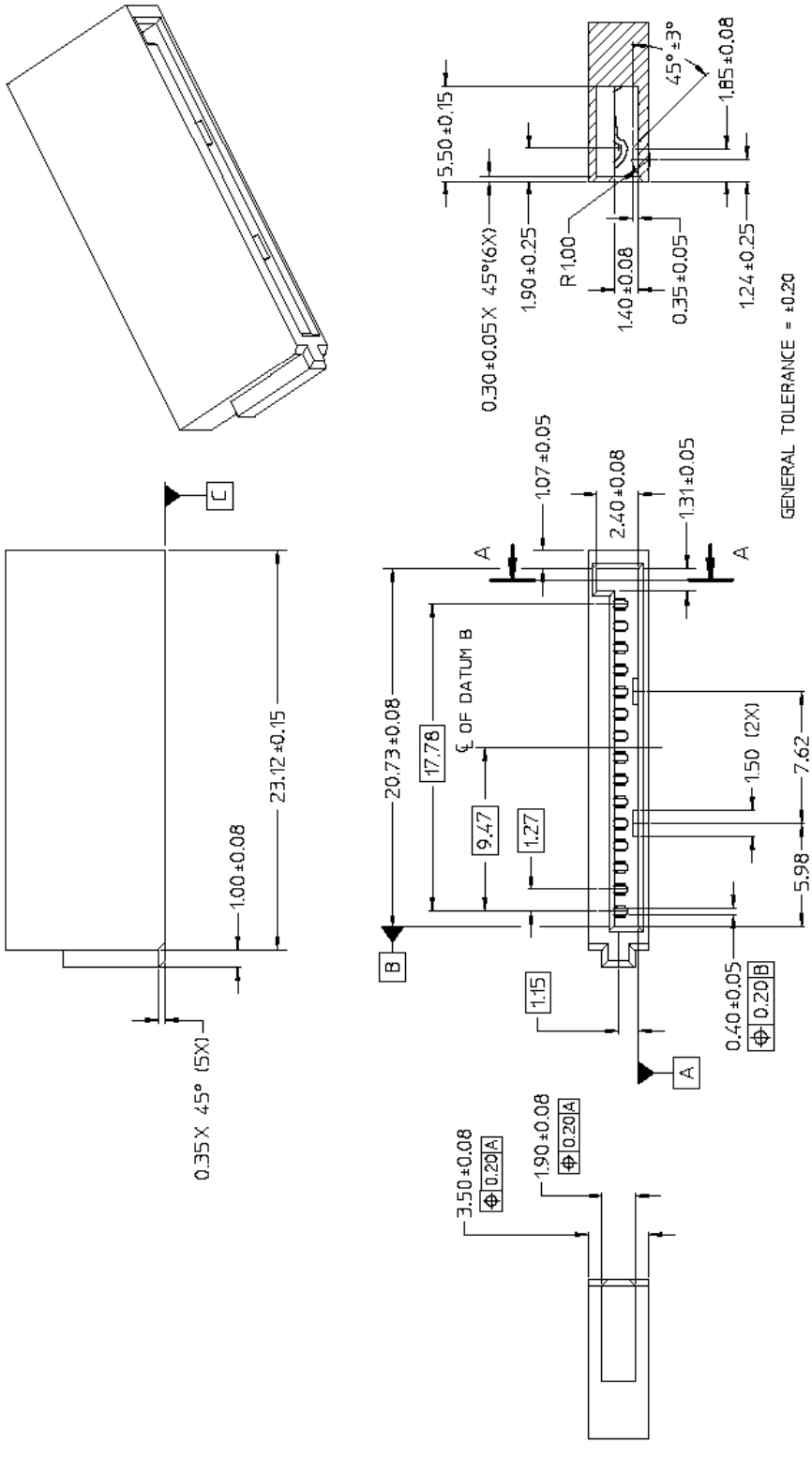



**Figure 15 – Power receptacle connector interface dimensions**

The power receptacle connector is terminated onto 18 AWG wires that are connected to the system power supply or other power sources. Five 18 AWG wires may be used, with three wires terminated to the nine power pins for the three voltages, while the remaining two wires to the six ground pins.

### 6.3.6 Serial ATA cable

The Serial ATA cable consists of four conductors in two differential pairs. If necessary, the cable may also include drain wires to be terminated to the ground pins in the Serial ATA cable receptacle connectors. The cable size may be 30 to 26 AWG. The cable maximum length is one meter.

This specification does not specify a standard Serial ATA cable. Any cable that meets the electrical requirements to be described later in this section is considered an acceptable Serial ATA cable. The connector and cable vendors have the flexibility to choose cable constructions and termination methods based on performance and cost considerations. An example cable construction is given in Appendix J.2 for an informational purpose only.

### 6.3.7 Backplane connector configuration and blind-mating tolerance

The maximum blind-mate misalignment tolerances are ±1.50 mm and ±1.00 mm, respectively, for two perpendicular axes illustrated in Figure 16. Any skew angle of the plug, with respect to the receptacle, will reduce the blind-mate tolerances.

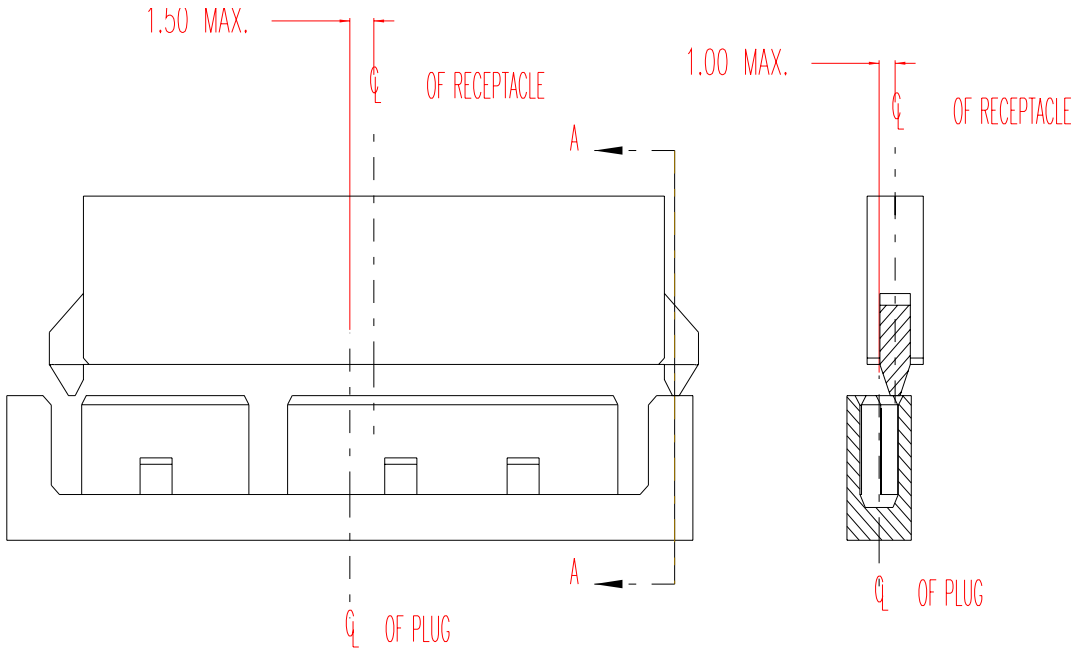



**Figure 16 – Connector pair blind-mate misalignment tolerance**

The device-to-backplane mating configuration is shown in Figure 17 - Device-backplane mating configuration. Note that two values (8.45 and 14.45 mm) are allowed for dimension A

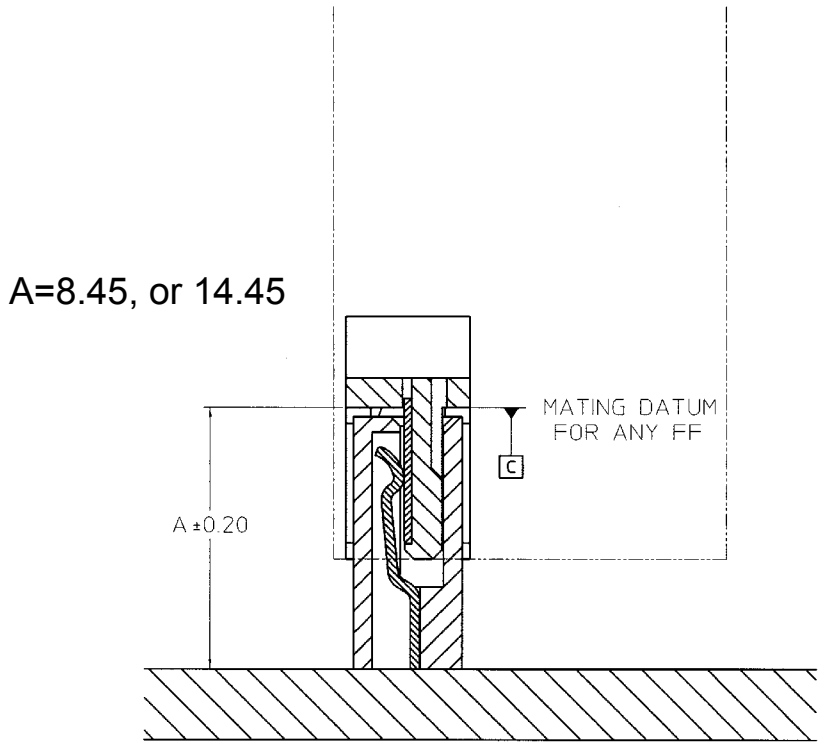



**Figure 17 - Device-backplane mating configuration**

### 6.3.8 Connector labeling

Labeling on a connector with the connector manufacturing logo or Serial ATA logo is optional.

### 6.3.9 Connector and cable assembly requirements and test procedures

Unless otherwise specified, all measurements shall be performed within the following lab conditions:

- Mated
- Temperature: 15° to 35° C
- Relative Humidity: 20% to 80%
- Atmospheric Pressure: 650 mm to 800 mm of Hg

If an EIA (Electronic Industry Association) test is specified without a letter suffix in the test procedures, the latest approved version of that test shall be used.

#### 6.3.9.1 Signal

The test board shall consist of differential traces (100 ohm+-5 ohm) over a ground plane (single ended 50 ohm +/-2.5 ohm).

Open or shorted traces with the same length as the input signal traces shall be provided to measure the system input risetime and to synchronize pulses. Traces for crosstalk measurements will diverge from each other. Provisions for attenuation reference measurement shall also be provided.

Unless otherwise specified, the requirements are for the entire signal path from the host connector mated pair to the device plug mated pair connector, but not including PCB traces.

A cable assembly shall meet Table 4 electrical signaling parameters and requirements when tested with the above specified test fixture or equivalent.

**Table 4 – Signal integrity requirements and test procedures**

| Parameter | Procedure | Requirements |
|---|---|---|
| Mated connector impedance | 1. Minimize skew (see NOTE 1).<br>2. Set the Time Domain Reflectometer (TDR) pulsers in differential mode with a positive going pulse (V+) and a negative going pulse (V-). Define a reflected differential trace: Vdiff = V+ - V-.<br>3. With the TDR connected to the risetime reference trace, verify an input risetime of 70 ps (measured 20%-80% Vp). Filtering may be used to slow the system down (see NOTE 2).<br>4. Connect the TDR to the sample measurement traces. Calibrate the instrument and system (see NOTE 3).<br>5. Measure and record the maximum and minimum values of the near end connector impedance. | 100 ohm +/- 15% |
| Cable absolute impedance | 1. Minimize skew (see NOTE 1).<br>2. Set the Time Domain Reflectometer (TDR) pulsers in differential mode with a positive going pulse (V+) and a negative going pulse (V-). Define a reflected differential trace: Vdiff = V+ - V-.<br>3. With the TDR connected to the risetime reference trace verify an input risetime of 70 ps (measured 20%-80% Vp). Filtering may be used to slow the system down (see NOTE 2).<br>4. Connect the TDR to the sample measurement traces. Calibrate the instrument (see NOTE 3).<br>5. Measure and record maximum and minimum cable impedance values in the first 500 ps of cable response following any vestige of the connector response. | 100 ohm +/- 10% |

| Parameter | Procedure | Requirements |
|---|---|---|
| Cable pair matching | 1. Set the Time Domain Reflectometer (TDR) to differential mode.<br>2. With the TDR connected to the risetime reference traces verify an input risetime of 70 ps (measured 20%-80% Vp). Filtering may be used to slow the system down (see NOTE 2).<br>3. Connect the TDR to the sample measurement traces. Calibrate the instrument and system (see NOTE 3).<br>4. Measure and record the single–ended cable impedance of each cable within a pair Measure and record maximum and minimum cable impedance values in the first 500 ps of cable response following any vestige of the connector response.<br>5. The parameter then equals $Line1_{imp}$ - $Line2_{imp}$ . | +/- 5 ohm |
| Common mode impedance | 1. Set two TDR pulsers to produce a differential signal.<br>2. Minimize skew (see NOTE 1).<br>3. With the TDR connected to the risetime reference trace verify an input risetime of 70 ps (measured 20-80%) Filtering may be used to slow the system down (see NOTE 2).<br>4. Calibrate the TDR (see NOTE 3).<br>5. Set both TDR pulsers to produce positive going pulses.<br>6. Measure the even mode impedance of the first pulser. Divide this by 2 to get the common mode impedance.<br>7. Do the same for the other pulser. Both values shall meet the requirement. | 25 to 40 ohms |

| Parameter | Procedure | Requirements |
|---|---|---|
| Insertion loss | 1. Produce a differential signal with the signal source (see NOTE 4).<br>2. Assure that skew between the pairs is minimized. (see NOTE 1).<br>3. Measure and store the insertion loss (IL) of the fixturing, using the IL reference traces provided on the board, over a frequency range of 10 to 4500 MHz.<br>4. Measure and record the IL of the sample, which includes fixturing IL, over a frequency range of 10 to 4500 MHz.<br>5. The IL of the sample is then the results of procedure 4 minus the results of Procedure 3. | 6 dB max |
| Crosstalk: NEXT | 1. Produce a differential signal with the signal source (see NOTE 1).<br>2. Connect the source to the risetime reference traces. Assure that skew between the pairs is minimized. (see NOTE 1).<br>3. Terminate the far ends of the reference trace with loads of characteristic impedance.<br>4. Measure and record the system and fixturing crosstalk. This is the noise floor.<br>5. Terminate the far ends of the drive and listen lines with loads of characteristic impedance.<br>6. Connect the source to the drive pair and the receiver to the near-end of the listen pair.<br>7. Measure the NEXT over a frequency range of 10 to 4500 MHz.<br>8. Verify that the sample crosstalk is out of the noise floor. | -26 dB |

| Parameter | Procedure | Requirements |
|---|---|---|
| Rise time | 1. Minimize skew (see NOTE 1).<br>2. Set the Time Domain Reflectometer (TDR) pulsers in differential mode with a positive going pulse (V+) and a negative going pulse (V-). Define a reflected differential trace on the receive channels as: Vdiff = V + -V-.<br>3. With the TDR connected to the risetime reference trace measure and record the input risetime. Verify that the input risetime is between 25-35 ps (measured 20%-80% Vp) see NOTE 2.<br>4. Remove the reflected trace definition.<br>5. Connect the TDR to the sample measurement traces.<br>6. Define a differential trace on the receive channels as: Vdiff = V + -V-.<br>7. Measure (measured 20%-80% Vp) and record the output risetime. | 85 ps |
| Inter-symbol Interference | K – 28.5 signal source running at 1.5 Gb/sec. The average position of zero crossing should not move more than specified value. | 50 ps maximum |

| Parameter | Procedure | Requirements |
|---|---|---|
| Intra-Pair Skew | 1. Set one of the Time Domain Reflectometer (TDR) pulsers in differential mode with a positive going pulse (V+) and a negative going pulse (V-).<br>2. With the TDR connected to the risetime reference trace verify an input risetime of 70 ps (measured 20%-80% Vp). Filtering may be used to slow the system down (see NOTE 2).<br>3. Measure propagation delay (50% of Vp) of each line in a pair single-endedly. The skew equals the difference between each single ended propagation delay. | 10 ps max |
| NOTES –<br>1: Skew must be minimized. Time domain measurement equipment allows for delay adjustment of the pulses so launch times can be synchronized. Frequency domain equipment will require the use of phase matched fixturing. The fixturing skew should be verified to be <1 ps on a TDR.<br>2: The system risetime is to be set via equipment filtering techniques. The filter risetime is significantly close to the stimulus risetime. Therefore the filter programmed equals the square root of ($t_{r(observed)}$ )squared – ($t_{r(stimulus)}$)squared. After filtering, verify the risetime is achieved using the risetime reference traces on the PCB fixture.<br>3: Calibrate the system by substituting either precision 50 ohm loads or precision air lines (also terminated in 50 ohm loads) for the test fixture. This places the calibration plane directly at the input interface of the test fixture.<br>4: A network analyzer is preferred. If greater dynamic range is required a signal generator/spectrum analyzer may be used. Differential measurements require the use of a four port network analyzer although baluns or hybrid couplers may be used. | | |

### 6.3.9.2 Housing and contact electrical requirements

Table 5 is the connector housing and contact electrical requirements.

**Table 5 – Housing and contact electrical parameters, test procedures, and requirements**

| Parameter | Procedure | Requirement |
|---|---|---|
| Insulation resistance | EIA 364-21<br>After 500 VDC for 1 minute, measure the insulation resistance between the adjacent contacts of mated and unmated connector assemblies. | 1000 MΩ minimum |
| Dielectric withstanding voltage | EIA 364-20 Method B<br>Test between adjacent contacts of mated and unmated connector assemblies. | The dielectric shall withstand 500 VAC for 1 minute at sea level. |
| Low level contact resistance (LLCR) | EIA 364-23<br>Subject mated contacts assembled in housing to 20 mV maximum open circuit at 100 mA maximum | • Initially 30 mΩ maximum.<br>• Resistance increase 15 mΩ maximum after stress |
| Contact current rating (Power segment) | • Mount the connector to a test PCB<br>• Wire power pins P1, P2, P8, and P9 in parallel for power<br>• Wire ground pins P4, P5, P6, P10, and P12 in parallel for return<br>• Supply 6 A total DC current to the power pins in parallel, returning from the parallel ground pins (P4, P5, P6, P10, and P12)<br>• Record temperature rise when thermal equilibrium is reached | 1.5 A per pin minimum. The temperature rise above ambient shall not exceed 30° C at any point in the connector when contact positions are powered. The ambient condition is still air at 25° C. |

### 6.3.9.3 Mechanical and environmental requirements

Table 6 lists the mechanical parameters and requirements, while Table 7 the environmental and reliability tests and requirements:

**Table 6 – Mechanical test procedures, and requirements**

| Test description | Procedure | Requirement |
|---|---|---|
| Visual and dimensional inspections | EIA 364-18<br>Visual, dimensional and functional per applicable quality inspection plan. | Meets product drawing requirements. |
| Cable pull-out | EIA 364-38 Condition A<br>Subject a Serial ATA cable assembly to a 40 N axial load for a min of one minute while clamping one end of the cable plug. | No physical damage |
| Cable flexing | For round cable: EIA 364-41 Condition I<br>Dimension x=3.7 x cable diameter, 100 cycles in each of two planes.<br><br>For flat cable: EIA 364-41 Condition II<br>250 cycles using either Method 1 or 2. | No physical damage. No discontinuity over 1 μs during flexing. |
| Insertion force | EIA 364-13<br>Measure the force necessary to mate the connector assemblies at a max. rate of 12.5 mm (0.492") per minute. | 45 N maximum |
| Removal force | EIA 364-13<br>Measure the force necessary to unmate the connector assemblies at maximum rate of 12.5 mm (0.492") per minute. | 10 N minimum |
| Durability | EIA 364-09<br>50 cycles for internal cabled application; 500 cycles for backplane/blindmate application. Test done at a maximum rate of 200 cycles per hour. | No physical damage. Meet requirements of additional tests as specified in the test sequence in section 6.3.9.5 |

**Table 7 – Environmental parameters, test procedures, and requirements**

| Parameter | Procedure | Requirement |
|---|---|---|
| Physical shock | EIA 364-27 Condition H<br>Subject mated connectors to 30 g's half-sine shock pulses of 11 msec duration. Three shocks in each direction applied along three mutually perpendicular planes for a total of 18 shocks. See NOTE 2. | No discontinuities of 1 μs or longer duration. No physical damage. |
| Random vibration | EIA 364-28 Condition V Test letter A<br>Subject mated connectors to 5.35 g's RMS. 30 minutes in each of three mutually perpendicular planes. See NOTE 2. | No discontinuities of 1 μs longer duration. |
| Humidity | EIA 364-31 Method II Test Condition A.<br>Subject mated connectors to 96 hours at 40° C with 90% to 95% RH. | See NOTE 1 |
| Temperature life | EIA 364-17 Test Condition III Method A.<br>Subject mated connectors to temperature life at +85°C for 500 hours. | See NOTE 1. |
| Thermal shock | EIA 364-32 Test Condition I.<br>Subject mated connectors to 10 cycles between –55° C and +85° C. | See NOTE 1. |
| Mixed Flowing Gas | EIA 364-65, Class 2A<br>Half of the samples are exposed unmated for seven days, then mated for remaining seven days. Other half of the samples are mated during entire testing. | See NOTE 1. |

NOTE –

1. Shall meet EIA 364-18 Visual Examination requirements, show no physical damage, and shall meet requirements of additional tests as specified in the test sequence in section 6.3.9.5.
2. Shock and vibration test fixture is to be determined by each user with connector vendors.

An additional requirement is listed in **Table 8.**

**Table 8 – Additional requirement**

| Parameter | Procedure | Requirement |
|---|---|---|
| Flammability | UL 94V-0 | Material certification or certificate of compliance required with each lot to satisfy the Underwriters Laboratories follow-up service requirements. |

It should be pointed out that this specification does not attempt to define the connector and cable assembly reliability requirements that are considered application-specific. It is up to users and their connector suppliers to determine if additional requirements shall be added to satisfy the application

needs. For example, a user who requires a SMT connector may want to include additional requirements for SMT connector reliability.

### 6.3.9.4 Sample selection

Samples shall be prepared in accordance with applicable manufacturers' instructions and shall be selected at random from current production. Each test group shall provide 100 data points for a good statistical representation of the test result. For a connector with greater than 20 pins, a test group shall consist of a minimum of five connector pairs. From these connector pairs, a minimum of 20 contact pairs per mated connector shall be selected and identified. For connectors with less than 20 pins, choose the number of connectors sufficient to provide 100 data points.

### 6.3.9.5 Test sequence

Table 9 shows the connector test sequences for five groups of tests.

**Table 9 – Connector test sequences**

| **Test group ➔** | **A** | **B** | **C** | **D** | **E** |
|---|---|---|---|---|---|
| **Test or examination ⬇** | | | | | |
| Examination of the connector(s) | 1, 5 | 1, 9 | 1, 8 | 1, 8 | 1, 7 |
| Low-Level Contact Resistance (LLCR) | 2, 4 | 3, 7 | 2, 4, 6 | | 4, 6 |
| Insulation resistance | | | | 2, 6 | |
| Dielectric withstanding voltage | | | | 3, 7 | |
| Current rating | | | 7 | | |
| Insertion force | | 2 | | | |
| Removal force | | 8 | | | |
| Durability | 3 | 4[(a)] | | | 2[(a)] |
| Physical shock | | 6 | | | |
| Vibration | | 5 | | | |
| Humidity | | | | 5 | |
| Temperature life | | | 3 | | |
| Reseating (manually unplug/plug three times) | | | 5 | | 5 |
| Mixed Flowing Gas | | | | | 3 |
| Thermal shock | | | | 4 | |

NOTE –
(a) Preconditioning, 20 cycles for the 50-durability cycle requirement, 50 cycles for the 500-durability cycle requirement. The insertion and removal cycle is at the maximum rate of 200 cycles per hour.

For example, in Test Group A, one would perform the following tests:

1. Examination of the connector(s)
2. LLCR
3. Durability
4. LLCR
5. Examination of the connector(s)

## 6.4 Low level electronics block diagram

### 6.4.1 Diagram

The following block diagrams are provided as a reference for the following sections of this document. Although informative in nature, the functions of the blocks described herein provide the basis upon which the normative specifications apply. The individual blocks provided are considered the "industry standard" way to approach this design and are provided as an example of one possible implementation.

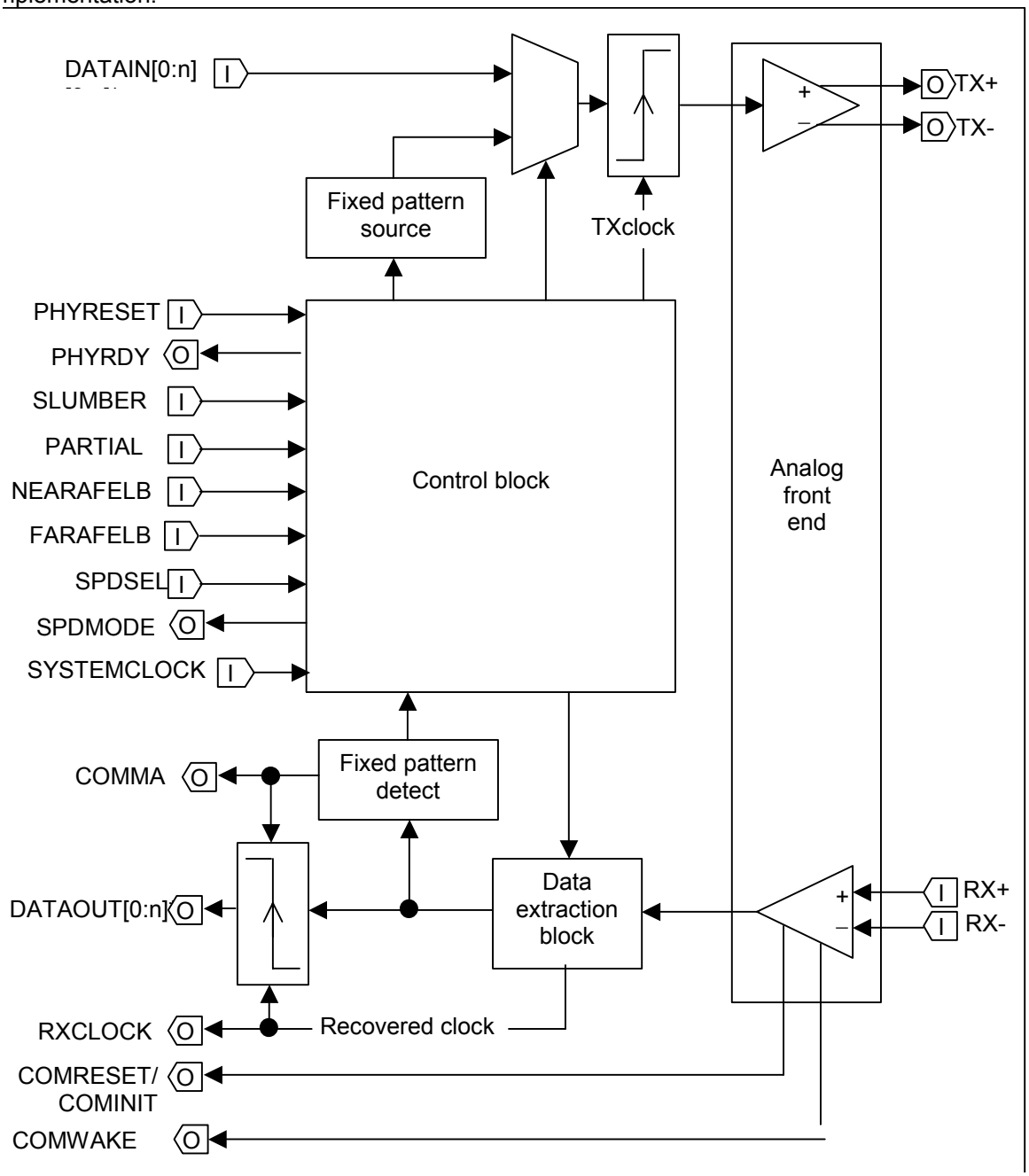



**Figure 18 – Physical plant overall block diagram**

### 6.4.2 Physical plant overall block diagram description

| | |
|---|---|
| Analog front end | This block is the basic interface to the transmission line. This block consists of the high speed differential drivers and receivers as well as the Out of Band signaling circuitry. |
| Control block | This block is a collection of logic circuitry that controls the overall functionality of the Physical plant circuitry. |
| Fixed pattern source | This block provides the support circuitry that generates the patterns as needed to implement the ALIGN primitive activity. |
| Fixed pattern detect | This block provides the support circuitry to allow proper processing of the ALIGN primitives. |
| Data extraction block | This block provides the support circuitry to separate the clock and data from the high speed input stream. |
| TX clock | This signal is internal to the Physical plant and is a reference signal that regulates the frequency at which the serial stream is sent via the high speed signal path |
| TX + / TX - | These signals are the outbound high speed differential signals that are connected to the serial ATA cable. |
| RX + / RX - | These signals are the inbound high speed differential signals that are connected to the serial ATA cable. |
| DATAIN | Data sent from the Link layer to the Phy layer for serialization and transmission. |
| PHYRESET | This input signal causes the PHY to initialize to a known state and start generating the COMRESET Out of Band signal across the interface. |
| PHYRDY | Signal indicating Phy has successfully established communications. The Phy is maintaining synchronization with the incoming signal to its receiver and is transmitting a valid signal on its transmitter. |
| SLUMBER | Causes the Phy layer to transition to the Slumber power management state. |
| PARTIAL | Causes the Phy layer to transition to the Partial power management state |
| NEARAFELB | Causes the Phy to loop back the serial data stream from its transmitter to its receiver |
| FARAFELB | Causes the Phy to loop back the serial data stream from its receiver to its transmitter |
| SPDSEL | Causes the control logic to automatically negotiate for a usable interface speed or sets a particular interface speed. The actual functionality of this input is vendor specific and varies from manufacturer to manufacturer. |
| SPDMODE | Output signal that reflects the current interface speed setting. The actual functionality of this signal is vendor specific and varies from manufacturer to manufacturer. |

SYSTEMCLOCK This input is the reference clock source for much of the control circuit and is the basis from which the transmitting interface speed is established.

COMMA This signal indicates that a K28.5 character was detected in the inbound high speed data stream.

DATAOUT Data received and deserialized by the Phy and passed to the Link layer

RX CLOCK / Recovered clock - This signal is derived from the high speed input data signal and determines when parallel data has been properly formed at the DATAOUT pins and is available for transfer to outside circuitry.

COMRESET / COMINIT
Host: Signal from the out of band detector that indicates the COMINIT out of band signal is being detected.
Device: Signal from the out of band detector that indicates the COMRESET out of band signal is being detected.

COMWAKE Signal from the out of band detector that indicates the COMWAKE out of band signal is being detected.

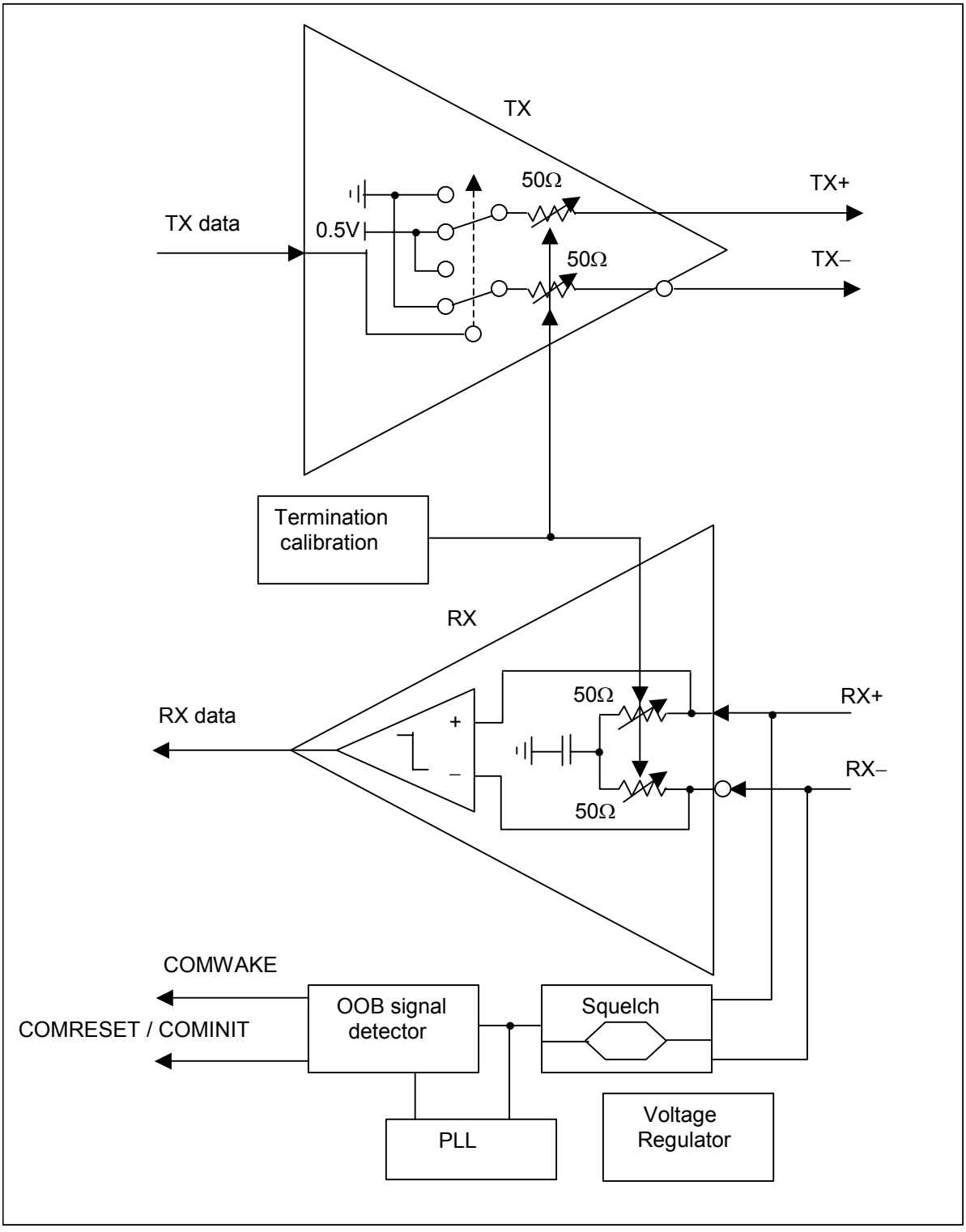



**Figure 19 – Analog front end (AFE) block diagram**

### 6.4.3 Analog front end (AFE) block diagram description

| | |
|---|---|
| TX | This block contains the basic high speed driver electronics |
| RX | This block contains the basic high speed receiver electronics |
| Termination calibration | This block is used to establish the impedance of the RX block in order to properly terminate the high speed serial cable. |
| Squelch | This block establishes a limit so that detection of a common mode signal can be properly accomplished. |
| OOB signal detector | This block decodes Out of Band signal from the high speed input signal path. |
| PLL | This block is used to synchronize an internal clocking reference so that the input high speed data stream may be properly decoded. |
| Voltage Regulator | This block stabilizes the internal voltages used in the other blocks so that reliable operation may be achieved. This block may or may not be required for proper operation of the balance of the circuitry. The need for this block is implementation specific. |
| TX + / TX - | This is the same signal as described in the previous section: Physical plant overall block diagram description |
| RX + / RX - | This is the same signal as described in the previous section: Physical plant overall block diagram description |
| TxData | Serially encoded 10b data attached to the high speed serial differential line driver. |
| RxData | Serially encoded 10b data attached to the high speed serial differential line receiver. |
| COMWAKE | This is the same signal as described in the previous section: Physical plant overall block diagram description |
| COMRESET / COMINIT | This is the same signal as described in the previous section: Physical plant overall block diagram description |

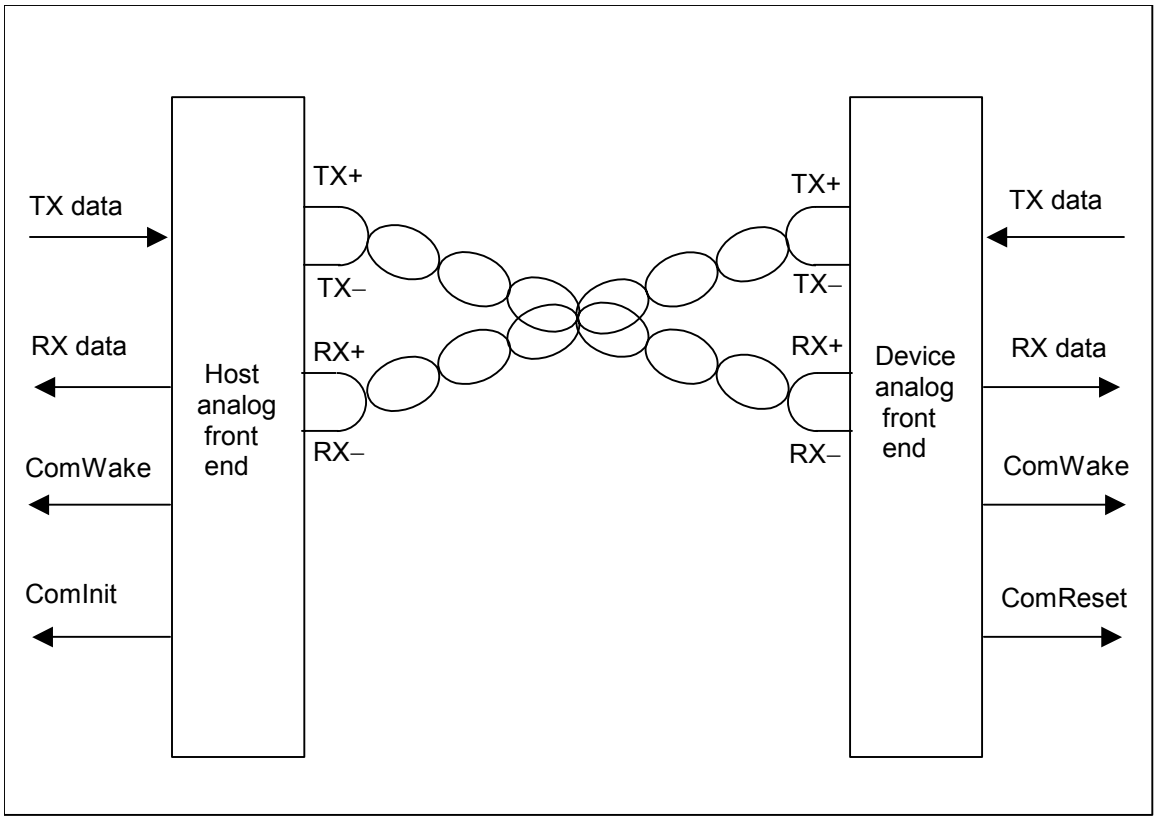



**Figure 20 – Analog front end (AFE) cabling**

## 6.5 General specifications

### 6.5.1 System

**Table 10. – General system specifications**

| Name | Nom | Min | Max | Units | Description |
|---|---|---|---|---|---|
| $\Phi_{G1}$ | 1.2 | | | Gbits/s | 1st Generation 8b data rate |
| $\phi_{G1}$ | 1.5 | | | Gbits/s | 1st Generation 10b bit rate |
| $\Phi_{G2}$ | 2.4 | | | Gbits/s | 2nd Generation 8b data rate |
| $\phi_{G2}$ | 3.0 | | | Gbits/s | 2nd Generation 10b bit rate |
| $\Phi_{G3}$ | 4.8 | | | Gbits/s | 3rd Generation 8b data rate |
| $\phi_{G3}$ | 6.0 | | | Gbits/s | 3rd Generation 10b bit rate |

## 6.6 Module specifications

All specifications in this section apply to a Generation 1 Serial ATA device. Generation 2 and beyond will be addressed in a separate document.

### 6.6.1 Definitions

RJ Random Jitter (peak to peak). Assumed to be Gaussian and equal to 14 times the 1$\sigma$ rms value given the $10^{-12}$ BER requirement.

DJ Deterministic Jitter (peak to peak). All jitter sources that do not have tails on their probability distribution function (i.e. values outside the bounds have probability zero). Four kinds of deterministic jitter are identified: duty cycle distortion, data dependent (ISI), sinusoidal, and uncorrelated (to the data) bounded. DJ is characterized by its bounded, peak-to-peak value.

TJ Total Jitter (peak to peak). Peak to peak measured jitter including DJ and RJ.

ISI Inter-symbol interference. Data-dependent deterministic jitter caused by the time differences required for the signal to arrive at the receiver threshold when starting from different places in bit sequences (symbols). For example media attenuates the peak amplitude of the bit sequence [ 0,1,0,1... ], more than it attenuates the peak amplitude of the bit sequence [ 0,0,0,0,1,1,1,1... ], thus the time required to reach the receiver threshold with the [ 0,1,0,1... ] sequence is less than required from the [ 0,0,0,0,1,1,1,1... ]sequence. The run length of 4 produces a higher amplitude which takes more time to overcome when changing bit values and therefore produces a time difference compared to the run length of 1 bit sequence. When different run lengths are mixed in the same transmission the different bit sequences (symbols) therefore interfere with each other. ISI is expected whenever any bit sequence has frequency components that are propagated at different rates by the transmission media. This translates into high-high-frequency, data-dependent, jitter.

UI Unit Interval. Equal to the time required to transmit one bit (666.667 ps for Gen1).

DC Strictly, the non-AC component of a signal. In this specification, DC means all frequency components below $f_{dc}$ = 100 kHz.

Differential Signal A signal derived by taking the difference between two conductors. In this spec a differential signal is comprised of a positive conductor and a negative conductor. The differential signal is the voltage on the positive conductor minus the voltage on the negative conductor (i.e. TX+ – TX-).

Burst A short pulse of data starting from and ending with the idle condition on the interface. These are used for low-level signaling when the high-speed communication channel is not established.

SSC Spead Spectrum Clocking. The technique of modulating the operating frequency of a circuit slightly to spread its radiated emissions over a range of frequencies rather than just one tone. This reduction in the maximum emission for a given frequency helps meet FCC requirements.

### 6.6.2 Electrical specifications

The Serial ATA physical layer shall comply to the electrical specifications depicted in Table 11. – General electrical specifications.

The physical layer module consists of the driver, receiver, PCB and mated connector pair. The physical layer module shall meet Table 11 electrical signal parameters and requirements. Unless

otherwise specified, all measurements shall be taken through the mated connector pair. The Receiver test fixture shown in Figure 46 or the Transmitter test fixture shown in Figure 45 is used. Driver and receiver designs must compensate for the effects of the path to/from the I/O connector.

**Table 11. – General electrical specifications**

| | Nom | Min | Max | units | Comments |
|---|---|---|---|---|---|
| T,UI | | 666.43 | 670.12 | ps | Operating data period (nominal value architecture specific) |
| $t_{rise}$ | 0.3 | 0.2 | 0.41 | UI | 20%-80% at transmitter |
| $t_{fall}$ | 0.3 | 0.2 | 0.41 | UI | 80%-20% at transmitter |
| $V_{cm,dc}$ | 250 | 200 | 450 | mV | Common mode DC level measured at receiver connector. This spec only applies to direct-connect designs or designs that hold the common-mode level. AC coupled designs may allow the common mode to float. See $V_{cm,ac\ coupled}$ requirements. |
| $V_{cm,ac\ coupled\ TX}$ | | 0 | 2.0 | V | Open circuit DC voltage level of each signal in the TX pair at the IC side of the coupling capacitor in an AC coupled PHY. Must be met during all possible power and electrical conditions of the PHY including power off and power ramping.<br><br>Transmitter common mode DC levels outside this range may be used provided that the following is met: The common mode voltage transients measured at the TX pins of the connector into an open-circuit load during all power states and transitions shall never exceed a +2.0V or –2.0V change from the CM value at the beginning of each transient. Test conditions shall include system power supply ramping at the fastest possible power ramp (up and down) for the system using such a PHY. |
| $V_{cm,ac\ coupled\ RX}$ | | 0 | 2.0 | V | Open circuit DC voltage level of each signal in the RX pair at the IC side of the coupling capacitor in an AC coupled PHY. Must be met during all possible power and electrical conditions of the PHY including power off and power ramping.<br><br>Receiver common mode DC levels outside this range may be used provided that the following is met: The common mode voltage transients measured at the RX |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | pins of the connector into an open-circuit load during all power states and transitions shall never exceed a +2.0V or –2.0V change from the CM value at the beginning of each transient. Test conditions shall include system power supply ramping at the fastest possible power ramp (up and down) for the system using such a PHY. |
| $V_{cm,ac}$ | | | 100 | mV | Max sinusoidal amplitude of common mode signal measured at receiver connector |
| $F_{CM}$ | | 2 | 200 | MHz | All receivers must be able to tolerate sinusoidal common-mode noise components inside this frequency range with an amplitude of $V_{cm,ac}$. |
| $T_{settle,CM}$ | | | 10 | ns | Maximum time for common-mode transients to settle to within 10% of DC value during transitions to and from the idle bus condition. |
| $V_{diff,tx}$ | 500 | 400 | 600 | $mV_{p-p}$ | +/- 250 mV differential nominal. Measured at Serial ATA connector on transmit side |
| $V_{diff,rx}$ | 400 | 325 | 600 | $mV_{p-p}$ | +/- 200 mV differential nominal. Measured at Serial ATA connector on receive side |
| Tx pair differential impedance | 100 | 85 | 115 | Ohm | As seen by a differential TDR with 100 ps (max) edge looking into connector (20%-80%). Measured with TDR in differential mode. |
| Rx pair differential impedance | 100 | 85 | 115 | Ohm | As seen by a differential TDR with 100 ps (max) edge looking into connector (20%-80%). Measured with TDR in differential mode. |
| Tx single-ended impedance | | 40 | | Ohm | As seen by TDR with 100 ps (max) edge looking into connector (20%-80%).<br>TDR set to produce simultaneous positive pulses on both signals of the Tx pair. Single-ended impedance is the resulting (even mode) impedance of each signal. Both signals must meet the single ended impedance requirement.<br>Must be met during all possible power and electrical conditions of the PHY including power off and power ramping. |
| Rx single-ended impedance | | 40 | | Ohm | As seen by TDR with 100 ps (max) edge looking into connector (20%-80%).<br>TDR set to produce simultaneous |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | positive pulses on both signals of the Rx pair. Single-ended impedance is the resulting (even mode) impedance of each signal. Both signals must meet the single ended impedance requirement. Must be met during all possible power and electrical conditions of the PHY including power off and power ramping. |
| $C_{ACcoupling}$ | | | 12 | nF | Coupling capacitance value for AC coupled TX and RX pairs. |
| TX DC clock frequency skew | | -350 | +350 | ppm | Specifies the allowed ppm tolerance for TX DC frequency variations around the nominal 1.500GHz. Excludes the +0/-5000ppm SSC downspread AC modulation per 6.6.4.5. |
| TX AC clock frequency skew | | -5000 | +0 | ppm | Specifies the allowed ppm extremes for the SSC AC modulation, subject to the "Downspread SSC" triangular modulation (30-33kHz) profile per 6.6.4.5. Note: Total TX Frequency variation around nominal 1.500G, includes [TXDC] + [TX AC] ppm variations. |
| TX differential skew | | | 20 | ps | (Nominal value architecture specific) |
| Squelch detector threshold | 100 | 50 | 200 | $mV_{p-p}$ | Minimum differential signal amplitude |
| COMRESET/COMINIT detector off threshold | | 175 | 525 | ns | Detector shall reject all bursts with spacings outside this spec. |
| COMRESET/COMINIT detector on threshold | 320 | 304 | 336 | ns | Detector shall detect all bursts with spacings meeting this period |
| COMRESET/COMINIT transmit spacing | 320.0 | 310.4 | 329.6 | ns | As measured from 100mV differential crosspoints of last and first edges of bursts |
| COMWAKE detector off threshold | | 55 | 175 | ns | Detector shall reject all bursts with spacings outside this spec. |
| COMWAKE detector on threshold | 106.7 | 101.3 | 112 | ns | Detector shall detect all burst spacings meeting this period |
| COMWAKE transmit spacing | 106.7 | 103.5 | 109.9 | ns | As measured from 100mV differential crosspoints from last to first edges of bursts |
| $UI_{OOB}$ | | 646.67 | 686.67 | ps | Operating data period during OOB burst transmission |

### 6.6.3 Differential voltage/timing (EYE) diagram

The EYE diagram is more of a qualitative measurement than a spec. Any low frequency (trackable) modulation must be tracked by the oscilloscope to prevent measurement error caused by benign EYE closure. It is also unrealistic to try to capture sufficient edges to guarantee the 14 sigma RJ requirement in order to achieve an effective $10^{-12}$ BER. Nonetheless, this method is useful and easy to set up in the lab.

If t3-t1 is set equal to $DJ+6*RJ_{1sigma}$, then less than one in every 750 edges will cross the illegal region for a well designed transmitter. By controlling the number of sweeps displayed, a quick health check may be performed with minimal setup. Note that this criteria is informative, and these conditions are not sufficient to guarantee compliance, but are to be used as part of the design guidelines.

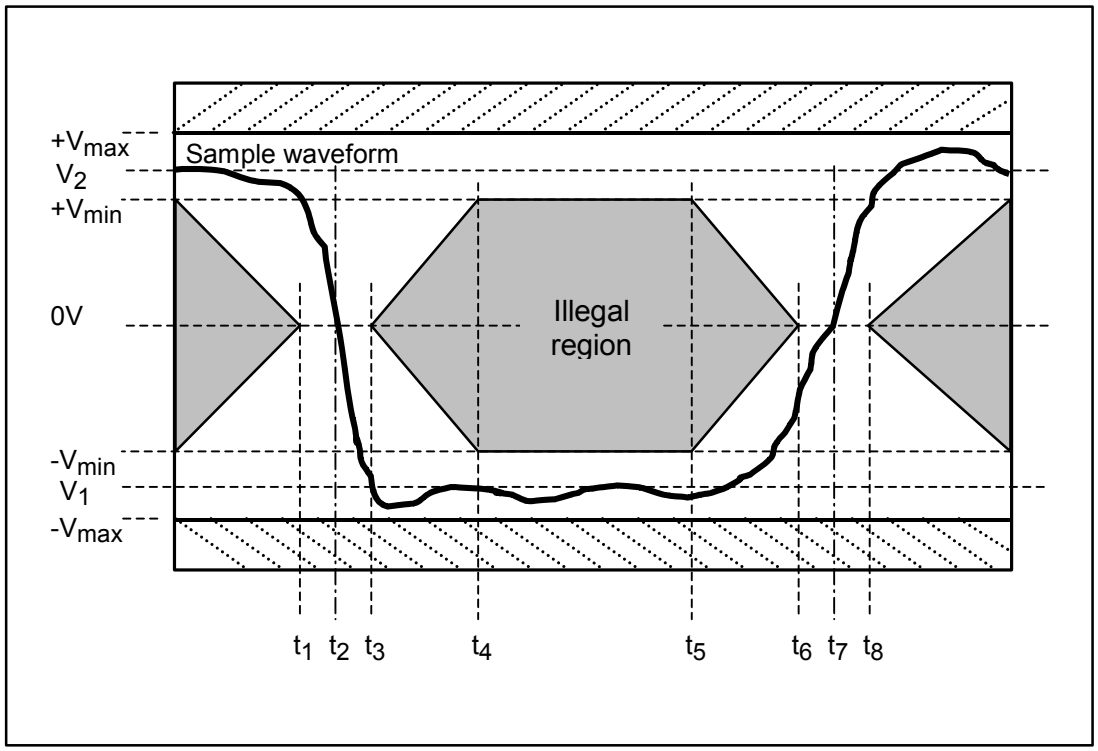



**Figure 21 – Voltage / timing margin base diagram**

**Table 12. – Voltage / Timing Margin Definition**

| Name | Definition | Notes |
|---|---|---|
| $\mathbf{t}_{jitter}$ | t3 - t1 | t3 - t1 = t8 - t6 |
| **T** | t7 - t2 | t2 - t1 = t3 - t2<br>t7 - t6 = t8 - t7 |
| $\mathbf{V}_{diff}$ | $V_2 - V_1$ | |

### 6.6.4 Sampling jitter specifications

#### 6.6.4.1 Jitter output/tolerance mask

The Serial ATA spectral jitter shall comply to the requirements as indicated in the Jitter Output/Tolerance Graph, shown in Figure 22 – Jitter output/tolerance mask with magnitudes defined in Table 13. – Sampling differential noise budget. $A_x$ are the maximum peak to peak transmitter output and the minimum peak to peak receiver tolerance requirements as measured from a data edge to any following data edge up to $n_x$*UI later (where x is 0,1, or 2 in accordance with Figure 23 - Jitter as a function of frequency).

Transmitter output data edge to data edge timing variation from $t_o$ to $t_y$ shall not exceed the value computed by the following:

- y is an integer from 1 to $n_x$,
- $t_y$ is the time between the data transition at $t_0$ and a data transition y*UI bit periods later.

This measurement can be made with an oscilloscope having a histogram function or with a Timing Interval Analyzer (TIA). For further information on jitter measurements see section 6.7. A receiver must be able to tolerate the peak to peak jitter specified.

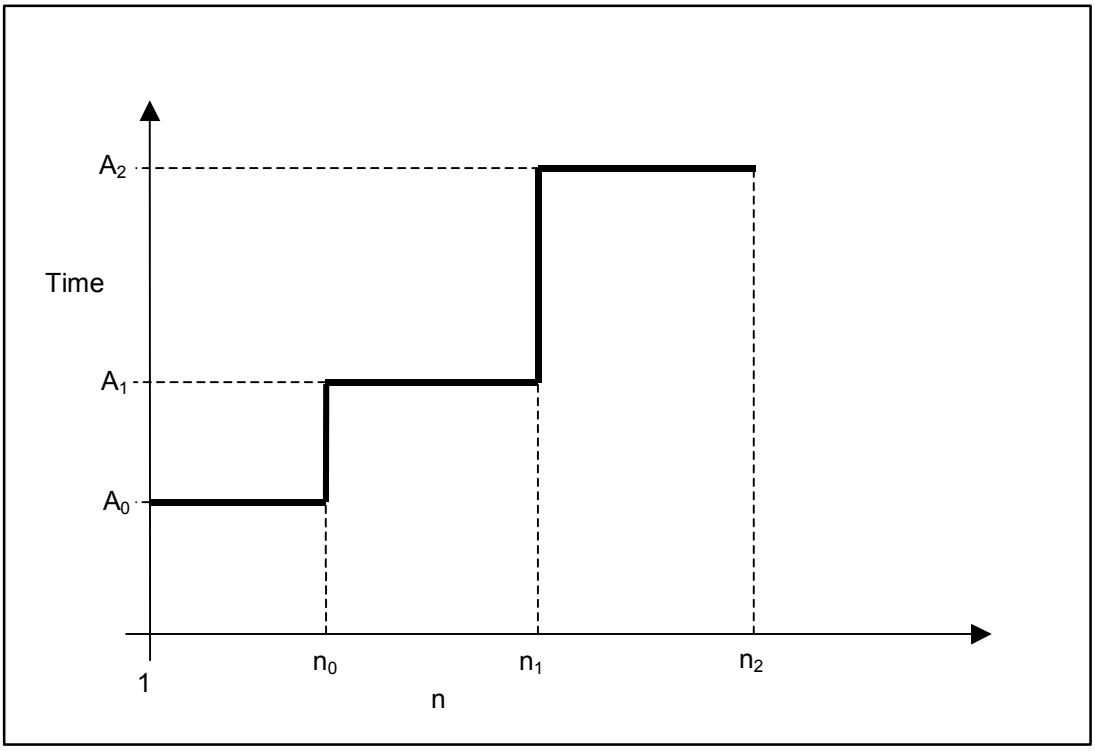



**Figure 22 – Jitter output/tolerance mask**

### 6.6.4.2 Sampling differential noise budget

Sampling jitter specifications relate to the relationship between the sampling clock and the data. Any phase error that results in the sample being improperly read (i.e. prior bit or following bit sampled) will result in a bit error.

These error components have been broken out into Deterministic Jitter (DJ) and Total Jitter (TJ) where appropriate. DJ is the peak to peak phase variation in the 0 $V_{differential}$ crossing point of the data stream that is fixed given any specific set of conditions. TJ is defined as DJ + Random Jitter (RJ). RJ is defined as 14 times the rms (1 sigma) value of the jitter that is Gaussian (normal).

The Serial ATA interface jitter characteristics should comply to within the jitter budget allocations tabulated in Table 13

**Table 13. – Sampling differential noise budget**

| Descr | Driver output[1] | | Driver PCB connector | | Receiver PCB connector | | Receiver input[2] | | Note ref |
|---|---|---|---|---|---|---|---|---|---|
| | DJ | TJ | DJ | TJ | DJ | TJ | DJ | TJ | |
| $\mathbf{A_{0,p-p}}$ (UI) | 0.15 | 0.33 | 0.175 | 0.355 | 0.25 | 0.43 | 0.275 | 0.455 | 4,5 |
| $\mathbf{n_0}$ | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4,5 |
| $\mathbf{A_{1,p-p}}$ (UI) | 0.2 | 0.45 | 0.22 | 0.47 | 0.35 | 0.6 | 0.37 | 0.62 | 4,6 |
| $\mathbf{n_1}$ | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 4,6 |
| $\mathbf{A_{2,p-p}}$ (UI) | 40 | | 40 | | 40 | | 40 | | 3,4,7 |
| $\mathbf{n_2}$ | 25000 | | 25000 | | 25000 | | 25000 | | 3,4,7 |

NOTES –
1. The driver output is the maximum jitter that a driver may exhibit to guarantee operation.
2. This field is the maximum jitter that a receiver must tolerate to guarantee operation.
3. For low frequency (trackable) jitter, total jitter is specified (DJ not broken out)
4. Does not include UI error due to frequency skew (XTAL or SSC related)
5. Primarily determined by non-tracking architecture requirements
6. Primarily determined by tracking architecture requirements
7. Primarily determined by Spread Spectrum Clocking (+/-0.25% AC portion). Doesn't include the –0.25% fixed skew (additional 26 UI).

### 6.6.4.3 Relationship of frequency to the jitter specification

It is often useful to look at the jitter specification as a function of frequency. This section is provided as clarifying information (informative, not normative). Figure 23 shows a plot of the maximum amplitude (in UI) sine wave at a given frequency that satisfies all the jitter specifications and the calculations leading to this graph. Two graphs are included – the second is a close-up of the high-frequency requirements.

Jitter specification at the receiver input

$A0 := 0.455 \qquad A1 := 0.62 \qquad A2 := 40$

$N0 := 5 \qquad N1 := 250 \qquad N2 := 25000$

$datarate := 1.5 \cdot 10^9$

$f(n) := \frac{datarate}{2 \cdot n}$ Frequency of sine wave with peak values separated by n datarate cycles

$a(n, n_x, a_x) := \frac{a_x}{\sin\left(\frac{\pi \cdot n_x}{2 \cdot n}\right)}$ Maximum amplitude of of sine wave with frequency f(n) and a slewrate constrained by jitter spec $n_x, a_x$

Maximum compliant peak-peak amplitude

$$p(n) := \begin{cases} A0 & \text{if } (n \le N0) \\ \min((a(n, N0, A0)\ A1)) & \text{if } (n > N0) \cdot (n \le N1) \\ \min((a(n, N1, A1)\ A2)) & \text{if } (n > N1) \\ 0 & \text{otherwise} \end{cases}$$

Maximum compliant peak-peak amplitude plotted against log(frequency)

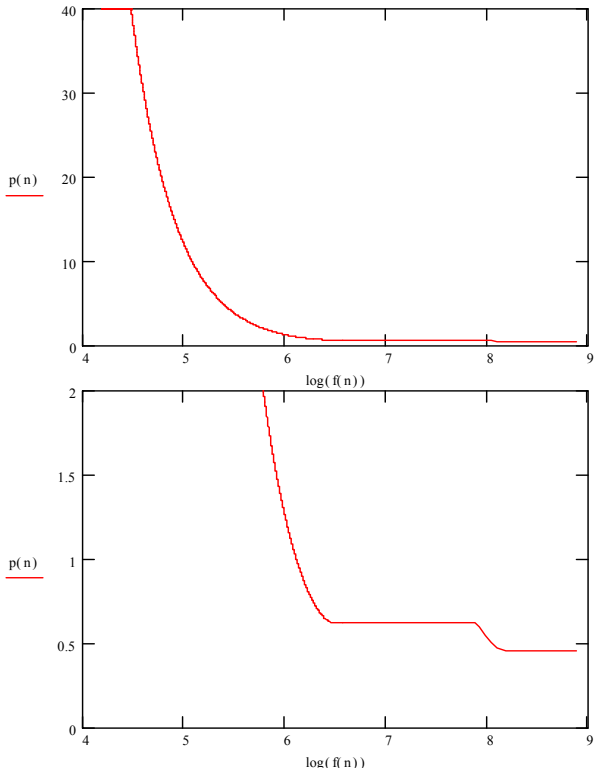



**Figure 23 - Jitter as a function of frequency**

### 6.6.4.4 Sampling BER and jitter formulas

The values for RJ and DJ above are calculated using the following relationship between BER and jitter. This is provided for reference (informative) and not intended for use as a compliance requirement (normative). The over-sampling formula assumes a worse-case oversampling ratio (osr) of three and sets the $A_0,n_0$ jitter requirements. The tracking architecture sets the $A_1,n_1$ requirement. $A_2,n_2$ is set to guarantee SSC compliance. Also shown below are $DJ_{RX}$ and $RJ_{RX}$ numbers for the two architectures. These are example budgets for the local receiver that satisfy the $10^{-12}$ BER goal.

**Gaussian Distribution Error Function**

$$G(x) := \frac{1}{\sqrt{2\cdot\pi}} \cdot \int_0^x e^{\frac{-\xi^2}{2}}\, d\xi + 0.5 \quad x > 0$$

**BER Due to Sampling (oversampling architecture)**

$osr := 3$ $\quad DJ_{RX} := 0.1$ $\quad RJ_{RX} := 0.1$

$DJ := 0.275 + DJ_{RX}$ $\quad RJ := 0.18 + RJ_{RX}$

$$BER_{os} := 2 - G\left[\frac{\frac{1}{osr} - \frac{DJ}{2}}{\frac{RJ}{14}}\right] - G\left[\frac{\frac{osr-1}{osr} + \frac{DJ}{2}}{\frac{RJ}{14}}\right]$$

$$BER_{os} = 1.53\cdot 10^{-13}$$

**BER Due to Sampling (tracking architecture)**

$DJ_{RX} := 0.15$ $\quad RJ_{RX} := 0.18$

$DJ := 0.37 + DJ_{RX}$ $\quad RJ := 0.25 + RJ_{RX}$

$$BER_{tr} := 2 - G\left[\frac{\frac{1}{2} - \frac{DJ}{2}}{\frac{RJ}{14}}\right] - G\left[\frac{\frac{1}{2} + \frac{DJ}{2}}{\frac{RJ}{14}}\right]$$

$$BER_{tr} = 2.665\cdot 10^{-15}$$

**Figure 24 - Sampling bit error rate formulas**

### 6.6.4.5 Spread spectrum clocking (SSC) clarification:

Spread Spectrum functionality define follows

1. All device timings (including jitter, skew, min-max clock period, output rise/fall time) MUST meet the existing non-spread spectrum specifications
2. The minimum clock period cannot be violated. The preferred method is to adjust the spread technique to not allow for modulation above the nominal frequency. This technique is often called “down-spreading”. An example triangular frequency modulation profile is shown in Figure 25. The modulation profile in a modulation period can be expressed as:

$$f = \begin{cases} (1-\delta) f_{nom} + 2 f_m \cdot \delta \cdot f_{nom} \cdot t & when \quad 0 < t < \dfrac{1}{2 f_m}; \\ (1+\delta) f_{nom} - 2 f_m \cdot \delta \cdot f_{nom} \cdot t & when \quad \dfrac{1}{2 f_m} < t < \dfrac{1}{f_m}, \end{cases}$$

where $f_{nom}$ is the nominal frequency in the non-SSC mode, $f_m$ is the modulation frequency, $\delta$ is the modulation amount, and t is time.

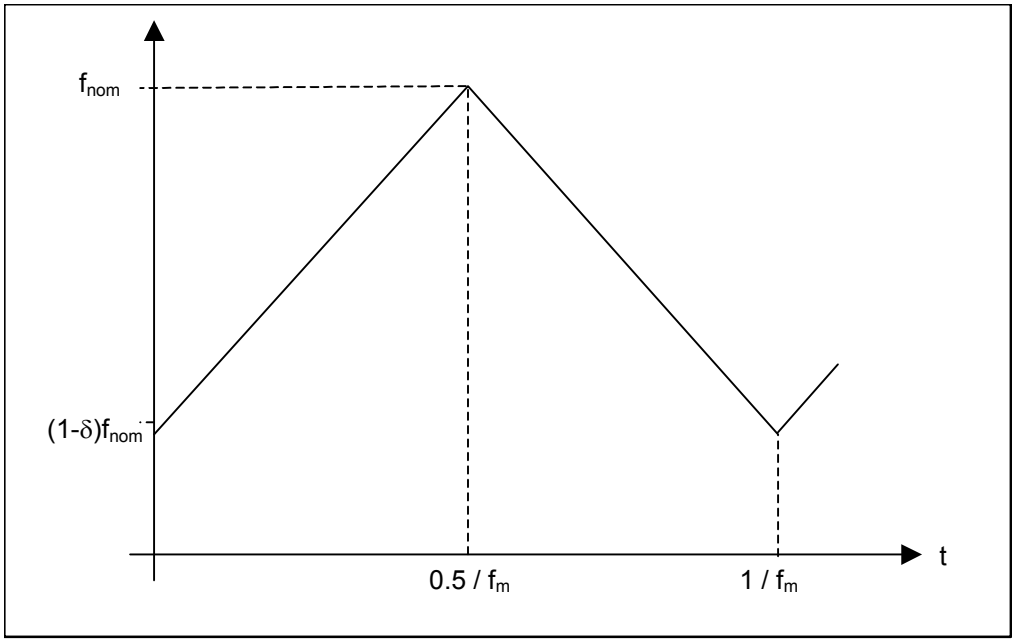



**Figure 25 - Triangular frequency modulation profile**

3. For triangular modulation, the clock frequency deviation ($\delta$) is required to be no more than 0.5% “down-spread” from the corresponding nominal frequency, i.e., +0%/-0.5%. The absolute spread amount at the fundamental frequency is shown in Figure 26, as the width of its spectral distribution (between the –3 dB roll-off). The ratio of this width to the fundamental frequency cannot exceed 0.5%. This parameter can be measured in the frequency domain using a spectrum analyzer.

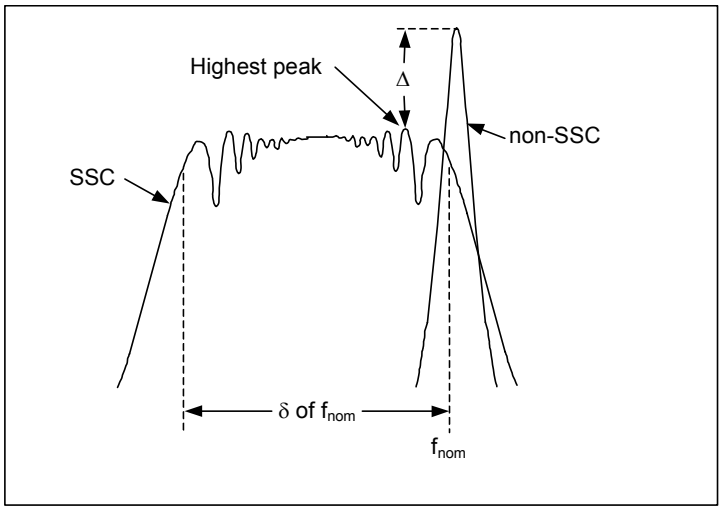



**Figure 26 - Spectral fundamental frequency comparison**

4. To achieved sufficient system-level EMI reduction, it is desired that SSC reduce the spectral peaks in the non-SSC mode by the amount specified in Table 14. The peak reduction Δ is defined, as shown in Figure 26, as the difference between the spectral peaks in SSC and non-SSC modes at the specified measurement frequency

**Table 14 – Desired peak amplitude reduction by SSC.**

| Clock freq. | Peak reduction Δ | Measurement freq. |
|---|---|---|
| 66 MHz | 7 dB | 466 MHz ($7^{th}$ harmonics) |
| 75 MHz | 7 dB | 525 MHz ($7^{th}$ harmonics) |

NOTES –
The spectral peak reduction is not necessarily the same as the system EMI reduction. However, this relative measurement gives the component-level indication of SSC's EMI reduction capability at the system level.

It is recommended that a spectrum analyzer be used for this measurement. The spectrum analyzer should have measurement capability out to 1 GHz. The measured SSC clock needs to be fed into the spectrum analyzer via a high-impedance probe compatible with the spectrum analyzer. The output clock should be loaded with 20 pF capacitance. The resolution bandwidth of the spectrum analyzer needs to be set at 120 KHz to comply with FCC EMI measurement requirements. The video band needs to be set at higher than 300 KHz for appropriate display. 100 KHz may be used as the resolution bandwidth in case of measurement equipment limitation. The display should be set with maximum hold. The corresponding harmonic peak readings should be recorded in both the non-SSC and the SSC modes, and be compared to determine the magnitude of the spectral peak reduction.

5. The modulation frequency of SSC is required to be in the range of 30-33 KHz to avoid audio band demodulation and to minimize system timing skew.

## 6.7 Functional specifications

### 6.7.1 Overview

The purpose of this section is to provide the normative functional definitions for various operations that a hardware platform is expected to accommodate.

### 6.7.2 Common-mode biasing

Since the Serial ATA protocol supports both direct coupled and AC-coupled solutions, there are four scenarios to be considered when applying the DC bias to TX and RX designs. In Figure 27 below, it is shown that only DC-coupled designs need sustain the specified 250 mV common-mode level to ensure interoperability.

A DC-coupled receiver (no blocking capacitors) shall weakly hold the common-mode level of its inputs at 250 mV. A DC-coupled transmitter shall transmit with the nominally specified 250 mV common-mode level.

An AC-coupled receiver shall meet the common mode AC coupled electrical requirements of Table 11and need not sustain the cable side of the its blocking capacitors. Similarly, an AC-coupled transmitter shall meet the common mode AC coupled electrical requirements of Table 11and need not sustain the common-mode level on the cable side of its blocking capacitors.

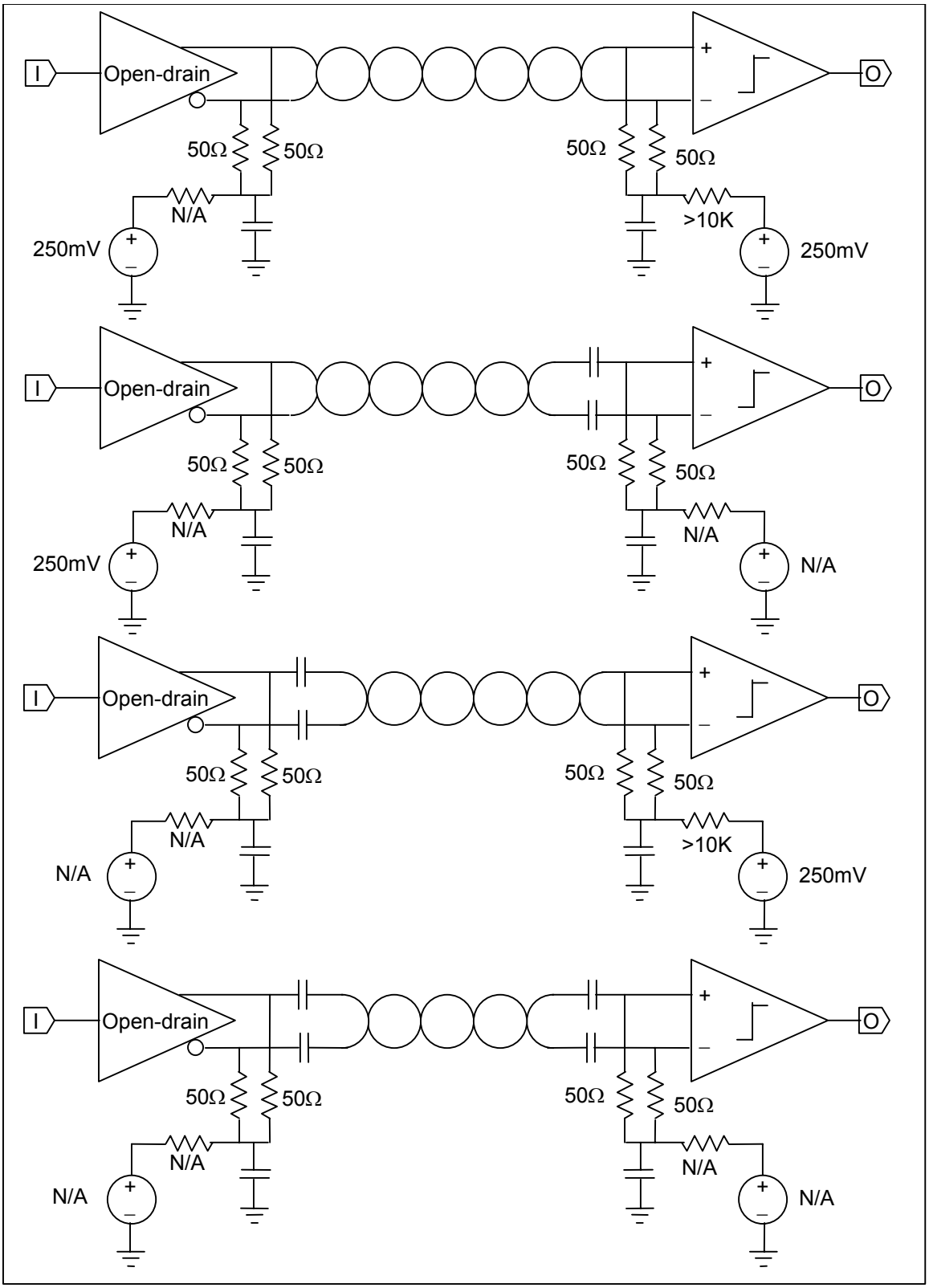



**Figure 27 – Common-mode biasing**

### 6.7.3 Matching

The host controller shall employ on-chip adaptive impedance matching circuits to ensure best possible termination for both its TX and RX. Device peripherals also shall provide impedance terminations, as per the specified parameters in Table 11 of section 6.6.2, and may adapt their termination impedance as well.

The host controller, since it is given the first opportunity to calibrate during the power on sequence, cannot assume that the far end of the cable is calibrated yet. For this reason, the host controller must utilize a separate reference to perform calibration. In a desktop system, the cable provides the optimal impedance reference for calibration.

Using Time Domain Reflectometry (TDR) techniques, the host may launch a step waveform from its transmitter, so as to get a measure of the impedance of the transmitter, with respect to the cable, and adjust its impedance settings as necessary.

In a mobile system environment, where the cable is small or non-existent, the host controller must make use of a separate reference (such as an accurate off-chip resistor) for the calibration phase.

The device, on the other hand, can assume that the termination on the far side (host side) of the cable is fully calibrated, and may make use of this as the reference. Using the host termination as the calibration reference allows the devices operating in both the desktop and the mobile system environment to use the same hardware.

Signals generated for the impedance calibration process shall not duplicate the OOB signals, COMWAKE, COMINIT, or COMRESET. Signals generated for the impedance calibration process shall not exceed the normal operating voltage levels, cited in section 6.6.2 See the power management section for suggested times to perform calibration during power-on.

### 6.7.4 Out of band signaling

There shall be three Out Of Band (OOB) signals used/detected by the Phy, COMRESET, COMINIT, and COMWAKE. COMINIT, COMRESET and COMWAKE OOB signaling shall be achieved by transmission of 160 Gen1 UI bursts of ALIGN primitives, each separated by idle periods (at common-mode levels), having durations of 160 Gen1 UI, as depicted in Figure 28, below. The content of the burst is reserved and set to be ALIGN primitives.

During OOB signaling transmissions, the differential and common mode levels of the signal lines shall comply with the same electrical specifications as for in-band data transmission, specified in section 6.6.2. In Figure 28 below, COMRESET, COMINIT, and COMWAKE are shown. OOB signals are observed by detecting the temporal spacing between adjacent bursts of activity, on the differential pair.

Any spacing less then 55 ns or greater then 175 ns shall invalidate the COMWAKE detector output. The COMWAKE OOB signaling is used to bring the Phy out of a power-down state (PARTIAL or SLUMBER) as described in section 6.8. The interface must be held inactive for at least 175ns after the last burst to ensure far-end detector detects the deassertion properly. The device may hold the interface inactive no more then 228.3ns (175ns + 2 Gen1 dwords) at the end of a COMWAKE to prevent susceptibility to crosstalk.

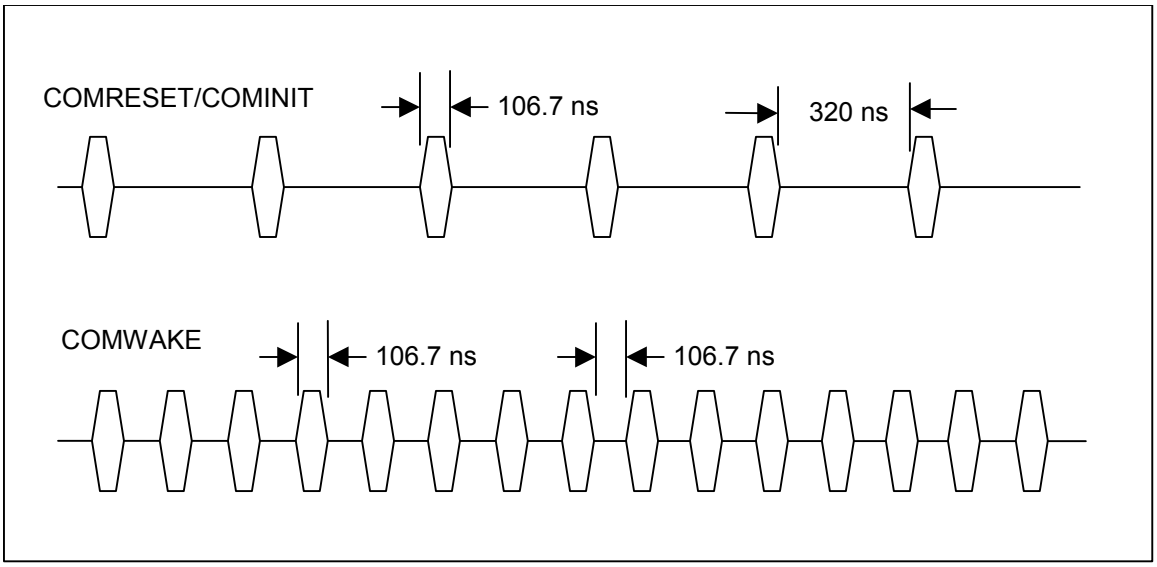



**Figure 28 – Out of band signals**

#### 6.7.4.1 Idle bus status

During the idle bus condition, the differential signal diminishes to zero while the common mode level of 250 mV remains.

Common-mode transients, shall not exceed the maximum amplitude levels (Vcm,ac) cited in section 6.6.2, and shall settle to within 25 mV of Vcm,DC within Tsettle,CM, cited in in section 6.6.2. The following figure shows several transmitter examples, and how the transition to and from the idle state may be implemented.

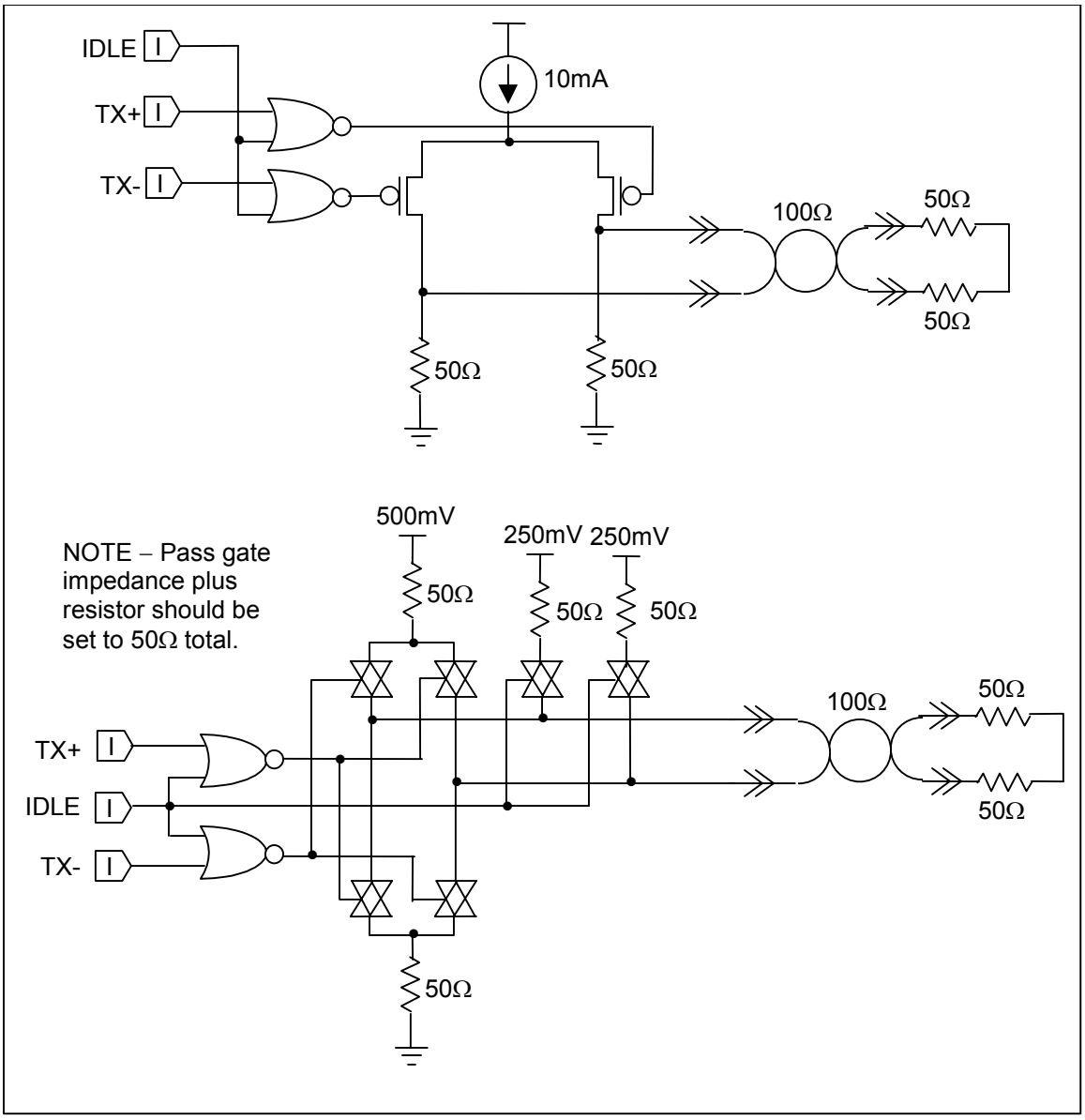



**Figure 29 – Transmitter examples**

*(continued)*

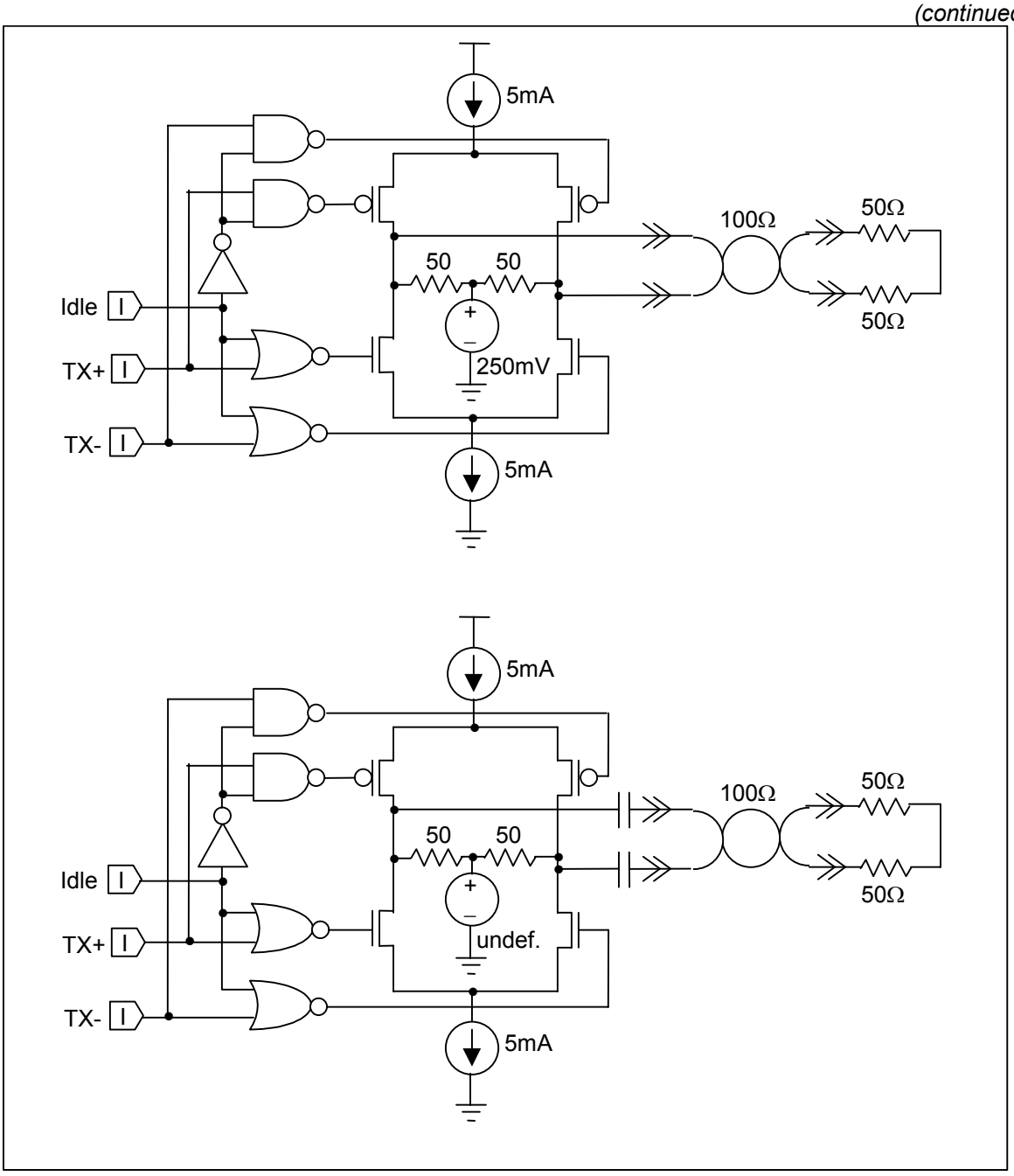



**Figure 29 – Transmitter examples** *(concluded)*

### 6.7.4.2 COMRESET

COMRESET always originates from the host controller, and forces a hard reset in the device. It is indicated by transmitting bursts of data separated by an idle bus condition.

The OOB COMRESET signal shall consist of no less than six data bursts, including inter-burst temporal spacing. The COMRESET signal shall be:

1) sustained/continued uninterrupted as long as the system hard reset is asserted, or
2) started during the system hard reset and ended some time after the deassertion of system hard reset, or
3) transmitted immediately following the deassertion of the system hard reset signal.

The host controller shall ignore any signal received from the device from the assertion of the hard reset signal until the COMRESET signal is transmitted.

Each burst shall be be 160 Gen1 UI's long (106.7 ns) and each inter-burst idle state shall be 480 Gen1 UI's long (320 ns). A COMRESET detector will look for four consecutive bursts with 320 ns spacing (nominal).

Any spacing less than 175 ns or greater than 525 ns shall invalidate the COMRESET detector output. The COMRESET interface signal to the Phy layer will initiate the Reset sequence shown in Figure 30 – COMRESET sequence below. The interface must be held inactive for at least 525ns after the last burst to ensure far-end detector detects the deassertion properly.

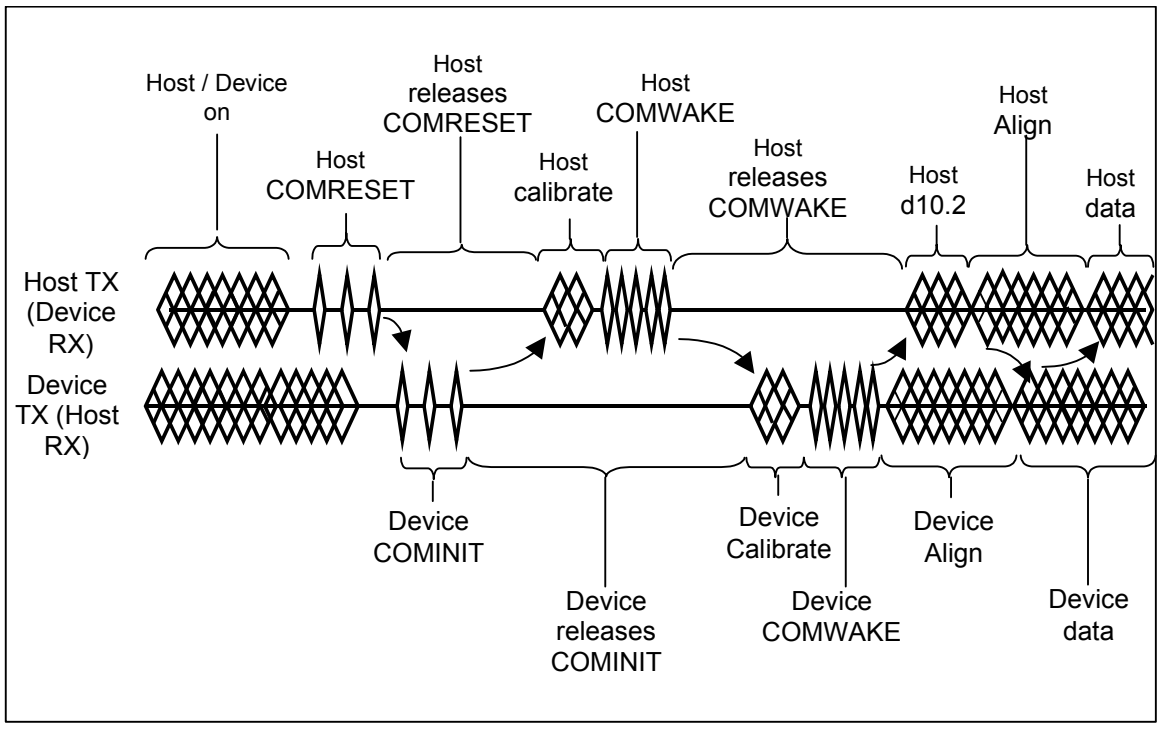



**Figure 30 – COMRESET sequence**

Description:

1. Host/device are powered and operating normally with some form of active communication.
2. Some condition in the host causes the host to issue COMRESET
3. Host releases COMRESET. Once the condition causing the COMRESET is released, the host releases the COMRESET signal and puts the bus in a quiescent condition.
4. Device issues COMINIT – When the device detects the release of COMRESET, it responds with a COMINIT. This is also the entry point if the device is late starting. The device may initiate communications at any time by issuing a COMINIT.
5. Host calibrates and issues a COMWAKE.
6. Device responds – The device detects the COMWAKE signal on its RX pair and calibrates its transmitter (optional). Following calibration, the device sends a six burst COMWAKE signal and then sends a continuous stream of the ALIGN sequence. After 2048 ALIGN Dwords have been sent without response from the host as determined by detection of ALIGN primitives received from the host, the device may assume that the host cannot communicate at that speed. If additional legacy speeds are available, the device will try the next fastest speed by sending 2048 ALIGN dwords at that rate. This step is repeated for as many legacy speeds are supported. Once the lowest speed has been reached without response from the host, the device will enter an error state.
7. Host locks – after detecting the COMWAKE, the host starts transmitting D10.2 characters (see 6.7.6) at its lowest supported rate. Meanwhile, the host receiver locks to the ALIGN sequence and, when ready, returns the ALIGN sequence to the device at the same speed as received. A host must be designed such that it can acquire lock given 2048 ALIGN Dwords. The host should allow for at least 32768 Gen1 dwords (880us) after detecting the release of COMWAKE to receive the first ALIGN. This will ensure interoperability with multi-generational and synchronous devices. If no ALIGN is received within 32768 Gen1 dwords (880us), the host shall restart the power-on sequence – repeating indefinitely until told to stop by the application layer.
8. Device locks – the device locks to the ALIGN sequence and, when ready, sends the SYNC primitive indicating it is ready to start normal operation.
9. Upon receipt of three back-to-back non-ALIGN primitives, the communication link is established and normal operation may begin.

### 6.7.4.3 COMINIT

COMINIT always originates from the drive and requests a communication initialization. It is electrically identical to the COMRESET signal except that it originates from the device and is sent to the host. It is used by the device to request a reset from the host in accordance to the sequence shown in Figure 31, below.

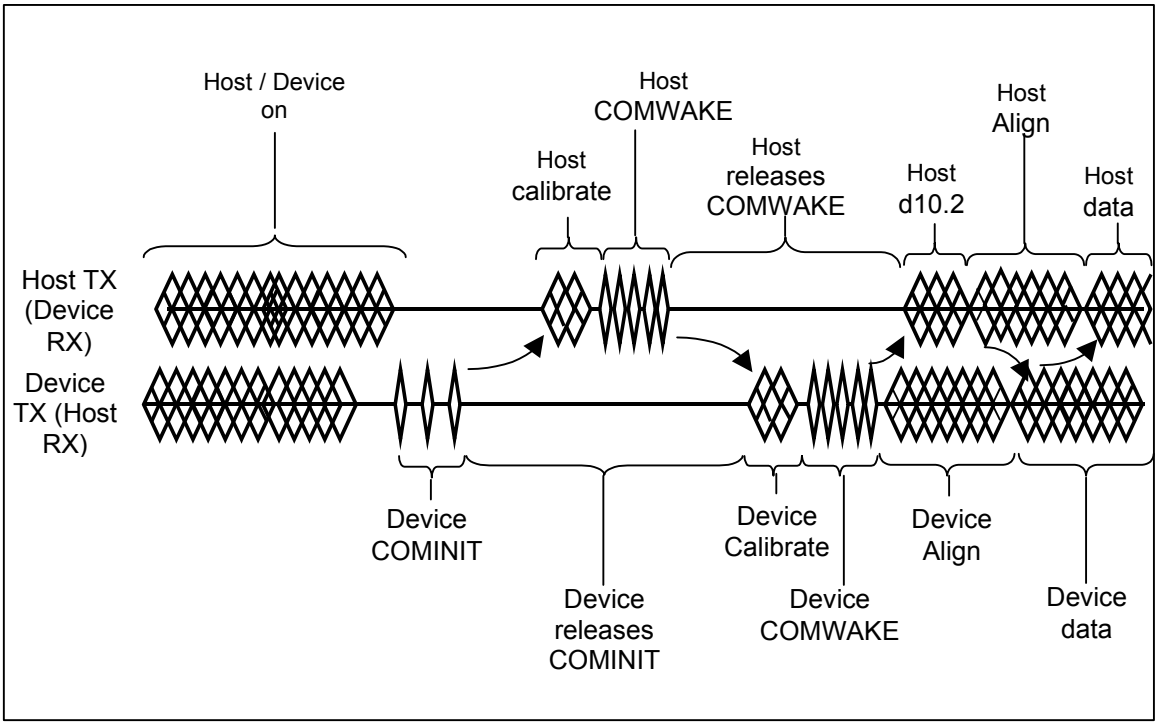



**Figure 31 – COMINIT sequence**

Description:

1. Host/device are powered and operating normally with some form of active communication.
2. Some condition in the device causes the device to issues a COMINIT
3. Host calibrates and issues a COMWAKE.
4. Device responds – The device detects the COMWAKE signal on its RX pair and calibrates its transmitter (optional). Following calibration, the device sends a six burst COMWAKE signal and then sends a continuous stream of the ALIGN sequence. After 2048 ALIGN Dwords have been sent without response from the host as detected by the envelope detector, the device may assume that the host cannot communicate at that speed. If additional legacy speeds are available, the device will try the next fastest speed by sending 2048 ALIGN dwords at that rate. This step is repeated for as many legacy speeds are supported. Once the lowest speed has been reached without response from the host, the device will enter an error state.
5. Host locks – after detecting the COMWAKE, the host starts transmitting D10.2 characters (see 6.7.6) at its lowest supported rate. Meanwhile, the host receiver locks to the ALIGN sequence and, when ready, returns the ALIGN sequence to the device at the same speed as received. A host must be designed such that it can acquire lock given 2048 ALIGN Dwords. The host should allow for at least 32768 Gen1 dwords (880us) after detecting the release of COMWAKE to receive the first ALIGN. This will ensure interoperability with multi-generational and synchronous devices. If no ALIGN is received within 32768 Gen1 dwords (880us), the host shall restart the power-on sequence – repeating indefinitely until told to stop by the application layer.
6. Device locks – the device locks to the ALIGN sequence and, when ready, sends the SYNC primitive indicating it is ready to start normal operation.
7. Upon receipt of three back-to-back non-ALIGN primitives, the communication link is established and normal operation may begin.

### 6.7.4.4 COMWAKE

COMWAKE can originate from either the host controller or the device. It is signaled by transmitting six bursts of data separated by an idle bus condition.

The OOB COMWAKE signaling shall consist of no less than six data bursts, including inter-burst temporal spacing.

Each burst shall be 160 Gen1 UI's long and each inter-burst idle state shall be 160 Gen1 UI's long. A COMWAKE detector will look for four consecutive burst with a 106.7 ns spacing (nominal).

Any spacing less than 55 ns or greater than 175 ns shall invalidate the COMWAKE detector output. The COMWAKE OOB signaling is used to bring the Phy out of a power-down state (PARTIAL or SLUMBER) as described in section 6.8. The interface must be held inactive for at least 175ns after the last burst to ensure far-end detector detects the deassertion properly. The device may hold the interface inactive no more then 228.3ns (175ns + 2 Gen1 dwords) at the end of a COMWAKE to prevent susceptibility to crosstalk.

### 6.7.4.5 Design example

This section is informative and represents one possible design example for detecting COMRESET/COMINIT and COMWAKE. Other design implementations are possible as long as they adhere to the requirements listed in this specification.

The output of the squelch detector is fed into four frequency comparators. When the period is within the window determined by the RC time constants for three consecutive cycles, the appropriate signal is asserted.

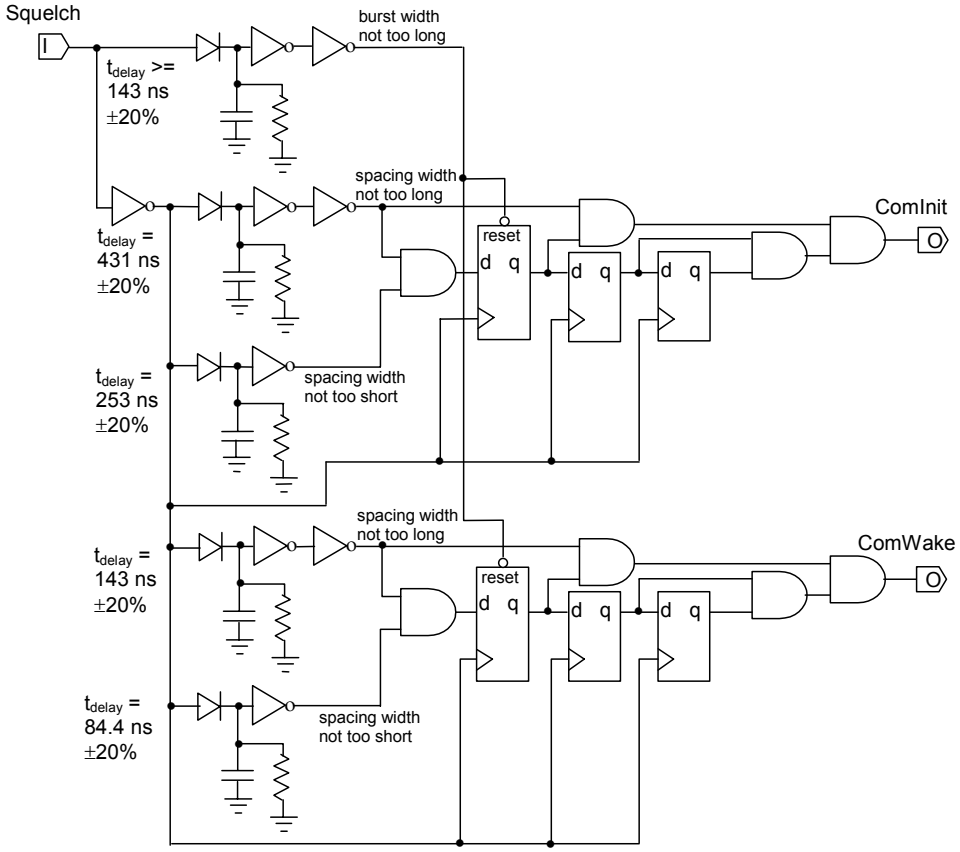



**Figure 32 – OOB signal detector**

The Squelch detector example below makes use of a receiver with built-in hysteresis to filter out any signal not meeting the minimum amplitude. The squelch detector receiver must be true differential to ensure common-mode noise is rejected.

The full-swing output is fed into a pulse generator that charges up the capacitor through the diode. In the absence of signal, a resistor discharges the capacitor to ground. The circuit outputs a true signal when the capacitor voltage is below the turn-on threshold of the Schmitt trigger buffer – indicating insufficient signal level. This circuit must be enabled in all power management states and should, therefore, be implemented with a small power budget.

Figure 33, like the OOB Signal Detector figure shown in Figure 32, is intended to show functionality (informative) only, and other solutions may be used to improve power consumption as long as they comply to the electrical specifications of section 6.6.2, for the worst case noise environment (common-mode) conditions.

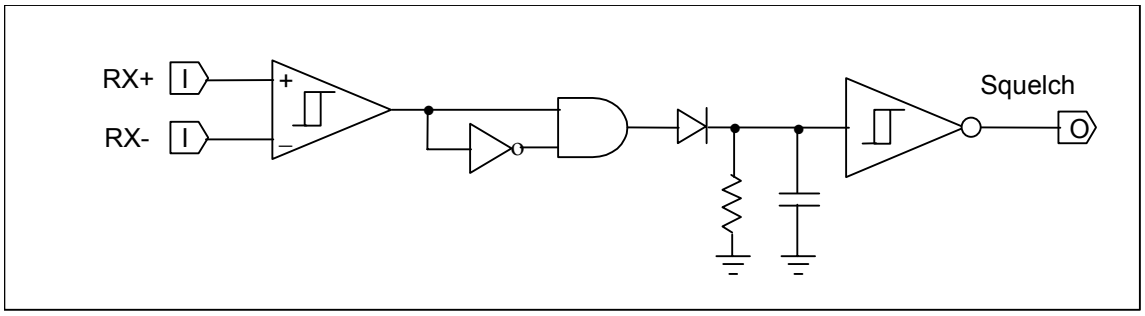



**Figure 33 – Squelch detector**

### 6.7.5 Idle bus condition

During power management states (Partial and Slumber), the electrical interface shall maintain the proper common-mode levels, as cited in section 6.6.2, with zero differential on both signal pairs (all four conductors at 250 mV) for all interface scenarios, except for the case where both, the device and the host-controller, are AC-coupled and the conductor pairs are allowed to float.

All transmitter designs must ensure that transition to and from the idle bus condition do not result in a disturbance in the differential baseline on the conductors. To accomplish this, an AC-coupled transmitter must hold its outputs at zero differential with the same common-mode level as normal operation when in the partial power management mode. When operating in the slumber power management mode, the common mode level of the AC coupled transmitter is allowed to float (while maintaining zero differential) as long as it remains within the limits cited in section 6.6.2.

It is unacceptable to hold the TX outputs at a logical zero or one state during the idle bus condition since this will result in a baseline shift when communications are resumed.

### 6.7.6 Elasticity buffer management

For non-tracking implementations elasticity buffer circuitry may be required to absorb the slight differences in frequencies between the host and device. The greatest frequency difference result from a SSC compliant device talking to a non SSC device. The average frequency difference will be just over 0.25% with excursions as much as 0.5%.

The Serial ATA standard is written to support both tracking and non-tracking architectures. A non-tracking architecture shall contain the elasticity buffer within the Phy layer.
Note that since this elasticity buffer will be designed to have finite length, there needs to be a mechanism at the physical layer protocol level that allows this receiver buffer to be reset without dropping or adding any bits to the data stream. This is especially important during reception of long continuous streams of data. This physical layer protocol must not only support oversampling architectures but must also accommodate unlimited frame sizes (the frame size is limited by the CRC polynomial).

The Link Layer shall keep track of a resettable counter that rolls over at most every 1024 transmitted characters (256 Dwords). Prior to, or at the pre-roll-over point (all 1's), the Link Layer shall trigger the issuance of dual, consecutive ALIGN primitives which shall be included in the Dword count.

After communications have been established, the first and second words out of the Link Layer shall be the dual-ALIGN primitive sequence, followed by at most 254 non-ALIGN Dwords. The cycle repeats starting with another dual-consecutive ALIGN primitive sequence. The Link may issue more

than one dual ALIGN primitive sequence but shall not send an unpaired ALIGN primitive (i.e. ALIGN primitives are always sent in pairs) except as noted for retimed loopback.

The ALIGN primitive consists of the following four characters

| (rd+) | (rd-) | |
|---|---|---|
| 1100000101 | 0011111010 | Align1 (K28.5) |
| 0101010101 | 0101010101 | Align2 (D10.2) |
| 0101010101 | 0101010101 | Align3 (D10.2) |
| 1101100011 | 0010011100 | Align4 (D27.3) |

### 6.7.7 Test considerations

#### 6.7.7.1 Physical plant as a system

A number of components of the system make up the system's performance, not just by mutually exclusively summing up the low-level error parameters detected at the bottom of the protocol stack can be made, but also including the retry algorithms.

There are two basic classes of errors that will affect the bit-error rate performance that have to be considered: bit-errors, and burst-errors. The following section addresses the methodology, for in-system bit-error-rate computations.

For in-system testing, the criteria for testing to a specified maximum frame error rate will be addressed, using a set of reference frames, defined by a specific set of ordered test patterns within the frame.

##### 6.7.7.1.1 Test bit patterns and sequence characteristics

Test bit sequences are those bit sequences that are transmitted onto a serial link, so as to test the Serial ATA interface's jitter compliance, as well as its derived communications-link performance.

As discussed earlier, jitter is generally classified as Random Jitter (RJ), and Deterministic Jitter (DJ). RJ is Gaussian in nature, adds on a power basis, and is measured as an RMS value. DJ is also referred to as systematic jitter, and usually is introduced into the serial link due to duty cycle distortion, power supply noise, substrate noise, or inter-symbol interference.

We will focus on various types of bit sequence patterns that emphasize low/high transition density patterns, as well as low/high frequency patterns.

a) Low transition density patterns are those patterns containing long runs of ones and zeroes, intended to create inter-symbol interference by varying the excursion times at either extreme of the differential signaling levels.

b) High transition density patterns are those patterns containing short runs of ones and zeroes, also intended to create inter-symbol interference.

c) Bit patterns that contain low frequency spectral components are a good test of the input high pass filter circuitry, more specifically, introduced amplitude signal distortion, due to a marginal design. These bit patterns are a better test than those bit patterns having high frequency spectral content.

d) Simultaneous switching outputs patterns are achieved by transmitting alternating 1's complement bit patterns (10-bits) for recovery at the receiver. These patterns create worst case power supply, or chip substrate, noise, and are achieved by selecting bit test pattern sequences that maximize current extremes at the recovered bit pattern parallel interface. These patterns induce Ldi/dt noise into substrate supply, and are a good test of the receiver circuitry.

e) The intent of random bit patterns is to provide those patterns containing sufficiently broad spectral content, and minimal peaking, that can be used for both component, and system level architecture measurement of jitter output, and bit-error-rate performance. These patterns are also intended to be the common baseline pattern stimulus, for system/component vendor comparative testing, attributing the Transmit jitter output measurement to the component performance, and not to the spectral profile of the data pattern used.

These patterns will be used for compliance testing of the Serial ATA interfaces. The patterns will be classified in two categories:

a) Non-compliant patterns
b) Compliant patterns

Non-compliant patterns are those patterns that are used for baseline jitter measurements, and assessment of signal quality, given specified stimulus. These patterns do not comply to the required FIS formats, but are just a repeated selected set of 8b/10b characters.

Compliant patterns are those specified patterns that do contain the leading SOF primitive, the specified pattern as data content, and trailing CRC, and EOF primitives. There is no suppression of the dual-consecutive ALIGN primitive during stimulus with this class of pattern.

The test patterns illustrated in the following sections are indicated to start with negative running disparity for illustrative purposes only in order to convey the encoded 10b patterns transmitted on the wire for each sequence.

#### 6.7.7.1.1.1 Low transition density bit pattern sequences

Low transition density bit patterns, as shown in Figure 34 below, are those patterns containing long runs of ones and zeroes. These patterns create high frequency jitter due to inter-symbol interference, emphasized further when part of the composite pattern described in a later section.

Low transition density pattern (SATA)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| - | D17.7 (-) | D30.7 (+) | D7.1 (+) | D14.7 (+) | D30.7 (-) | D7.6 (-) | D30.3 (-) | D30.3 (+) |
| | F1h | FEh | 27h | EEh | FEh | C7h | 7Eh | 7Eh |
| | 100011 0111 | 100001 1110 | 000111 1001 | 011100 1000 | 011110 0001 | 111000 0110 | 011110 0011 | 100001 1100 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| D30.3 (-) | D30.3 (+) | D30.3 (-) | D30.3 (+) | D30.3 (-) | D30.3 (+) | D30.3 (-) | D30.3 (+) |
| 7Eh | 7Eh | 7Eh | 7Eh | 7Eh | 7Eh | 7Eh | 7Eh |
| 011110 0011 | 100001 1100 | 011110 0011 | 100001 1100 | 011110 0011 | 100001 1100 | 011110 0011 | 100001 1100 |

Byte group (D30.3(-), D30.3(+)) repeat n times (n>12)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| D30.3 (-) | D30.3 (+) | D30.3 (-) | D30.3 (+) | D3.7 (-) | D28.7 (+) | D3.7 (-) | D28.7 (+) |
| 7Eh | 7Eh | 7Eh | 7Eh | E3h | FCh | E3h | FCh |
| 011110 0011 | 100001 1100 | 011110 0011 | 100001 1100 | 110001 1110 | 001110 0001 | 110001 1110 | 001110 0001 |

**Figure 34 – Low transition density pattern**

#### 6.7.7.1.1.2 High transition density bit pattern sequences

High transition density patterns are those patterns containing short runs of ones and zeroes, also intended to create high frequency jitter due to inter-symbol interference, emphasized further when part of the composite pattern described in a later section. For this case there are two sub-classes of high-transition density patterns:

a) Half-rate high transition density bit pattern sequence
b) Quarter-rate high transition density bit pattern sequence

An amalgamate of the two sub-classes of high-transition density bit patterns will be used to represent high transition density test pattern.

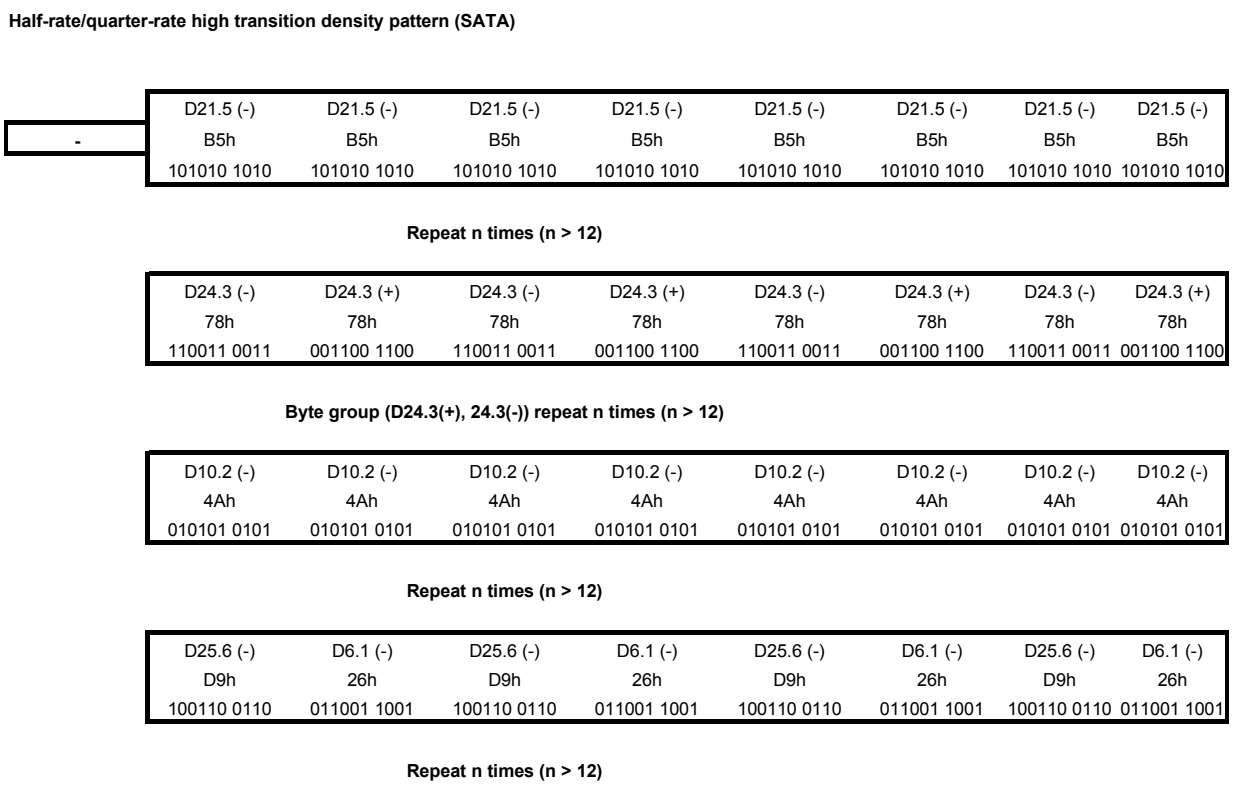



**Figure 35 - Half-rate / quarter-rate high transition density pattern**

#### 6.7.7.1.1.3 Low frequency spectral content bit pattern sequences

Bit patterns that contain low frequency spectral components are a good test of the input high pass filter circuitry, as far as introduced signal distortion, due to a marginal design.

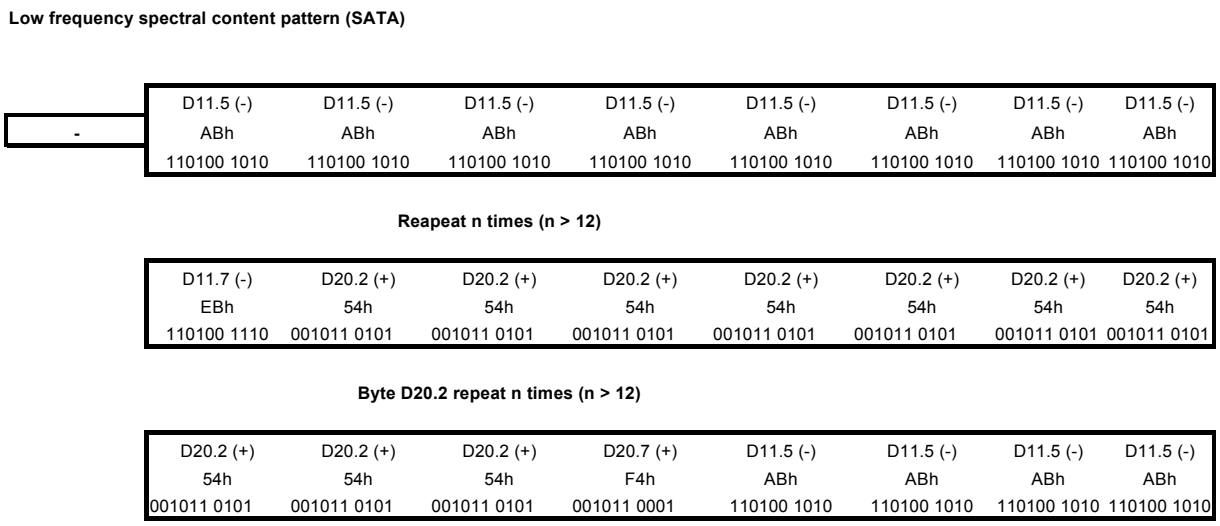



**Figure 36 – Low frequency spectral content pattern**

#### 6.7.7.1.1.4 Simultaneous switching outputs bit pattern sequences

Simultaneous switching outputs bit pattern sequences induce Ldi/dt noise into substrate supply, and is a good test of the receiver circuitry. This is achieved by transmitting alternating 1's complement bit patterns (10-bits) for recovery at the receiver.

**Simultaneous switching outputs patterns (SATA)**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | D31.3 (-) | D31.3 (+) | D31.3 (-) | D31.3 (+) | D31.3 (-) | D31.3 (+) | D31.3 (-) | D31.3 (+) |
| - | 7Fh | 7Fh | 7Fh | 7Fh | 7Fh | 7Fh | 7Fh | 7Fh |
| | 101011 0011 | 010100 1100 | 101011 0011 | 010100 1100 | 101011 0011 | 010100 1100 | 101011 0011 | 010100 1100 |

**Repeat n times (n > 512)**

**Figure 37 – Simultaneous switching outputs patterns**

#### 6.7.7.1.1.5 Composite bit pattern sequences

For the measurement of jitter, patterns should combine low frequency, low transition density, and high transition density patterns. All these combinations, but the Low Frequency Spectral Content class can be performed for relatively short test time intervals, for jitter and performance measurements.

The lower frequency pattern needs to be tested for longer interval periods to be able to observe the lower frequency jitter effects on the interface.

By including a composite set of the cited patterns, the resultant composite pattern stresses the interface components within the link with low and high frequency jitter, tests for component, and various amplitude distortions due to marginal receiver input circuitry, or interface components.

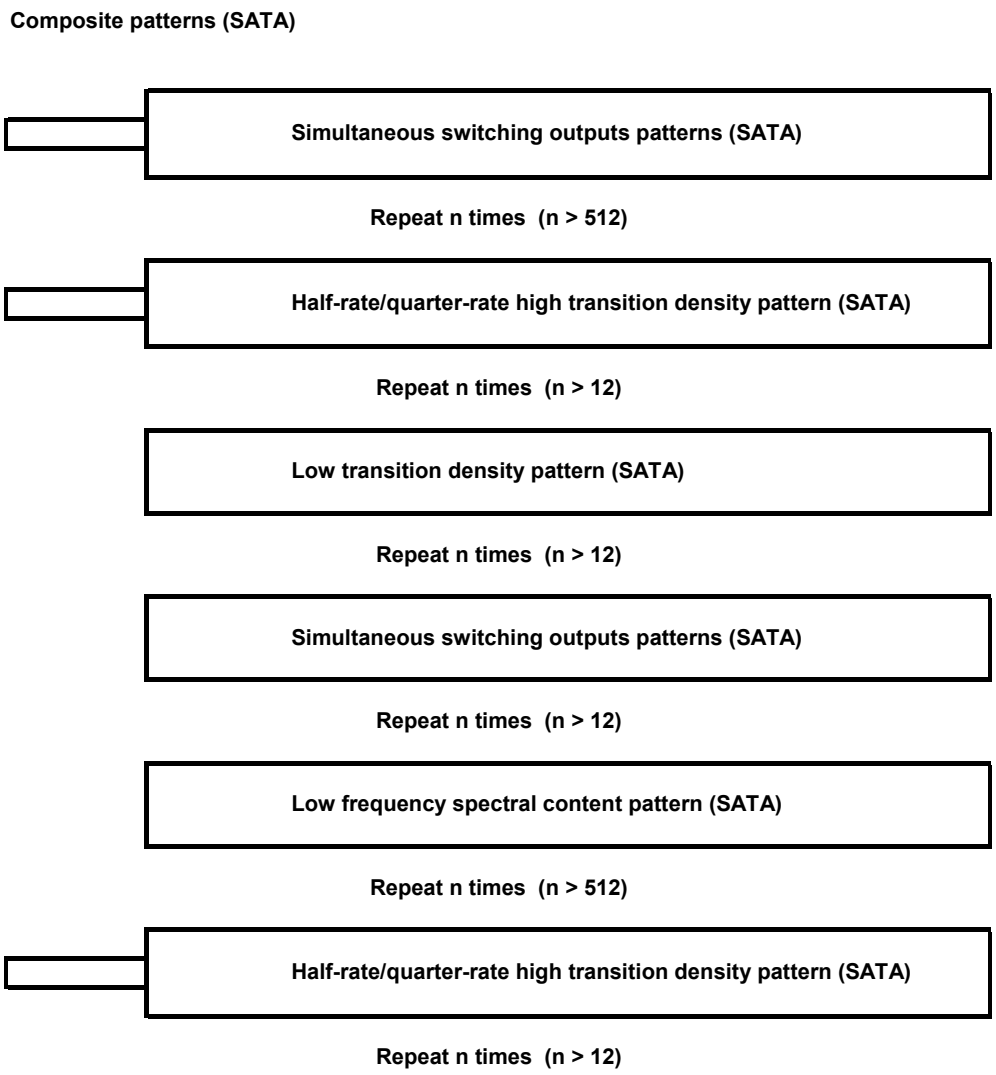



**Figure 38 – Composite patterns**

### 6.7.7.1.2 Bit error rate testing - Informative

In order to get a fair assessment of bit-error-rate performance, bit-errors, as well as burst errors, must be considered separately.

Note that a single bit error has a high probability of causing a byte-wise error, or an 8b/10b code violation error, due to the 8b/10b encoding, thus a single bit error translates to 8-bits or 10-bits of error. We will classify each of these occurrences, as "byte errors", and these form the basis for the bit-error-rate calculations.

Furthermore, a missing or an extra bit detected by the receiver will, translate into a series of errors that spans across multiple byte-boundaries until bit re-alignment via ALIGN primitives as per 6.7.6 - Elasticity buffer management, worst case. This series of errors will be defined as burst errors.

A third type of byte-wise error exists where, an entire byte is not received, this type of error will not be classified differently, as it will cause a burst error whose span can be limited by higher-layer protocol transmission conventions, at the cost of additional overhead ALIGN primitive sequences.

A single error may result in several related errors occurring closely together which in turn may result in multiple bit-error counts. A character might have a single bit error in it that causes a code-violation error. A disparity error might occur on a following character, caused by the same single error. To prevent multiple error counts from a single cause, the following concept of an error burst is introduced:

- An “error burst” is defined as a time period 1.5 +/- 0.5 seconds long, during which one or more invalid transmission words are recognized. This time may be exceeded due to infrequent unusual conditions.

- Only one error in an error burst shall be counted toward the error-burst-rate threshold.

#### 6.7.7.1.2.1 Error-burst-rate-thresholding measurement - Informative

The error-burst-rate thresholding process is designed to detect an increased error rate before performance degradation becomes serious. Measurement of error-burst-rate thresholding is accomplished by counting the number of error bursts that occur in a 5-minute period, when tested with the compliant frame patterns, defined by section 6.7.7.1.1.1, but with the N parameters extended so as to achieve the maximum frame length. When the count equals a specific number, the threshold is exceeded. This specific number is called the “threshold error count.”

An error-burst-rate threshold is detected when 15 error bursts occur in a 5-minute period. The required accuracy of the 5-minute period mentioned above is +/-0.3 seconds. Reaching the threshold error count of 15 in a 5-minute period is a positive indication of the interface failing the error rate requirements and may be used to terminate testing. However, not reaching the threshold may not be taken as indication that the interface is passing without also confirming that the frame error rate requirements are satisfied using statistically sound measurement techniques. Being under the threshold error count after a 5-minute period should therefore never be used as a measure of system compliance.

The error-burst-rate counting process will be restarted when PHY Ready state is entered, and when an amount of time has elapsed after a error-burst-rate threshold is detected. After an error-burst-rate threshold is detected, at least 15 additional error bursts will occur before the next error-burst-rate threshold is detected. In addition, the error-burst counting process may be restarted whenever the 5-minute time period has expired even though an error-burst-rate threshold is not reached.

#### 6.7.7.1.2.2 Bit-error-rate measurements - Informative

Bit error rates if measured and computed, byte-wise, should be no greater than $10^{-12}$ bit-errors when tested with the reference test patterns, cited in section 6.7.7.1.1.

Note that the Frame-Error Rate measurements constitute the basis for the applicable test requirements for this specification.

### 6.7.7.1.3 Frame error rate testing

Frame error rate testing is the measure of link performance using all the intermediate circuit blocks in the chain from low-level Phy Layer, Link Layer, through Transport Layer. Error detection is at the frame level using the CRC (Cyclic Redundancy Check) error detection mechanism, and respective reporting to the higher layer levels..

#### 6.7.7.1.3.1 Frame error-rate patterns

Frame error rate patterns will contain the elements of the test bit patterns and sequence of patterns, so as to thoroughly be able to test the serial interface in system, using the higher level CRC error detection/reporting from the lower protocol level layers to the Application level layer.

The frame patterns shall be comprised of the set of the Composite Patterns, cited in section 6.7.7.1.1.5, but with the N parameters extended so as to achieve the maximum frame length.

#### 6.7.7.1.3.2 Frame error-rate measurements

Frame Error Rates shall be measured and computed, framewise, to be no greater than ($8.196*10^{-8}$) frame errors when tested with any given 8b/10b pattern, including the Frame-Error-Rate reference patterns cited in section 6.7.7.1.3.1.

Note that the cited patterns should appear on the wire, and the N parameters of the reference patterns shall be extended to achieve the maximum frame length, as specified in paragraph 7.5.1.

### 6.7.7.1.4 Loopback testing

Three types of Loopback test schemes have been identified, where certain of these will not form part of this specification, but will be classified as Vendor Specific.

a) Far-End Retimed - Required
b) Far-End Analog - Vendor Specific
c) Near-End Analog (Effectively Retimed) - Vendor Specific

#### 6.7.7.1.4.1 Loopback -- Far end retimed

Figure 24 below, illustrates the scope, at the architectural block diagram level, of the Far-End Retimed loopback. As this loopback scheme needs a specific action from the far-end connected interface, this mode shall be entered by way of the BIST FIS described in section 8.5.7.

The Far-End Interface shall remain in this Far-End Retimed Loopback, until receipt of the COMRESET/COMINIT OOB Signaling sequence.

As a minimum, Far-End Retimed Loopback shall involve far-end circuitry such that the datastream, at the Far-End interface, is extracted by the Serializer/Deserializer (SerDes) and data recovery circuit (DRC) before being sent back through the SerDes and Transmitter with appropriately inserted retiming ALIGN primitives as described in section 8.5.6.1. The data may be decoded and descrambled in order to provide testing coverage for those portions of the device, provided the data is re-scrambled using the same sequence of scrambler syndromes. The returned data shall be the same as the received data with the exception that the returned data may be encoded with different starting running disparity.

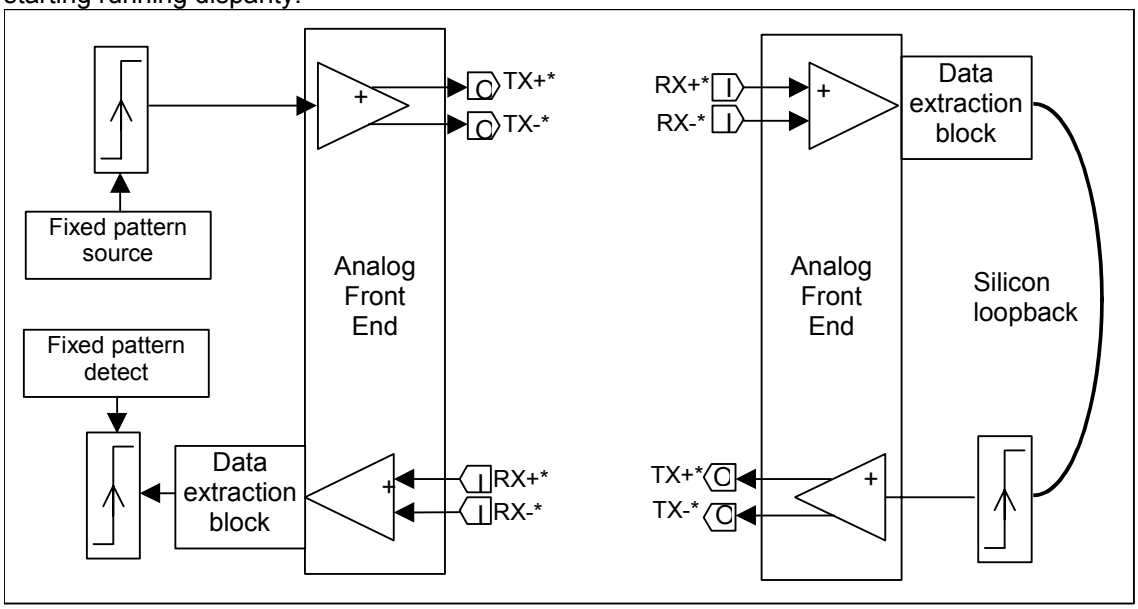



**Figure 39 – Loopback far-end retimed**

#### 6.7.7.1.4.2 Loopback -- far-end analog (vendor specific)

Figure 40 below, illustrates the scope, at the architectural block diagram level, of the Far-End Analog loopback. As this loopback scheme needs a specific action from the far-end connected interface, this mode, if implemented, shall be entered by way of the BIST FIS described in section 8.5.7.

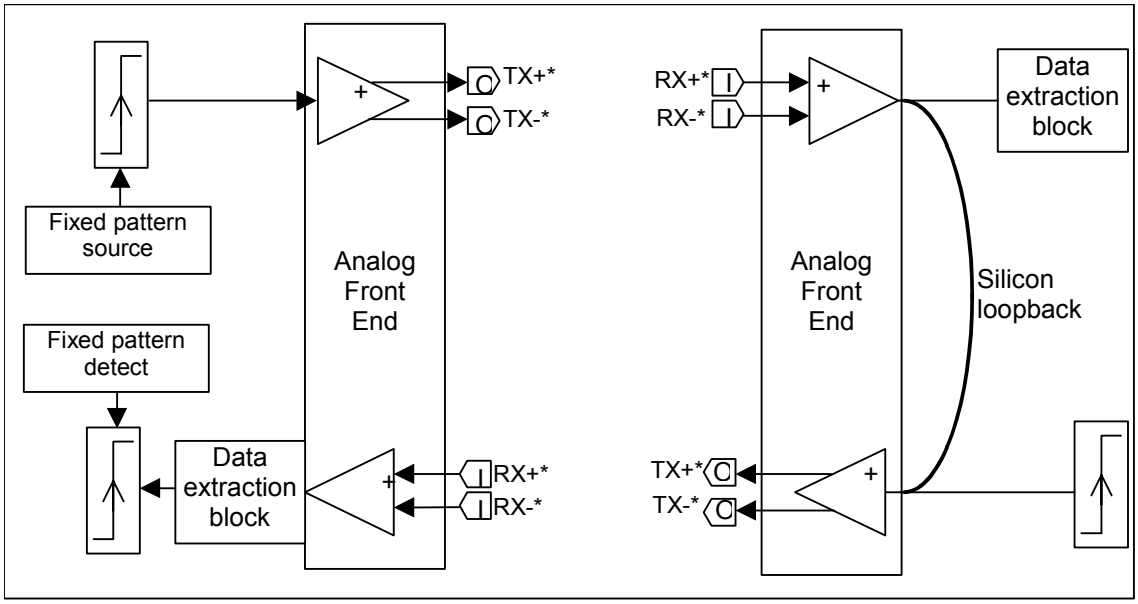



**Figure 40 – Loopback far-end analog**

If implemented, the Far-End Interface shall remain in this Far-End Analog Loopback mode, until receipt of the COMRESET/COMINIT OOB Signaling sequence.

#### 6.7.7.1.4.3 Loopback -- near-end analog (vendor specific)

Figure 41 below, illustrates the scope, at the architectural block diagram level, of the Near-End Analog loopback. This loopback scheme, if implemented, needs the far-end connected interface to be in a non-transmitting mode, such as Slumber, or Partial interface power management states.

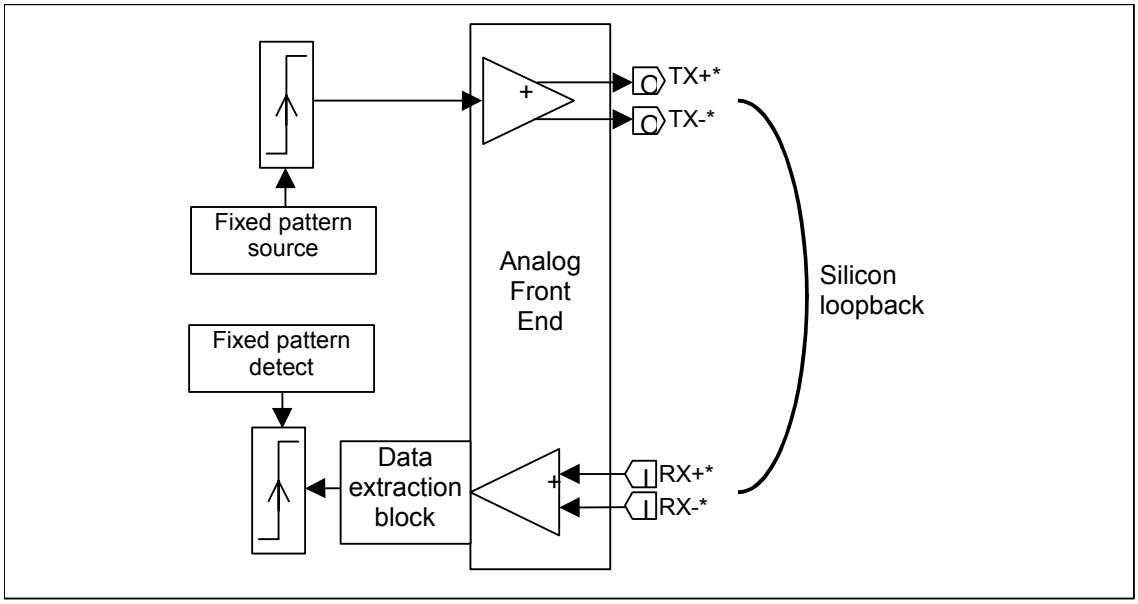



**Figure 41 – Loopback - near-end analog**

### 6.7.7.1.5 Test requirements

Test patterns cited in this section used as stimulus that will be used to verify the Serial ATA interface compliance and signal integrity, using the following test models:

a) Non-compliant test patterns for jitter measurements, physical connection media tests, and electrical parameter testing.
b) Compliant test patterns for frame error rate testing and in-system tests.

#### 6.7.7.1.5.1 Test requirements - non-compliant patterns

Electrical parameters of section 6.6.2 shall be verified using the following set of test patterns, using the BIST FIS, Far-End Transmit Mode, described in section 8.5.7:

a) Low transition density bit patterns, as per section 6.7.7.1.1.1
b) High transition density bit patterns as per section 6.7.7.1.1.2
c) Low frequency spectral component bit patterns as per section 6.7.7.1.1.3
d) Simultaneous switching outputs bit patterns as per section 6.7.7.1.1.4

#### 6.7.7.1.5.2 Test requirements - compliant frame patterns

The frame error rates specified in section 6.7.7.1.3.2 shall be tested for compliance when subjected to any implementation-determined worst-case compliant patterns, as well as the following set of compliant patterns:

a) Compliant low transition density bit patterns, as per section 6.7.7.1.1.1
b) Compliant high transition density bit patterns as per section 6.7.7.1.1.2
c) Compliant low frequency spectral component bit patterns as per section 6.7.7.1.1.3
d) Compliant simultaneous switching outputs bit patterns as per section 6.7.7.1.1.4
e) Compliant composite patterns as per section 6.7.7.1.1.5

Where the qualifying prefix term "compliant" signifies transmission of the cited pattern encapsulated in the data portion of the Frame Information Structure, and used in a Serial ATA operational transmission context.

Note that the cited patterns should appear on the wire, and the N parameters of the reference patterns shall be extended to achieve the maximum frame length. These compliant patterns contain the necessary SOF leading primitive, the calculated CRC, and the trailing EOF primitive, as shown in the Figure below:

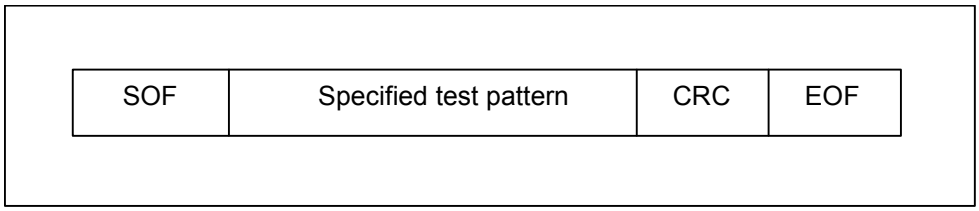



**Figure 42 – Compliant test patterns**

#### 6.7.7.1.6 Test requirements - loopback

As specified earlier in section 6.7.7.1.4, only one of the three types of Loopback test schemes will form part of this specification: Far-End Retimed Loopback. Both Far-end and Near-End Analog Loopback schemes are classified as Vendor Specific, thus will not have specified test requirements, herein.

##### 6.7.7.1.6.1 Test requirements - loopback - far-end retimed

Generation of loopback test patterns is optional and vendor specific. As this loopback scheme needs a specific action from the far-end connected interface, this mode shall be entered by way of the BIST FIS described in section 8.5.7.

This loopback test is intended for a relatively quick assessment of interface health, accommodating an in-system test capability,

In this Far-End Retimed Loopback mode, the interface shall operate without a CRC error for a sustained minimum period of 1000 ms at Gen1 speed, using the following set of compliant reference frame patterns if the test pattern generation is supported by the initiating device:

a) Compliant composite reference frame patterns as specified in section 6.7.7.1.3.1.
b) Compliant low transition density bit patterns, as per section 6.7.7.1.1.1
c) Compliant high transition density bit patterns as per section 6.7.7.1.1.2.
d) Compliant low frequency spectral component bit patterns as per section 6.7.7.1.1.3.
e) Compliant simultaneous switching outputs bit patterns as per section 6.7.7.1.1.4
f) Compliant composite patterns as per section 6.7.7.1.1.5

Where the qualifying prefix term "compliant" signifies transmission of the cited pattern encapsulated in the data portion of a valid Frame Information Structure, and used in a Serial ATA operational transmission context.

The Far-End Interface shall remain in this Far-End Retimed Loopback, until receipt of the COMRESET/COMINIT OOB Signaling sequence.

##### 6.7.7.1.6.2 Test requirements - loopback - far-end analog (vendor specific)

The test requirements for this Loopback scheme, classified as Vendor Specific, are to be set by the Vendor.

##### 6.7.7.1.6.3 Test requirements - loopback - near-end analog (vendor specific)

The test requirements for this Loopback scheme, classified as Vendor Specific, are to be set by the Vendor.

#### 6.7.7.1.7 Test requirements - OOB signaling tests

Out-of-band signaling is used to signal specific actions during conditions where the receiving interface is in an active mode, a low interface power state, or a test mode.

This section specifies the set of test requirements to ensure that the OOB detector circuits will comply to the OOB signaling sequences under various conditions.

#### 6.7.7.1.7.1.1 Power-on sequence

#### 6.7.7.1.7.1.2 Calibration

When the host-controller performs impedance calibration, it shall adjust its own impedance such that the electrical requirements of section 6.6.2 are satisfied.

#### 6.7.7.1.7.1.3 Speed negotiation

Speed Negotiation and transition to lower serial interface data rates shall be implemented for Gen2 compatible interfaces, negotiating and transitioning down to Gen1 speeds. There is no requirement for speed negotiation and transition to lower speeds than Gen1.

References to Gen1 and Gen2 speeds pertain to the definitions, per section 2.2.

### 6.7.7.1.7.2 Interface power management sequences

#### 6.7.7.1.7.2.1 Partial

The interface shall detect the OOB signaling sequence "COMWAKE" and "COMRESET" when in the Partial Interface power management state.

While in the "Partial" state, the interface shall be subjected to the low-transition density bit pattern sequences of section 6.7.7.1.1; the interface shall remain in the Partial state until receipt of a valid COMWAKE (or COMRESET) OOB signaling sequence.

Power dissipation in this Partial state shall be measured or calculated to be less than the Phy Active State, but more than the Slumber State defined in section 6.8.

The requirement for a "not-to-exceed" power dissipation limit in the partial interface power management state will be classified as vendor specific, and will be documented as part of the implementation performance specifications.

#### 6.7.7.1.7.2.2 Slumber

The interface shall detect the OOB signaling sequence "COMWAKE" and "COMRESET" when in the Slumber Interface power management state.

While in the "Slumber" state, the interface shall be subjected to the low-transition density bit pattern sequences of section 6.7.7.1.1; the interface shall remain in the Slumber state until receipt of a valid COMWAKE (or COMRESET) OOB signaling sequence.

Power dissipation in this Slumber state shall be measured or calculated to be less than the Phy Ready State, and less than the Partial State defined in section 6.8.

The requirement for a "not-to-exceed" power dissipation limit in the Slumber interface power management state will be classified as vendor specific, and will be documented as part of the implementation performance specifications.

## 6.7.7.2 Physical plant sub-modules

### 6.7.7.2.1 Receiver

#### 6.7.7.2.1.1 Jitter tolerance

The requirements for receiver jitter tolerance have been implicitly addressed by other relevant sections.

#### 6.7.7.2.1.2 Differential voltage

When subjected to the high-transition density patterns of section 6.7.7.1.1, the Differential Voltage measured at the Receiver shall comply to the electrical specifications of section 6.6.2.

#### 6.7.7.2.1.3 Common-mode voltage

References to peak-to-peak voltages are cited in section 6.6.2.

The Receiver should operate to within the bit error rates cited in section 6.7.7.1.2, when subjected to a sinusoidal interfering signal with peak-to-peak voltage , and swept from the frequency range extremes, at a sweep rate period no shorter than 33.33 us.

The Receiver shall operate to within the frame error rates cited in section 6.7.7.1.3, when subjected to a sinusoidal interfering signal with peak-to-peak voltage and swept from the frequency range extremes, at a sweep rate period no shorter than 33.33 us.

### 6.7.7.2.2 Transmitter

#### 6.7.7.2.2.1 Jitter output

Jitter output tests are intended to determine the jitter amplitudes of the random jitter, and deterministic jitter components.

##### 6.7.7.2.2.1.1 Jitter measurements

The jitter specifications are based in a data transition to data transition Timing Interval Analyzer (TIA) format. If such a device is not available, equivalent measurements may be made with an oscilloscope. The oscilloscope is set up to trigger on the data and an EYE diagram is examined some integer number of unit intervals (UI) later. The histogram function can be used to monitor the distribution of the zero crossing to extract jitter information. Figure 43 shows an example where the oscilloscope is set up to examine an edge 16 UI away from the trigger point.

The DJ and RJ (or TJ) should be less than the maximum $A_x$ indicated in Figure 43 – Jitter measurement example for $n_x >= 16$. For example, assume $n_0$ is five and $n_1$ is 250. In this case, the DJ and RJ for $t_{16}$ must be less than $A_1$ since 250>16 but need not meet the $A_0$ requirement since 5<16. This method is employed because of the ease of measurement and to enable the specification to be relaxed at lower frequencies. If DJ is large enough, it can be fairly accurately measured from the histogram as the distance between the two outside peaks (there may be more than the three peaks shown in the example). If DJ and RJ are too intermixed for measurement, then other methods must be used. One such method is to transmit a clock pattern (fixed frequency components) to determine the RJ (DJ is zero for fixed frequencies), transmit a random patter to measure the TJ, and derive the DJ from these. Another alternative is to wait until $>>10^{12}$ edges are included in the histogram and measure the Total Jitter (TJ).

Since $t_0$ is triggered from the serial signal rather than a reference clock the resulting measurements do represent a combination of the jitter at $t_0$ and $t_n$.

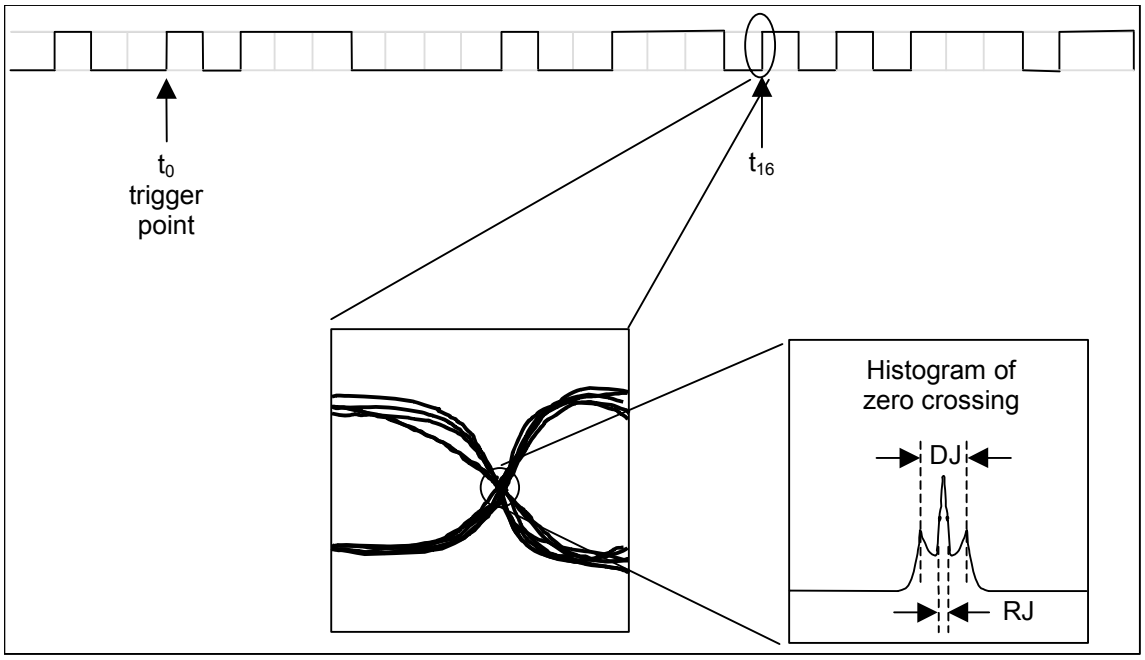



**Figure 43 – Jitter measurement example**

#### 6.7.7.2.2.2 Differential voltage

When subjected to the high-transition density patterns of section 6.7.7.1.1, the Differential Voltage measured at the Transmitter shall comply to the respective electrical specifications of section 6.6.2.

#### 6.7.7.2.2.3 Common-mode voltage

The Transmitter shall comply to the electrical specifications of section 6.6.2, when subjected to a sinusoidal interfering signal with peak-to-peak voltage, and swept from the frequency range extremes, at a sweep rate period no shorter than 33.33 us.

#### 6.7.7.2.2.4 Rise/fall times

Output rise times and fall times are measured between 20% and 80% of the signal, see Signal Rise & Fall Times, Figure 44. Rise and fall time requirements apply to differential transitions, for both In-Band and Out-Of-Band signaling.

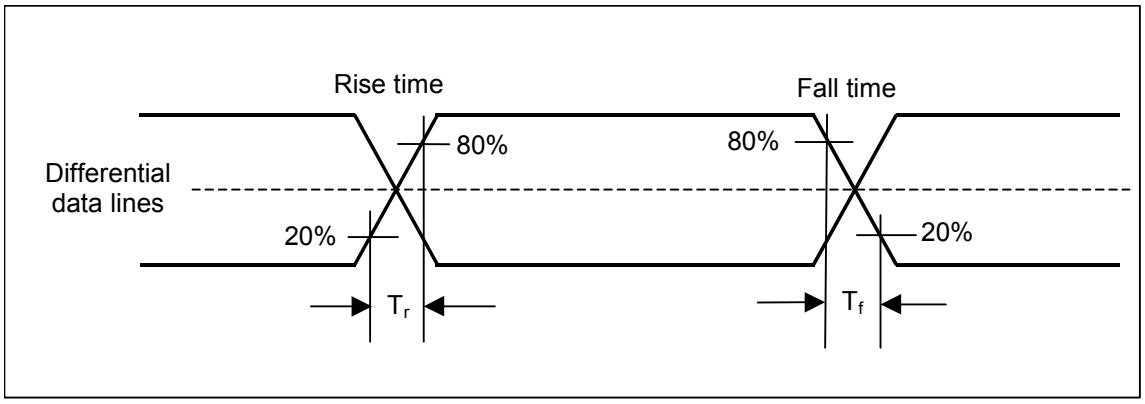



**Figure 44 – Signal rise and fall times**

The rise and fall times for transmitter differential buffer lines are measured with the load fixture shown in Figure 45, at the transmitter interface, and must comply to Electrical Specification paragraph 6.6.2 as well as be matched within ± 10% of each other to minimize RFI emissions and signal skew.

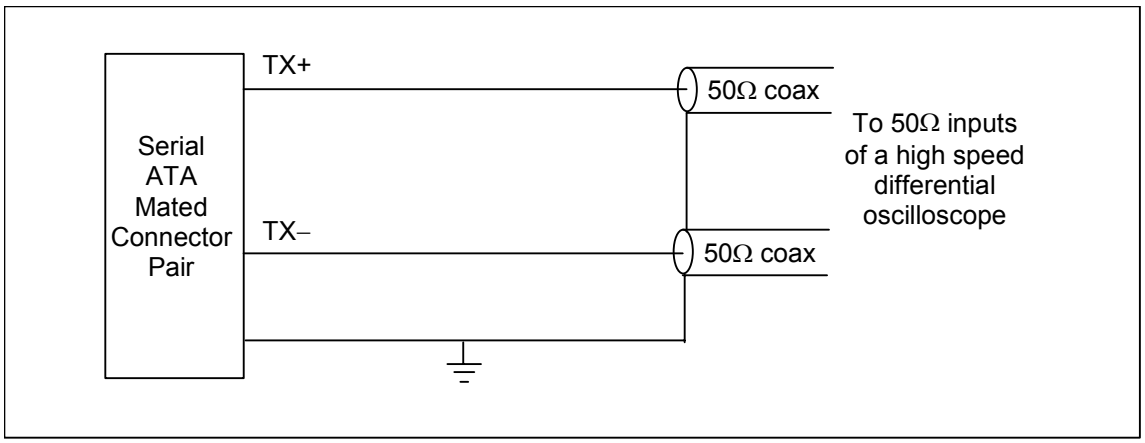



**Figure 45 – Transmit test fixture**

When testing the Serial ATA interface with the specified bit pattern sequences of section 6.7.7.1.1, use the Receiver Test fixture as shown in Figure 46.

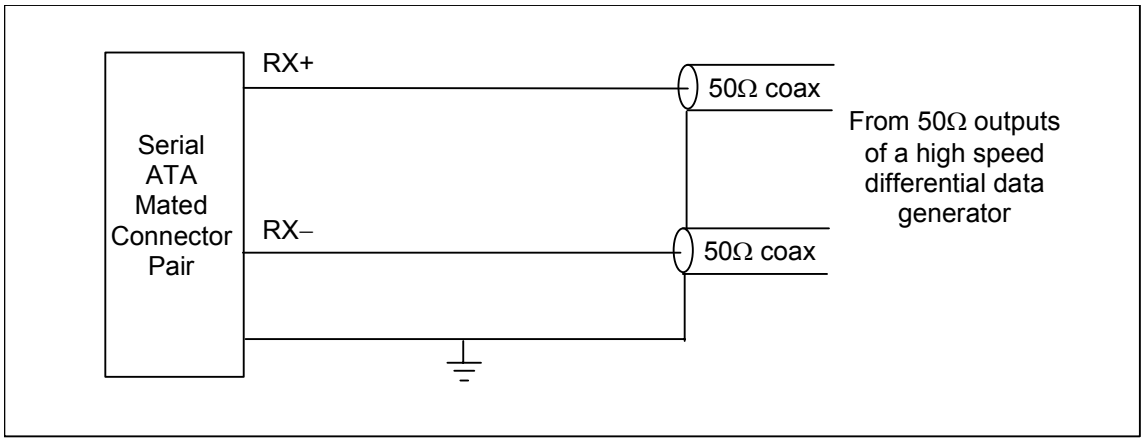



**Figure 46 – Receive test fixture**

## 6.8 Interface power states

Serial ATA Interface Power States will be controlled by the device and host controller. The Serial Interface Power States shall be defined as described in Table 15. – Interface Power States.

| | |
|---|---|
| **PHY READY** | The PHY logic and main PLL are both on and active. The interface is synchronized and capable of receiving and sending data. |
| **Partial** | The PHY logic is powered, but is in a reduced power state. Both signal lines on the interface are at a neutral logic state (common mode voltage). The exit latency from this state shall be no longer than 10μs. |
| **Slumber** | The PHY logic is powered but is in a reduced power state. Both signal lines on the interface are at the neutral logic state (common mode voltage). The exit latency from this state shall be no longer than 10ms. |

**Table 15. – Interface Power States**

### 6.8.1 Interface power state sequences

#### 6.8.1.1 Power-on sequence state diagram

The following state diagrams specify the expected behavior of the host and device PHY from power-on to the establishment of an active communication channel.

In those states where the Phy relies on detection of received ALIGN primitives for state transitions, the Phy may rely on positive decode of the K28.5 leading character or the comma sequence (of either running disparity) as positive indication of ALIGN primitive reception rather than decoding the complete ALIGN primitive.

##### 6.8.1.1.1 Host Phy Initialization state machine

As described in section 6.7.4.3, reception of a COMINIT signal shall cause the host to reinitialize communications with the device and shall unconditionally force the Host Phy state machine to transition to the HP2B:HR_AwaitNoCOMINIT state regardless of other conditions. Reception of COMINIT is effectively an additional transition into the HP2B:HR_AwaitNoCOMINIT state that appears in every Host Phy state. For the sake of brevity, this implied transition has been omitted from all the states.

| HP1: HR_Reset | Transmit COMRESET[2, 3] | | |
|---|---|---|---|
| | 1. Power-on reset and explicit reset request deasserted | → | HR_AwaitCOMINIT |
| | 2. Power-on reset or explicit reset request asserted | → | HR_Reset |
| | NOTE :<br>1. This state is entered asynchronously any time in response to power-on reset or an explicit reset request.<br>2. Must transmit COMRESET for a minimum of 6 bursts (and a multiple of 6)<br>3. As described in section 6.7.4.2, COMRESET may be transmitted for the duration of this state, or it may be transmitted starting in this state and cease transmission after departure of this state, or it may be transmitted upon departure of this state. | | |

| HP2: HR_AwaitCOMINIT | Interface quiescent | | |
|---|---|---|---|
| | 1. COMINIT detected from device | → | HR_AwaitNoCOMINIT |
| | 2. COMINIT not detected from device | → | HR_AwaitCOMINIT |

| HP2B: HR_AwaitNoCOMINIT | Interface quiescent | | |
|---|---|---|---|
| | 1. COMINIT not detected from device | → | HR_Calibrate |
| | 2. COMINIT detected from device | → | HR_AwaitNoCOMINIT |
| | NOTE :<br>1. This state is entered asynchronously any time in response to COMINIT unless during a power-on reset or an explicit reset request (in which case HP1 is entered). | | |

| HP3: HR_Calibrate | Perform calibration[1] | | |
|---|---|---|---|
| | 1. Calibration complete or bypass not implemented | → | HR_COMWAKE |
| | 2. Calibration not complete | → | HR_Calibrate |
| | NOTE :<br>1. Calibration is optional. If bypassed or not implemented, proceed directly to HR_COMWAKE. | | |

| HP4: HR_COMWAKE | Transmit COMWAKE | | |
|---|---|---|---|
| | 1. COMWAKE not detected from device | → | HR_AwaitCOMWAKE |
| | 2. COMWAKE detected from device | → | HR_AwaitNoCOMWAKE |

| HP5: HR_AwaitCOMWAKE | Interface quiescent | | |
|---|---|---|---|
| | 1. COMWAKE detected from device | → | HR_AwaitNoCOMWAKE |
| | 2. COMWAKE not detected from device | → | HR_AwaitCOMWAKE |

| HP5B: HR_AwaitNoCOMWAKE | Interface quiescent | | |
|---|---|---|---|
| | 1. COMWAKE not detected from device | → | HR_AwaitAlign |
| | 2. COMWAKE detected from device | → | HR_AwaitNoCOMWAKE |

| HP6: HR_AwaitAlign | Host transmits D10.2 characters at lowest supported rate[2] | | |
|---|---|---|---|
| | 1. ALIGN detected from device (at any supported speed)[3] | → | HR_AdjustSpeed |
| | 2. ALIGN not detected from device and 880us (32768 Gen1 dwords) has elapsed since entry to HR_AwaitAlign | → | HR_Reset[1,4] |
| | 3. ALIGN not detected from device and less than 880us (32768 Gen1 dwords) has elapsed since entry to HR_AwaitAlign | → | HR_AwaitAlign |
| | NOTE :<br>1. Host retries the power-on sequence indefinitely unless explicitly turned off by the application layer<br>2. Host must start transmitting d10.2 characters no later than 533ns (20 Gen1 dwords) after COMWAKE is deasserted as specified in the out of band signaling section<br>3. Host designers should be aware that the device is allowed 53.3ns (2 Gen1 dwords) after releasing COMWAKE (by holding the idle condition for more than 175ns) to start sending characters. Until this occurs, the bus will be at an idle condition and may be susceptible to crosstalk from other devices. Care must be taken so that crosstalk during this window doesn't result in a false detection of an ALIGN. For example: a compliant host may detect the deassertion of COMWAKE in as little as 112ns, such a host should wait at least 116.3ns (175+53.3-112) after detecting the release of COMWAKE to start looking for ALIGNs.<br>4. The Host PHY state machine may use the transition to HR_Reset as a method of speed negotiation. | | |

| HP7: HR_SendAlign | Transmit ALIGN at speed detected | |
|---|---|---|
| 1. Three back-to-back non-ALIGN primitives[2] detected from device | → | HR_Ready |
| 2. Three back-to-back non-ALIGN primitives not detected from device | → | HR_SendAlign[1] |
| NOTE :<br>1. Host retries indefinitely unless explicitly turned off by the application layer<br>2. Non-ALIGN primitives can be detected by the presence of the k28.3 control character in the byte0 position | | |

| HP8: HR_Ready | Transmit word from Link[1] | |
|---|---|---|
| 1. Partial signal from Link asserted | → | HR_Partial |
| 2. Slumber signal from Link asserted | → | HR_Slumber |
| 3. No power management request received | → | HR_Ready |
| NOTE :<br>1. PhyRdy asserted only when in the HR_Ready state and the Phy is maintaining synchronization with the incoming signal to its receiver and is transmitting a valid signal on its transmitter. | | |

| HP9: HR_Partial | Interface quiescent | |
|---|---|---|
| 1. Partial signal from Link deasserted and no COMWAKE detected from device[1] | → | HR_COMWAKE |
| 2. Partial signal from Link deasserted and COMWAKE detected from device[1] | → | HR_AwaitNoCOMWAKE |
| 3. Partial signal from Link asserted | → | HR_Partial |
| NOTE :<br>1. Host Phy must remember if COMWAKE was detected during Partial to determine if the wakeup request originated from the host or the Phy. | | |

| HP10: HR_Slumber | Interface quiescent | | |
|---|---|---|---|
| | 1. Slumber signal from Link deasserted and no COMWAKE detected from device[1,2] | → | HR_COMWAKE |
| | 2. Slumber signal from Link deasserted and COMWAKE detected from device[1,2] | → | HR_AwaitNoCOMWAKE |
| | 3. Slumber signal from Link asserted | → | HR_Slumber |
| | NOTE :<br>1. Host Phy must remember if COMWAKE was detected during Slumber to determine if the wakeup request originated from the host or the Phy.<br>2. The host Phy may take this transition only after it has recovered from slumber mode and the Phy is prepared to initiate communications. If Phy has not yet recovered from the slumber mode it shall remain in this state. | | |

| HP11: HR_AdjustSpeed | Interface undefined but not quiescent[1] | | |
|---|---|---|---|
| | 1 Transition to appropriate speed completed | → | HR_SendAlign |
| | 2 Transition to appropriate speed not completed | → | HR_AdjustSpeed |
| | NOTE :<br>1. Some implementations may undergo a transient condition where invalid signals are transmitted during the change in their internal transmission/reception speed. The host may transmit invalid signals for a period of up to 53ns (two Gen1 dwords) during the speed transition. Transmit jitter and unit interval timing requirements may not be met during this period but shall be met for all other bits transmitted in this state. A Phase shift may occur across the speed transition time. | | |

#### 6.8.1.1.2 Device Phy Initialization state machine

As described in section 6.7.4.2, reception of a COMRESET signal shall be treated by the device as a hard reset signal and shall unconditionally force the Device Phy state machine to transition to the DP1:DR_Reset initial state regardless of other conditions. Reception of COMRESET is effectively an additional transition into the DP1:DR_Reset state that appears in every Device Phy state. For the sake of brevity, this implied transition has been omitted from all the states.

| DP1: DR_Reset[1] | Interface quiescent | |
|---|---|---|
| 1. COMRESET not detected and power-on reset deasserted | → | DR_COMINIT |
| 2. COMRESET detected or power-on reset asserted | → | DR_Reset |
| NOTE :<br>1. This state is entered asynchronously any time in response to power-on reset or receipt of a COMRESET signal from the host | | |

| DP2: DR_COMINIT | Transmit COMINIT[1] | |
|---|---|---|
| 1. Unconditional | → | DR_AwaitCOMWAKE |
| NOTE :<br>1. COMINIT transmitted for a 6 bursts duration | | |

| DP3: DR_AwaitCOMWAKE | Interface quiescent | |
|---|---|---|
| 1. COMWAKE detected from host | → | DR_AwaitNoCOMWAKE |
| 2. COMWAKE not detected from host | → | DR_AwaitCOMWAKE |

| DP3B: DR_AwaitNoCOMWAKE | Interface quiescent | |
|---|---|---|
| 1. COMWAKE not detected from host and part of power-on reset sequence[1] | → | DR_Calibrate |
| 2. COMWAKE not detected from host and part of partial/slumber awake sequence[1] | → | DR_COMWAKE |
| 3. COMWAKE detected from host | → | DR_AwaitNoCOMWAKE |
| NOTE :<br>1. Device must remember if it was sent to partial or slumber mode for proper wakeup action. | | |

| DP4: DR_Calibrate | Perform calibration[1] | |
|---|---|---|
| 1. Calibration complete or bypass not implemented | → | DR_COMWAKE |
| 2. Calibration not complete | → | DR_Calibrate |
| NOTE :<br>1. Calibration is optional. If bypassed or not implemented, proceed directly to DR_COMWAKE. | | |

| DP5: DR_COMWAKE | Transmit COMWAKE | |
|---|---|---|
| 1. Unconditional | → | DR_SendAlign |

| DP6: DR_SendAlign | Transmit ALIGN[1,2,3,5] | |
|---|---|---|
| 1. ALIGN detected from host (device locked to incoming data)[4] | → | DR_Ready |
| 2. ALIGN not detected from host and ALIGN primitives transmitted for 54.6us (2048[5] Gen1 ALIGN primitives) at speed other than lowest[6] | → | DR_ReduceSpeed |
| 3. ALIGN not detected from host and ALIGN primitives transmitted for 54.6us (2048[5] Gen1 ALIGN primitives) at lowest speed[6] | → | DR_Error |
| 4. ALIGN not detected from host and ALIGN primitives transmitted for less than 54.6us (2048 Gen1 ALIGN primitives) | → | DR_SendAlign |

NOTE :
1. ALIGN should be sent at the devices fastest supported speed first
2. ALIGNS should be sent only at valid frequencies (if PLL not locked, send D10.2)
3. After COMWAKE is released as specified in the out of band signaling section, the device must ensure the interface is active (not quiescent).
4. Device designers should be aware that the host is allowed 533ns (20 Gen1 dwords) after detecting the deassertion of COMWAKE to start sending d10.2 characters. Until this occurs, the bus will be at an idle condition and may be susceptible to crosstalk from other devices. Care must be taken so that crosstalk during this window doesn't result in a false detection of an ALIGN. Devices may extend this timeout up to an additional 54.6us (2048 Gen1 dwords) (for a max total of 109.2us), as necessary to allow their receiver time to lock to the host ALIGN.
5. Device must not leave the bus idle more than 53.3ns (2 Gen1 dwords) longer than the required 175ns to deassert COMWAKE.
6. If this is part of a device-initiated recovery from the Slumber or Partial power management state, the device Phy should resume at the speed previously negotiated and should not reduce its speed in response to failure to establish communications. Upon failing to establish communications it should instead transition directly to the DR_Error state to initiate a re-try of the ComWake sequence.

| DP7: DR_Ready[1] | Transmit word from Link | |
|---|---|---|
| 1. Partial signal from Link asserted | → | DR_Partial |
| 2. Slumber signal from Link asserted | → | DR_Slumber |
| 3. No power management request received | → | DR_Ready |

NOTE :
1. PhyRdy asserted only when in the DR_Ready state and the Phy is maintaining synchronization with the incoming signal to its receiver and is transmitting a valid signal on its transmitter.

| DP8: DR_Partial | Interface quiescent | |
|---|---|---|
| 1. Partial signal from Link deasserted | → | DR_COMWAKE |
| 2. Partial signal from Link deasserted and COMWAKE detected from host | → | DR_AwaitNoCOMWAKE |
| 3. Partial signal from Link asserted | → | DR_Partial |

| DP9: DR_Slumber | Interface quiescent | |
|---|---|---|
| 1. Slumber signal from Link deasserted | → | DR_COMWAKE |
| 2. Slumber signal from Link deasserted and COMWAKE detected from host | → | DR_AwaitNoCOMWAKE |
| 3. Slumber signal from Link asserted | → | DR_Slumber |

| DP10: DR_ReduceSpeed | Interface quiescent | |
|---|---|---|
| 1. Transition to legacy (slower) speed complete | → | DR_SendAlign[1] |
| 2. Transition to legacy speed not complete | → | DR_ReduceSpeed |
| NOTE :<br>1. Transition to a new speed is defined as being complete when the device is accurately transmitting a valid signal within the defined signaling tolerances for that speed. | | |

| DP11: DR_Error | Interface quiescent | |
|---|---|---|
| 1. Error not due to failure to resume | → | DR_Error |
| 2. Resume from Slumber or Partial failed | → | DR_COMWAKE |

### 6.8.1.2 Power-on sequence timing diagram

The following timing diagrams and descriptions are provided for clarity and are informative. The state diagrams provided in section 6.8.1.1 comprise the normative behavior specification and is the ultimate reference.

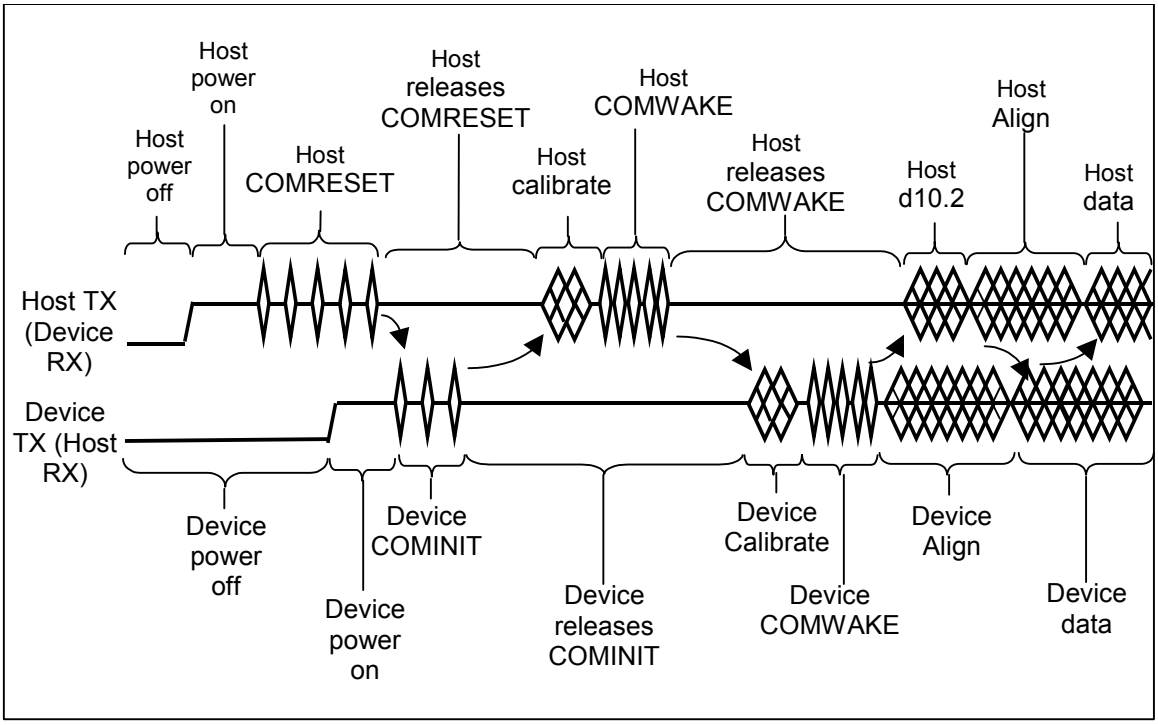



**Figure 47 – Power-on Sequence**

Description:

1. Host/device power-off - Host and device power-off. There is host side signal conditioning that will pull the host TX and RX pairs low if power is off.
2. Power is applied - Host side signal conditioning pulls TX and RX pairs to neutral state (common mode voltage).
3. Host issues COMRESET
4. Host releases COMRESET. Once the power-on reset is released, the host releases the COMRESET signal and puts the bus in a quiescent condition.
5. Device issues COMINIT – When the device detects the release of COMRESET, it responds with a COMINIT. This is also the entry point if the device is late starting. The device may initiate communications at any time by issuing a COMINIT.
6. Host calibrates and issues a COMWAKE.
7. Device responds – The device detects the COMWAKE signal on its RX pair and calibrates its transmitter (optional). Following calibration, the device sends a six burst COMWAKE signal and then sends a continuous stream of the ALIGN sequence. After ALIGN primitives have been sent for 54.6us (2048 Gen1 ALIGN Dwords) without a valid response from the host as determined by detection of received ALIGN primitives , the device may assume that the host cannot communicate at that speed. If additional legacy speeds are available, the device will try the next slower speed by sending ALIGN primitives for 54.6us at that rate. This step is repeated for as many legacy speeds are supported. Once the lowest speed has been reached without response from the host, the device will enter an error state.
8. Host locks – after detecting the COMWAKE, the host starts transmitting D10.2 characters (see 6.7.6) at its lowest supported rate. Meanwhile, the host receiver locks to the ALIGN sequence and, when ready, returns the ALIGN sequence to the device at the same speed as received. A host must be designed such that it can acquire lock within 54.6us (2048 Gen1 ALIGN Dwords). The host should wait for at least 880us (32768 Gen1 dwords) after detecting the release of COMWAKE to receive the first ALIGN. This will ensure interoperability with multi-generational and synchronous devices. If no ALIGN is received within 880us (32768 Gen1 dwords), the host shall restart the power-on sequence – repeating indefinitely until stopped by the application layer.
9. Device locks – the device locks to the ALIGN sequence and, when ready, sends the SYNC primitive indicating it is ready to start normal operation.
10. Upon receipt of three back-to-back non-ALIGN primitives, the communication link is established and normal operation may begin.

#### 6.8.1.3 Partial/Slumber to on

##### 6.8.1.3.1 Host initiated

The host can initiate a wakeup from the partial or slumber states by entering the power-on sequence at the “Host COMWAKE” point in the state machine. Calibration and speed negotiation is bypassed since it has already been performed at power-on and system performance depends on quick resume latency. The device, therefore, shall transmit ALIGNs at the speed determined at power-on.

##### 6.8.1.3.2 Device initiated

The device can initiate a wakeup from the partial or slumber states by entering the power-on sequence at the “Device COMWAKE” point in the state machine. Calibration and speed negotiation is bypassed since it has already been performed at power-on and system performance depends on quick resume latency. The device, therefore, shall transmit ALIGNs at the speed determined at power-on.

### 6.8.1.4 On to Partial/Slumber

#### 6.8.1.4.1 Host initiated

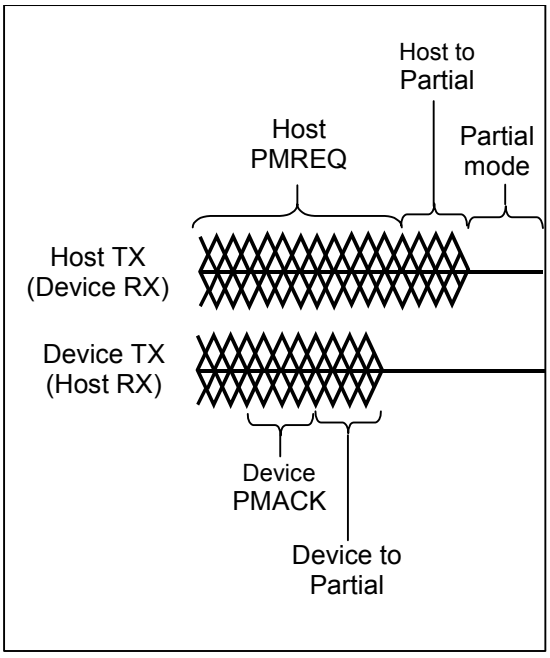



**Figure 48 – On to Partial/Slumber - host initiated**

##### 6.8.1.4.1.1 Detailed sequence

1. Host Application layer sends request to host Transport layer.
2. Host Transport layer transmits request to host Link layer.
3. Host Link layer encodes request as PMREQ primitive and transmits it four times to host Phy layer.
4. Host Phy layer serializes PMREQ primitives and transmits them to device Phy layer.
5. Device Phy de-serializes PMREQ primitives and transmits them to device Link layer.
6. Device Link layer decodes PMREQ primitives and transmits request to device Transport layer.
7. Device Transport layer transmits request to device Application layer.
8. Device Application layer processes and accepts request. Issues accept to device Transport layer.
9. Device Transport layer transmits acceptance to device Link layer.
10. Device Link layer encodes acceptance as PMACK primitive and transmits it four times to device Phy layer.
11. Device Phy layer transmits four PMACK primitives to host Phy layer.
12. Device Link layer places device Phy layer in Partial/Slumber state.
13. Host Phy layer de-serializes PMACK primitives and transmits them to host Link layer.
14. Host Link layer decodes PMACK primitives and transmits acceptance to host Transport layer.
15. Host Link layer places host Phy layer in Partial/Slumber State.
16. Host Transport layer transmits acceptance to host Application layer.

#### 6.8.1.4.2 Device initiated

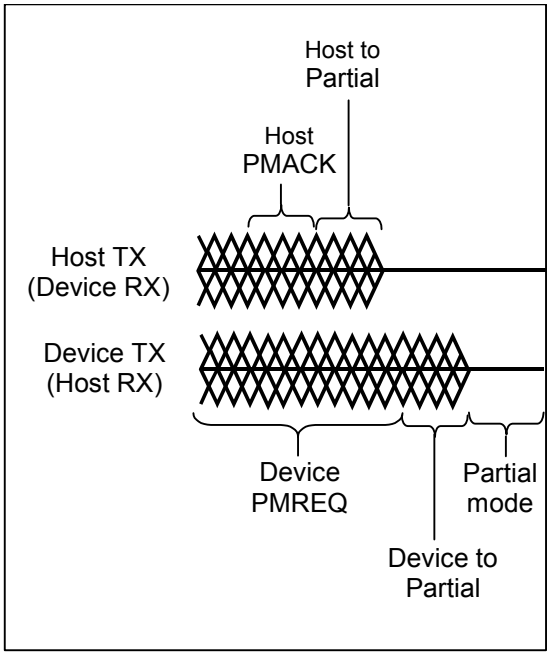



**Figure 49 – ON to Partial/Slumber - device initiated**

##### 6.8.1.4.2.1 Detailed sequence

1. Device Application layer sends request to device Transport layer.
2. Device Transport layer transmits request to device Link layer.
3. Device Link layer encodes request as PMREQ primitive and transmits it to device Phy layer.
4. Device Phy layer serializes PMREQ primitives and transmits them to host Phy layer.
5. Host Phy de-serializes PMREQ primitives and transmits them to host Link layer.
6. Host Link layer decodes PMREQ primitives and transmits request to host Transport layer.
7. Host Transport layer transmits request to host Application layer.
   NOTE – In this context, the host Application layer does not necessarily imply BIOS or other host CPU programming. Rather, the Application layer is the intelligent control section of the chipset logic.
8. Host Application layer processes and accepts request. Issues accept to host Transport layer.
9. Host Transport layer transmits acceptance to host Link layer.
10. Host link layer encodes acceptance as PMACK primitive and transmits it four times to host Phy layer.
11. Host Phy layer transmits four PMACK primitives to device Phy layer.
12. Host Link layer asserts Partial/Slumber signal and places host Phy layer in Partial/Slumber state.
13. Host Phy layer negates Ready signal.
14. Device Phy layer de-serializes PMACK primitives and transmits them to device Link layer.
15. Device Link layer decodes PMACK primitives and transmits acceptance to device Transport layer.
16. Device Link layer asserts Partial/Slumber signal and places device Phy layer in Partial/Slumber State.
17. Device Phy layer negates Ready signal.
18. Device Transport layer transmits acceptance to device Application layer.

# 7 Link layer

### 7.1.1 Overview

The Link layer transmits and receives frames, transmits primitives based on control signals from the Transport layer, and receives primitives from the Phy layer which are converted to control signals to the Transport layer. The Link layer need not be cognizant of the content of frames. Host and device Link layer state machines differ only in the fact that it is assumed that the host shall back off when transmission buffer is full.

#### 7.1.1.1 Frame transmission

When requested by the Transport layer to transmit a frame, the Link layer provides the following services:

- Negotiates with its peer Link layer to transmit a frame, resolves arbitration conflicts if both host and device request transmission
- Inserts frame envelope around Transport layer data (i.e., SOF, CRC, EOF, etc.).
- Receives data in the form of Dwords from the Transport layer.
- Calculates CRC on Transport layer data.
- Transmits frame.
- Provides frame flow control in response to requests from the FIFO or the peer Link layer.
- Receives frame receipt acknowledge from peer Link layer.
- Reports good transmission or Link/Phy layer errors to Transport layer.
- Performs 8b/10b encoding
- Transforms (scrambles) control and data Dwords in such a way to distribute the potential EMI emissions over a broader range

#### 7.1.1.2 Frame receipt

When data is received from the Phy layer, the Link layer provides the following services:

- Acknowledges to the peer Link layer readiness to receive a frame.
- Receives data in the form of encoded characters from the Phy layer.
- Decodes the encoded 8b/10b character stream into aligned Dwords of data.
- Removes the envelope around frames (i.e., SOF, CRC, EOF).
- Calculates CRC on the received Dwords.
- Provides frame flow control in response to requests from the FIFO or the peer Link layer.
- Compares the calculated CRC to the received CRC.
- Reports good reception or Link/Phy layer errors to Transport layer and the peer Link layer.
- Untransforms (descrambles) the control and data Dwords received from a peer Link layer.

## 7.2 Encoding method

Information to be transmitted over the Serial ATA bus shall be encoded a byte (eight bits) at a time along with a data or control character indicator into a 10-bit encoded character and then sent serially bit by bit. Information received over the Serial ATA bus shall be collected ten bits at a time, assembled into an encoded character, and decoded into the correct data characters and control characters. The 8b/10b code allows for the encoding of all 256 combinations of eight-bit data. A smaller subset of the control character set is utilized by Serial ATA.

### 7.2.1 Notation and conventions

Serial ATA uses a letter notation for describing data bits and control variables. A description of the translation process between these notations follows. This section also describes a convention used

to differentiate data characters from control characters. Finally, translation examples for both a data character and a control character are presented.

An unencoded byte of data is composed of eight bits A,B,C,D,E,F,G,H and the control variable Z. The encoding process results in a 10 bit character a,b,c,d,e,i,f,g,h,j. A bit is either a binary zero or binary one. The control variable, Z, has a value of D or K. When the control variable associated with a byte has the value D, the byte is referred to as a data character. When the control variable associated with a byte has the value K, the byte is referred to as a control character.

If a data byte is not accompanied with a specific control variable value the control variable Z is assumed to be Z = D and the data byte shall be encoded as a data character.

The following figure illustrates the association between the numbered unencoded bits in a byte, the control variable, and the letter-labeled bits in the encoding scheme:

| Data byte notation | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Control variable |
|---|---|---|---|---|---|---|---|---|---|
| Unencoded bit notation | H | G | F | E | D | C | B | A | Z |

**Figure 50 – Bit designations**

Each character is given a name Zxx.y where Z is the value of the control variable (D for a data character, K for a control character), xx is the decimal value of the binary number composed of the bits E, D, C, B and A in that order, and y is the decimal value of the binary number composed of the bits H, G and F.

Figure 51, following, shows the relationship between the various representations.

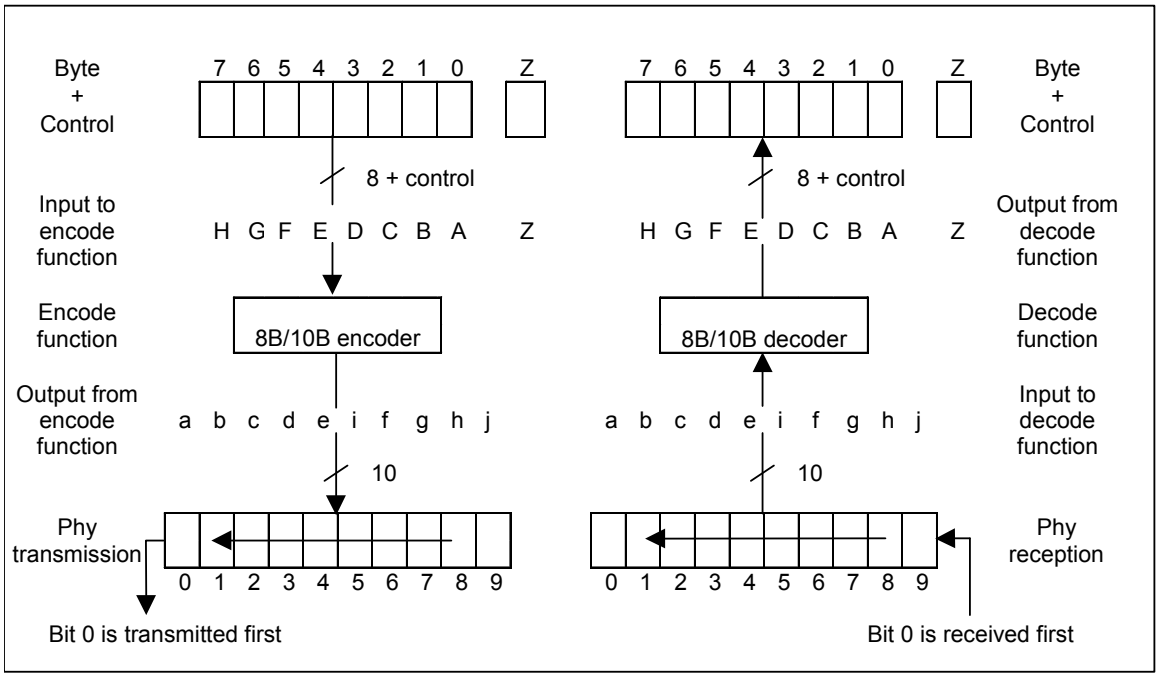



**Figure 51 – Nomenclature reference**

Figure 52 shows conversions from byte notation to character notation for a control and data byte. The examples chosen have special significance and will also be used during the conversion from data notation to the 8b/10b code values.

| Byte notation | BCh, control character | | 4Ah, data character | |
|---|---|---|---|---|
| Bit notation | 76543210<br>10111100 | Control variable<br>K | 76543210<br>01001010 | Control variable<br>D |
| Unencoded bit notation | HGF EDCBA<br>101 11100 | Z<br>K | HGF EDCBA<br>010 01010 | Z<br>D |
| Bit notation reordered to conform with Zxx.y convention | Z EDCBA HGF<br>K 11100 101 | | Z EDCBA HGF<br>D 01010 010 | |
| Character name | K 28 .5 | | D 10 .2 | |

**Figure 52 – Conversion examples**

### 7.2.2 Character code

The coding scheme used by Serial ATA translates unencoded data and control bytes to characters. The encoded characters are then transmitted by the physical layer over the serial line where they are received from the physical layer and decoded into the corresponding byte and control value.

Serial ATA uses a subset of the 8b/10b coding method described by Widmer and Franaszek. [1] [2] The Serial ATA code uses all 256 data byte encodings while only two of the control codes are used. The reception of any unused code is a class of reception error referred to as a code violation.

#### 7.2.2.1 Code construction

The 8b/10b coding process is defined in two stages. The first stage encodes the first five bits of the unencoded input byte into a six bit sub-block using a 5B/6B encoder. The input to this stage includes the current running disparity value. The second stage uses a 3B/4B encoder to encode the remaining three bits of the data byte and the running disparity as modified by the 5B/6B encoder into a four bit value.

In the derivations that follow, the control variable (Z) is assumed to have a value of D, and thus is an implicit input.

#### 7.2.2.2 The concept of running disparity

Running Disparity is a binary parameter with either the value negative (-) or the value positive (+).

After transmitting any encoded character, the transmitter shall calculate a new value for its Running Disparity based on the value of the transmitted character.

After a COMRESET, initial power-up, exiting any power management state, or exiting any diagnostic mode, the receiver shall assume either the positive or negative value for its initial Running Disparity. Upon reception of an encoded character the receiver shall determine whether the encoded character is valid according to the following rules and tables and shall calculate a new value for its Running Disparity based on the contents of the received character.

The following rules shall be used to calculate a new Running Disparity value for the transmitter after it sends an encoded character (transmitter's new Running Disparity) and for the receiver upon reception of an encoded character (receiver's new Running Disparity).

Running Disparity for an encoded character shall be calculated on two sub-blocks where the first six bits (abcdei) form one sub-block – the six-bit sub-block. The last four bits (fghj) form the second sub-block – the four-bit sub-block. Running Disparity at the beginning of the six-bit sub-block is the Running Disparity at the end of the last encoded character or the initial conditions described above for the first encoded character transmitted or received. Running Disparity at the beginning of the four-bit sub-block is the resulting Running Disparity from the six-bit sub-block. Running Disparity at the end of the encoded character – and the initial Running Disparity for the next encoded character – is the Running Disparity at the end of the four-bit sub-block.

Running Disparity for each of the sub-blocks shall be calculated as follows:

Running Disparity at the end of any sub-block is positive if the sub-block contains more ones than zeros. It is also positive at the end of the six-bit sub-block if the value of the six-bit sub-block is 000111, and is positive at the end of the four-bit sub-block if the value of the four-bit sub-block is 0011.

Running Disparity at the end of any sub-block is negative if the sub-block contains more zeros than ones. It is also negative at the end of the six-bit sub-block if the value of the six-bit sub-block is 111000, and is negative at the end of the four-bit sub-block if the value of the four-bit sub-block is 1100.

Otherwise, for any sub-block with an equal number of zeros and ones, the Running Disparity at the end of the sub-block is the same as at the beginning of the sub-block. Sub-blocks with an equal number of zeros and ones are said to have neutral disparity.

The 8b/10b code restricts the generation of the 000111, 111000, 0011 and 1100 sub-blocks in order to limit the run length of zeros and ones between sub-blocks. Sub-blocks containing 000111 or 0011 are generated only when the running disparity at the beginning of the sub-block is positive, resulting in positive Running Disparity at the end of the sub-block. Similarly, sub-blocks containing 111000 or 1100 are generated only when the running disparity at the beginning of the sub-block is negative and the resulting Running Disparity will also be negative.

The rules for Running Disparity will result in generation of a character with disparity that is either the opposite of the previous character or neutral.

Sub-blocks with non-zero (non-neutral) disparity must be of alternating disparity.

#### 7.2.2.3 Data encoding

Table 16 and Table 17 describe the code and running disparity generation rules for each of the sub-blocks. The results can be used to generate the data in the data character tables.

The digital logic which is used to generate the results can be found in Franaszek and Widmer [2]. The generation of control characters is also covered in the patent but not here.

In the tables which follow rd+ or rd- represent the current (incoming) running disparity and rd' represents the resulting Running Disparity. The resulting Running Disparity columns use –rd to indicate a change in Running Disparity polarity while rd indicates the resulting sub-block has neutral disparity.

**Table 16. – 5B/6B coding**

| Inputs | | abcdei outputs | | rd' | Inputs | | abcdei outputs | | rd' |
|---|---|---|---|---|---|---|---|---|---|
| Dx | EDCBA | rd+ | rd- | | Dx | EDCBA | rd+ | rd- | |
| D0 | 00000 | 011000 | 100111 | -rd | D16 | 10000 | 100100 | 011011 | -rd |
| D1 | 00001 | 100010 | 011101 | | D17 | 10001 | 100011 | | rd |
| D2 | 00010 | 010010 | 101101 | | D18 | 10010 | 010011 | | |
| D3 | 00011 | 110001 | | rd | D19 | 10011 | 110010 | | |
| D4 | 00100 | 001010 | 110101 | -rd | D20 | 10100 | 001011 | | |
| D5 | 00101 | 101001 | | | D21 | 10101 | 101010 | | |
| D6 | 00110 | 011001 | | rd | D22 | 10110 | 011010 | | |
| D7 | 00111 | 000111 | 111000 | | D23 | 10111 | 000101 | 111010 | -rd |
| D8 | 01000 | 000110 | 111001 | -rd | D24 | 11000 | 001100 | 110011 | |
| D9 | 01001 | 100101 | | | D25 | 11001 | 100110 | | rd |
| D10 | 01010 | 010101 | | | D26 | 11010 | 010110 | | |
| D11 | 01011 | 110100 | | rd | D27 | 11011 | 001001 | 110110 | -rd |
| D12 | 01100 | 001101 | | | D28 | 11100 | 001110 | | rd |
| D13 | 01101 | 101100 | | | D29 | 11101 | 010001 | 101110 | -rd |
| D14 | 01110 | 011100 | | | D30 | 11110 | 100001 | 011110 | |
| D15 | 01111 | 101000 | 010111 | -rd | D31 | 11111 | 010100 | 101011 | |

**Table 17. – 3B/4B coding**

| Inputs | | fghj outputs | | rd' |
|---|---|---|---|---|
| Dx.y | HGF | rd+ | rd- | |
| Dx.0 | 000 | 0100 | 1011 | -rd |
| Dx.1 | 001 | 1001 | | rd |
| Dx.2 | 010 | 0101 | | |
| Dx.3 | 011 | 0011 | 1100 | |
| Dx.4 | 100 | 0010 | 1101 | -rd |
| Dx.5 | 101 | 1010 | | rd |
| Dx.6 | 110 | 0110 | | |
| Dx.P7 | 111 | 0001 | 1110 | -rd |
| Dx.A7 | 111 | 1000 | 0111 | |

NOTE –
A7 replaces P7 if[(rd>0) and (e=i=0)] or [(rd<0) and (e=i=1)]

#### 7.2.2.4 Encoding examples

The coding examples in Figure 53 illustrate how the running disparity calculations are done.

The first conversion example completes the translation of data byte value 4Ah (which is the character name of D10.2) into an encoded character value of “abcdei fghj” = “010101 0101”. This value has special significance because (1) it is of neutral disparity, and also contains an alternating zero/one pattern that represents the highest data frequency which can be generated.

In the second example the 8b/10b character named D11.7 is encoded. Assuming a positive value for the incoming Running Disparity, this example shows the Dx.P7/Dx.A7 substitution. With an initial rd+ value, D10 translates to an abcdei value of 110100, with a resulting Running Disparity of positive for the 6-bit sub-block. Encoding the 4-bit sub-block will trigger the substitution clause of Dx.A7 for Dx.P7 since [(rd>0) AND (e=i=0)].

| Initial rd | Character name | abcdei output | 6-bit sub-block rd | fghj output | 4-bit sub-block rd | Encoded character | Ending rd |
|---|---|---|---|---|---|---|---|
| - | D10.2 | 010101 | - | 0101 | - | 010101 0101 | - |
| + | D11.7 | 110100 | + | 1000 | - | 110100 1000 | - |

**Figure 53 – Coding examples**

#### 7.2.2.5 8b/10b valid encoded characters

The following tables define the valid data characters and valid control characters. These tables shall be used for generating encoded characters (encoding) for transmission. In the reception process, the table is used to look up and verify the validity of received characters (decoding).

In the tables, each data character and control character has two columns that represent two encoded characters. One column represents the output if the current Running Disparity is negative and the other is the output if the current Running Disparity is positive.

#### 7.2.2.5.1 Data characters

**Table 18. – Valid data characters**

| Name | Byte | abcdei fghj output | | Name | Byte | abcdei fghj output | |
|---|---|---|---|---|---|---|---|
| | | Current rd- | Current rd+ | | | Current rd- | Current rd+ |
| D0.0 | 00h | 100111 0100 | 011000 1011 | D0.1 | 20h | 100111 1001 | 011000 1001 |
| D1.0 | 01h | 011101 0100 | 100010 1011 | D1.1 | 21h | 011101 1001 | 100010 1001 |
| D2.0 | 02h | 101101 0100 | 010010 1011 | D2.1 | 22h | 101101 1001 | 010010 1001 |
| D3.0 | 03h | 110001 1011 | 110001 0100 | D3.1 | 23h | 110001 1001 | 110001 1001 |
| D4.0 | 04h | 110101 0100 | 001010 1011 | D4.1 | 24h | 110101 1001 | 001010 1001 |
| D5.0 | 05h | 101001 1011 | 101001 0100 | D5.1 | 25h | 101001 1001 | 101001 1001 |
| D6.0 | 06h | 011001 1011 | 011001 0100 | D6.1 | 26h | 011001 1001 | 011001 1001 |
| D7.0 | 07h | 111000 1011 | 000111 0100 | D7.1 | 27h | 111000 1001 | 000111 1001 |
| D8.0 | 08h | 111001 0100 | 000110 1011 | D8.1 | 28h | 111001 1001 | 000110 1001 |
| D9.0 | 09h | 100101 1011 | 100101 0100 | D9.1 | 29h | 100101 1001 | 100101 1001 |
| D10.0 | 0Ah | 010101 1011 | 010101 0100 | D10.1 | 2Ah | 010101 1001 | 010101 1001 |
| D11.0 | 0Bh | 110100 1011 | 110100 0100 | D11.1 | 2Bh | 110100 1001 | 110100 1001 |
| D12.0 | 0Ch | 001101 1011 | 001101 0100 | D12.1 | 2Ch | 001101 1001 | 001101 1001 |
| D13.0 | 0Dh | 101100 1011 | 101100 0100 | D13.1 | 2Dh | 101100 1001 | 101100 1001 |
| D14.0 | 0Eh | 011100 1011 | 011100 0100 | D14.1 | 2Eh | 011100 1001 | 011100 1001 |
| D15.0 | 0Fh | 010111 0100 | 101000 1011 | D15.1 | 2Fh | 010111 1001 | 101000 1001 |
| D16.0 | 10h | 011011 0100 | 100100 1011 | D16.1 | 30h | 011011 1001 | 100100 1001 |
| D17.0 | 11h | 100011 1011 | 100011 0100 | D17.1 | 31h | 100011 1001 | 100011 1001 |
| D18.0 | 12h | 010011 1011 | 010011 0100 | D18.1 | 32h | 010011 1001 | 010011 1001 |
| D19.0 | 13h | 110010 1011 | 110010 0100 | D19.1 | 33h | 110010 1001 | 110010 1001 |
| D20.0 | 14h | 001011 1011 | 001011 0100 | D20.1 | 34h | 001011 1001 | 001011 1001 |
| D21.0 | 15h | 101010 1011 | 101010 0100 | D21.1 | 35h | 101010 1001 | 101010 1001 |
| D22.0 | 16h | 011010 1011 | 011010 0100 | D22.1 | 36h | 011010 1001 | 011010 1001 |
| D23.0 | 17h | 111010 0100 | 000101 1011 | D23.1 | 37h | 111010 1001 | 000101 1001 |
| D24.0 | 18h | 110011 0100 | 001100 1011 | D24.1 | 38h | 110011 1001 | 001100 1001 |
| D25.0 | 19h | 100110 1011 | 100110 0100 | D25.1 | 39h | 100110 1001 | 100110 1001 |
| D26.0 | 1Ah | 010110 1011 | 010110 0100 | D26.1 | 3Ah | 010110 1001 | 010110 1001 |
| D27.0 | 1Bh | 110110 0100 | 001001 1011 | D27.1 | 3Bh | 110110 1001 | 001001 1001 |
| D28.0 | 1Ch | 001110 1011 | 001110 0100 | D28.1 | 3Ch | 001110 1001 | 001110 1001 |
| D29.0 | 1Dh | 101110 0100 | 010001 1011 | D29.1 | 3Dh | 101110 1001 | 010001 1001 |
| D30.0 | 1Eh | 011110 0100 | 100001 1011 | D30.1 | 3Eh | 011110 1001 | 100001 1001 |
| D31.0 | 1Fh | 101011 0100 | 010100 1011 | D31.1 | 3Fh | 101011 1001 | 010100 1001 |

*(continued)*

**Table 18 – Valid data characters** *(continued)*

| Name | Byte | abcdei fghj output | | Name | Byte | abcdei fghj output | |
|---|---|---|---|---|---|---|---|
| | | Current rd- | Current rd+ | | | Current rd- | Current rd+ |
| D0.2 | 40h | 100111 0101 | 011000 0101 | D0.3 | 60h | 100111 0011 | 011000 1100 |
| D1.2 | 41h | 011101 0101 | 100010 0101 | D1.3 | 61h | 011101 0011 | 100010 1100 |
| D2.2 | 42h | 101101 0101 | 010010 0101 | D2.3 | 62h | 101101 0011 | 010010 1100 |
| D3.2 | 43h | 110001 0101 | 110001 0101 | D3.3 | 63h | 110001 1100 | 110001 0011 |
| D4.2 | 44h | 110101 0101 | 001010 0101 | D4.3 | 64h | 110101 0011 | 001010 1100 |
| D5.2 | 45h | 101001 0101 | 101001 0101 | D5.3 | 65h | 101001 1100 | 101001 0011 |
| D6.2 | 46h | 011001 0101 | 011001 0101 | D6.3 | 66h | 011001 1100 | 011001 0011 |
| D7.2 | 47h | 111000 0101 | 000111 0101 | D7.3 | 67h | 111000 1100 | 000111 0011 |
| D8.2 | 48h | 111001 0101 | 000110 0101 | D8.3 | 68h | 111001 0011 | 000110 1100 |
| D9.2 | 49h | 100101 0101 | 100101 0101 | D9.3 | 69h | 100101 1100 | 100101 0011 |
| D10.2 | 4Ah | 010101 0101 | 010101 0101 | D10.3 | 6Ah | 010101 1100 | 010101 0011 |
| D11.2 | 4Bh | 110100 0101 | 110100 0101 | D11.3 | 6Bh | 110100 1100 | 110100 0011 |
| D12.2 | 4Ch | 001101 0101 | 001101 0101 | D12.3 | 6Ch | 001101 1100 | 001101 0011 |
| D13.2 | 4Dh | 101100 0101 | 101100 0101 | D13.3 | 6Dh | 101100 1100 | 101100 0011 |
| D14.2 | 4Eh | 011100 0101 | 011100 0101 | D14.3 | 6Eh | 011100 1100 | 011100 0011 |
| D15.2 | 4Fh | 010111 0101 | 101000 0101 | D15.3 | 6Fh | 010111 0011 | 101000 1100 |
| D16.2 | 50h | 011011 0101 | 100100 0101 | D16.3 | 70h | 011011 0011 | 100100 1100 |
| D17.2 | 51h | 100011 0101 | 100011 0101 | D17.3 | 71h | 100011 1100 | 100011 0011 |
| D18.2 | 52h | 010011 0101 | 010011 0101 | D18.3 | 72h | 010011 1100 | 010011 0011 |
| D19.2 | 53h | 110010 0101 | 110010 0101 | D19.3 | 73h | 110010 1100 | 110010 0011 |
| D20.2 | 54h | 001011 0101 | 001011 0101 | D20.3 | 74h | 001011 1100 | 001011 0011 |
| D21.2 | 55h | 101010 0101 | 101010 0101 | D21.3 | 75h | 101010 1100 | 101010 0011 |
| D22.2 | 56h | 011010 0101 | 011010 0101 | D22.3 | 76h | 011010 1100 | 011010 0011 |
| D23.2 | 57h | 111010 0101 | 000101 0101 | D23.3 | 77h | 111010 0011 | 000101 1100 |
| D24.2 | 58h | 110011 0101 | 001100 0101 | D24.3 | 78h | 110011 0011 | 001100 1100 |
| D25.2 | 59h | 100110 0101 | 100110 0101 | D25.3 | 79h | 100110 1100 | 100110 0011 |
| D26.2 | 5Ah | 010110 0101 | 010110 0101 | D26.3 | 7Ah | 010110 1100 | 010110 0011 |
| D27.2 | 5Bh | 110110 0101 | 001001 0101 | D27.3 | 7Bh | 110110 0011 | 001001 1100 |
| D28.2 | 5Ch | 001110 0101 | 001110 0101 | D28.3 | 7Ch | 001110 1100 | 001110 0011 |
| D29.2 | 5Dh | 101110 0101 | 010001 0101 | D29.3 | 7Dh | 101110 0011 | 010001 1100 |
| D30.2 | 5Eh | 011110 0101 | 100001 0101 | D30.3 | 7Eh | 011110 0011 | 100001 1100 |
| D31.2 | 5Fh | 101011 0101 | 010100 0101 | D31.3 | 7Fh | 101011 0011 | 010100 1100 |

*(continued)*

**Table 18 – Valid data characters** *(continued)*

| Name | Byte | abcdei fghj output | | Name | Byte | abcdei fghj output | |
|---|---|---|---|---|---|---|---|
| | | Current rd- | Current rd+ | | | Current rd- | Current rd+ |
| D0.4 | 80h | 100111 0010 | 011000 1101 | D0.5 | A0h | 100111 1010 | 011000 1010 |
| D1.4 | 81h | 011101 0010 | 100010 1101 | D1.5 | A1h | 011101 1010 | 100010 1010 |
| D2.4 | 82h | 101101 0010 | 010010 1101 | D2.5 | A2h | 101101 1010 | 010010 1010 |
| D3.4 | 83h | 110001 1101 | 110001 0010 | D3.5 | A3h | 110001 1010 | 110001 1010 |
| D4.4 | 84h | 110101 0010 | 001010 1101 | D4.5 | A4h | 110101 1010 | 001010 1010 |
| D5.4 | 85h | 101001 1101 | 101001 0010 | D5.5 | A5h | 101001 1010 | 101001 1010 |
| D6.4 | 86h | 011001 1101 | 011001 0010 | D6.5 | A6h | 011001 1010 | 011001 1010 |
| D7.4 | 87h | 111000 1101 | 000111 0010 | D7.5 | A7h | 111000 1010 | 000111 1010 |
| D8.4 | 88h | 111001 0010 | 000110 1101 | D8.5 | A8h | 111001 1010 | 000110 1010 |
| D9.4 | 89h | 100101 1101 | 100101 0010 | D9.5 | A9h | 100101 1010 | 100101 1010 |
| D10.4 | 8Ah | 010101 1101 | 010101 0010 | D10.5 | AAh | 010101 1010 | 010101 1010 |
| D11.4 | 8Bh | 110100 1101 | 110100 0010 | D11.5 | ABh | 110100 1010 | 110100 1010 |
| D12.4 | 8Ch | 001101 1101 | 001101 0010 | D12.5 | ACh | 001101 1010 | 001101 1010 |
| D13.4 | 8Dh | 101100 1101 | 101100 0010 | D13.5 | ADh | 101100 1010 | 101100 1010 |
| D14.4 | 8Eh | 011100 1101 | 011100 0010 | D14.5 | AEh | 011100 1010 | 011100 1010 |
| D15.4 | 8Fh | 010111 0010 | 101000 1101 | D15.5 | AFh | 010111 1010 | 101000 1010 |
| D16.4 | 90h | 011011 0010 | 100100 1101 | D16.5 | B0h | 011011 1010 | 100100 1010 |
| D17.4 | 91h | 100011 1101 | 100011 0010 | D17.5 | B1h | 100011 1010 | 100011 1010 |
| D18.4 | 92h | 010011 1101 | 010011 0010 | D18.5 | B2h | 010011 1010 | 010011 1010 |
| D19.4 | 93h | 110010 1101 | 110010 0010 | D19.5 | B3h | 110010 1010 | 110010 1010 |
| D20.4 | 94h | 001011 1101 | 001011 0010 | D20.5 | B4h | 001011 1010 | 001011 1010 |
| D21.4 | 95h | 101010 1101 | 101010 0010 | D21.5 | B5h | 101010 1010 | 101010 1010 |
| D22.4 | 96h | 011010 1101 | 011010 0010 | D22.5 | B6h | 011010 1010 | 011010 1010 |
| D23.4 | 97h | 111010 0010 | 000101 1101 | D23.5 | B7h | 111010 1010 | 000101 1010 |
| D24.4 | 98h | 110011 0010 | 001100 1101 | D24.5 | B8h | 110011 1010 | 001100 1010 |
| D25.4 | 99h | 100110 1101 | 100110 0010 | D25.5 | B9h | 100110 1010 | 100110 1010 |
| D26.4 | 9Ah | 010110 1101 | 010110 0010 | D26.5 | BAh | 010110 1010 | 010110 1010 |
| D27.4 | 9Bh | 110110 0010 | 001001 1101 | D27.5 | BBh | 110110 1010 | 001001 1010 |
| D28.4 | 9Ch | 001110 1101 | 001110 0010 | D28.5 | BCh | 001110 1010 | 001110 1010 |
| D29.4 | 9Dh | 101110 0010 | 010001 1101 | D29.5 | BDh | 101110 1010 | 010001 1010 |
| D30.4 | 9Eh | 011110 0010 | 100001 1101 | D30.5 | BEh | 011110 1010 | 100001 1010 |
| D31.4 | 9Fh | 101011 0010 | 010100 1101 | D31.5 | BFh | 101011 1010 | 010100 1010 |

*(continued)*

**Table 18 – Valid data characters** *(continued)*

| Name | Byte | abcdei fghj output | | Name | Byte | abcdei fghj output | |
|---|---|---|---|---|---|---|---|
| | | Current rd- | Current rd+ | | | Current rd- | Current rd+ |
| D0.6 | C0h | 100111 0110 | 011000 0110 | D0.7 | E0h | 100111 0001 | 011000 1110 |
| D1.6 | C1h | 011101 0110 | 100010 0110 | D1.7 | E1h | 011101 0001 | 100010 1110 |
| D2.6 | C2h | 101101 0110 | 010010 0110 | D2.7 | E2h | 101101 0001 | 010010 1110 |
| D3.6 | C3h | 110001 0110 | 110001 0110 | D3.7 | E3h | 110001 1110 | 110001 0001 |
| D4.6 | C4h | 110101 0110 | 001010 0110 | D4.7 | E4h | 110101 0001 | 001010 1110 |
| D5.6 | C5h | 101001 0110 | 101001 0110 | D5.7 | E5h | 101001 1110 | 101001 0001 |
| D6.6 | C6h | 011001 0110 | 011001 0110 | D6.7 | E6h | 011001 1110 | 011001 0001 |
| D7.6 | C7h | 111000 0110 | 000111 0110 | D7.7 | E7h | 111000 1110 | 000111 0001 |
| D8.6 | C8h | 111001 0110 | 000110 0110 | D8.7 | E8h | 111001 0001 | 000110 1110 |
| D9.6 | C9h | 100101 0110 | 100101 0110 | D9.7 | E9h | 100101 1110 | 100101 0001 |
| D10.6 | CAh | 010101 0110 | 010101 0110 | D10.7 | EAh | 010101 1110 | 010101 0001 |
| D11.6 | CBh | 110100 0110 | 110100 0110 | D11.7 | EBh | 110100 1110 | 110100 1000 |
| D12.6 | CCh | 001101 0110 | 001101 0110 | D12.7 | ECh | 001101 1110 | 001101 0001 |
| D13.6 | CDh | 101100 0110 | 101100 0110 | D13.7 | EDh | 101100 1110 | 101100 1000 |
| D14.6 | CEh | 011100 0110 | 011100 0110 | D14.7 | EEh | 011100 1110 | 011100 1000 |
| D15.6 | CFh | 010111 0110 | 101000 0110 | D15.7 | EFh | 010111 0001 | 101000 1110 |
| D16.6 | D0h | 011011 0110 | 100100 0110 | D16.7 | F0h | 011011 0001 | 100100 1110 |
| D17.6 | D1h | 100011 0110 | 100011 0110 | D17.7 | F1h | 100011 0111 | 100011 0001 |
| D18.6 | D2h | 010011 0110 | 010011 0110 | D18.7 | F2h | 010011 0111 | 010011 0001 |
| D19.6 | D3h | 110010 0110 | 110010 0110 | D19.7 | F3h | 110010 1110 | 110010 0001 |
| D20.6 | D4h | 001011 0110 | 001011 0110 | D20.7 | F4h | 001011 0111 | 001011 0001 |
| D21.6 | D5h | 101010 0110 | 101010 0110 | D21.7 | F5h | 101010 1110 | 101010 0001 |
| D22.6 | D6h | 011010 0110 | 011010 0110 | D22.7 | F6h | 011010 1110 | 011010 0001 |
| D23.6 | D7h | 111010 0110 | 000101 0110 | D23.7 | F7h | 111010 0001 | 000101 1110 |
| D24.6 | D8h | 110011 0110 | 001100 0110 | D24.7 | F8h | 110011 0001 | 001100 1110 |
| D25.6 | D9h | 100110 0110 | 100110 0110 | D25.7 | F9h | 100110 1110 | 100110 0001 |
| D26.6 | DAh | 010110 0110 | 010110 0110 | D26.7 | FAh | 010110 1110 | 010110 0001 |
| D27.6 | DBh | 110110 0110 | 001001 0110 | D27.7 | FBh | 110110 0001 | 001001 1110 |
| D28.6 | DCh | 001110 0110 | 001110 0110 | D28.7 | FCh | 001110 1110 | 001110 0001 |
| D29.6 | DDh | 101110 0110 | 010001 0110 | D29.7 | FDh | 101110 0001 | 010001 1110 |
| D30.6 | DEh | 011110 0110 | 100001 0110 | D30.7 | FEh | 011110 0001 | 100001 1110 |
| D31.6 | DFh | 101011 0110 | 010100 0110 | D31.7 | FFh | 101011 0001 | 010100 1110 |

*(concluded)*

#### 7.2.2.5.2 Control characters

**Table 19. – Valid control characters**

| Name | abcdei fghj output | | Description |
|---|---|---|---|
| | Current rd- | Current rd+ | |
| K28.3 | 001111 0011 | 110000 1100 | Occurs only at byte 0 of all primitives except for the ALIGN primitive |
| K28.5 | 001111 1010 | 110000 0101 | Occurs only at byte 0 of the ALIGN primitive |

In Serial ATA only the K28.3 and K28.5 control characters are valid and are always used as the first byte in a four-byte primitive. The K28.3 control character is used to prefix all primitives other than the ALIGN primitive, while the K28.5 control character is used to prefix the ALIGN primitive. The encoding of characters within primitives follow the same rules as that applied to non-primitives, when calculating the running disparity between characters and between subblocks of each character within the primitive. The control characters K28.3 and K28.5 invert the current running disparity.
The running disparity at the end of the ALIGN primitive is the same as the running disparity at the beginning of the ALIGN primitive.

### 7.2.3 Transmission summary

#### 7.2.3.1 Transmission order

##### 7.2.3.1.1 Bits within a byte

The bits within an encoded character are labeled a,b,c,d,e,i,f,g,h,j. Bit “a” shall be transmitted first, followed in order by “b”, “c”, “d”, “e”, “i”, “f”, “g”, “h” and “j”. Note that bit “i” is transmitted between bits “e” and “f”, and that bit “j” is transmitted last, and not in the order that would be indicated by the letters of the alphabet. Figure 54 illustrates the transmission order within a byte

#### 7.2.3.1.2 Bytes within a Dword

For all transmissions and receptions, Serial ATA organizes all values as Dwords. Even when representing a 32-bit value, the Dword shall be considered a set of four bytes. The transmission order of the bytes within the Dword shall be from the least-significant byte (byte 0) to the most-significant byte (byte 3). This right-to-left transmission order differs from Fibre Channel. Figure 54 illustrates how the bytes are arranged in a Dword and the order in which bits are sent.

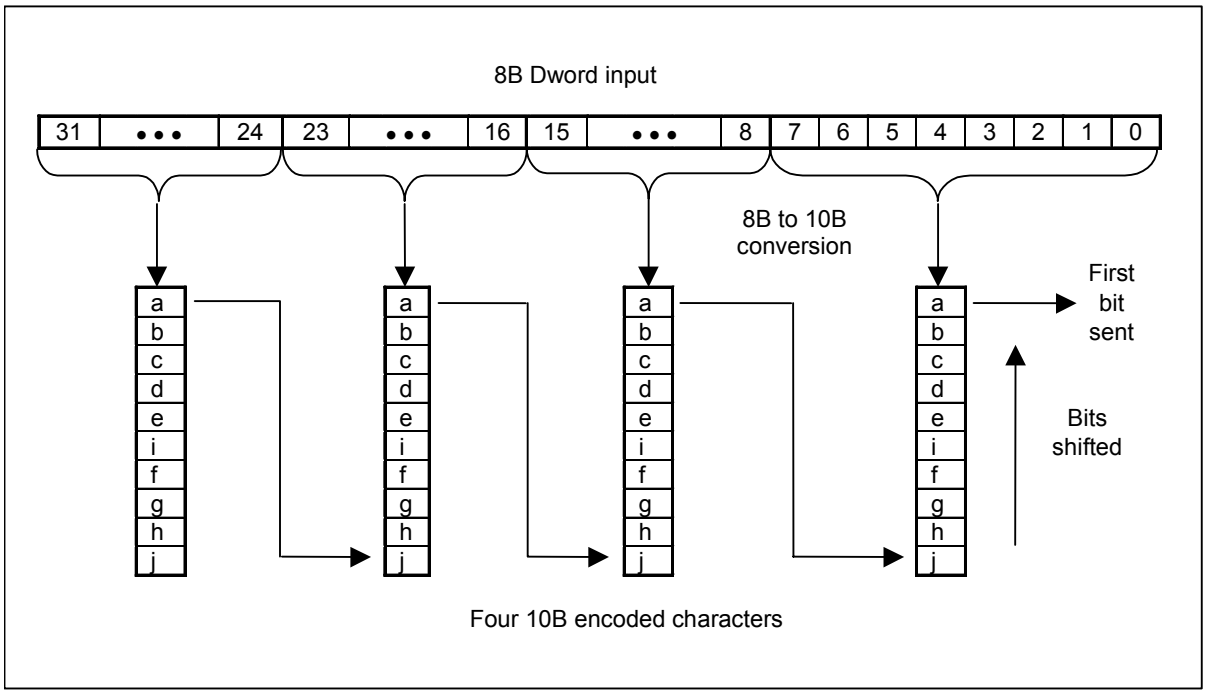



**Figure 54 – Bit ordering and significance**

#### 7.2.3.1.3 Dwords within a frame

A frame (as described in Section 8.3) shall be transmitted sequentially in ascending Dword order starting with the SOF delimiter, followed by the Dwords of the frame contents, followed by the CRC, and ending with the EOF delimiter.

NOTE – While this Serial ATA standard discusses a strict hierarchy of Dword transmission as an ordered series of bytes, it is not the intent to restrict implementations from implementing a wider data path. It is possible, and even desirable, to perform transmission in word-sized fields. 8b/10b encoders with a 16(unencoded)/20(encoded) data path do exist. The only restriction is the transmission order of each byte and running disparity for each sub-block is preserved.

### 7.2.4 Reception summary

Upon reception of an encoded character the column corresponding to the receiver's current Running Disparity shall be searched for the encoded character value. If the encoded character value is found in the table the received encoded character shall be considered a legal character and decoded, and the decoded character value is made available to the Link layer.

If the received encoded character is not found in that column, then the encoded character shall be marked as code violation and reported to the Link layer.

#### 7.2.4.1 Disparity and the detection of a code violation

Due to the propagation characteristics of the 8b/10b code, it is possible that although most errors will be detected, a single bit error might not be detected until several characters after the error is introduced. The following examples illustrate this effect. The first example shows a bit error being propagated two characters before being detected. The second shows a single character of propagation.

It is important to note that Serial ATA sends data in Dword increments, but the transmitter and receiver operate in units of a byte (character). The examples don't show Dword boundaries, so it is possible that an error in either of these cases could be deferred one full Dword.

The frequency of disparity errors and code violations is an indicator of channel quality and corresponds directly to the bit error rate of the physical serial link between a host and device. Implementations may elect to count such events and make them available to external firmware or software, although the method by which such counters are exposed is not defined in this specification.

Initial Running Disparity and the Running Disparity for each character is shown. In order to discover the errors note that Running Disparity is actually computed at the end of each sub-block and subsequently forwarded to the next sub-block. Footnotes indicate where the disparity error is detected. The error bit is underlined.

**Figure 55 – Single bit error with two character delay**

| | rd | Character | rd | Character | rd | Character | rd |
|---|---|---|---|---|---|---|---|
| Transmitted character stream | - | D21.1 | - | D10.2 | - | D23.5 | + |
| Transmitted bit stream | - | 101010 1001 | - | 010101 0101 | - | 111010 1010 | + |
| Received bit stream | - | 101010 1011[a] | + | 010101 0101 | + | 111010[b] 1010 | + |
| Decoded character stream | - | D21.0 | + | D10.2 | + | Code violation[b] | +[c] |

NOTES –
[a] Bit error introduced: 1001 → 1011
[b] Sub-blocks with non-neutral disparity must alternate polarity (i.e., + → -). In this case, rd does not alternate (it stays positive for two sub-blocks in a row). The resulting encoded character does not exist in the rd+ column in the data or control code table, and so an invalid encoded character is recognized.
[c] Running disparity must be computed on the received character regardless of the validity of the encoded character.

**Figure 56 – Single bit error with one character delay**

| | rd | Character | rd | Character | rd | Character | rd |
|---|---|---|---|---|---|---|---|
| Transmitted character stream | - | D21.1 | - | D23.4 | - | D23.5 | + |
| Transmitted bit stream | - | 101010 1001 | - | 111010 0010 | - | 111010 1010 | + |
| Received bit stream | - | 101010 1011[a] | + | 111010[b] 0010 | - | 111010 1010 | + |
| Decoded character stream | - | D21.0 | + | Code violation[b] | - | D23.5 | +[c] |

NOTES–
[a] Bit error introduced: 1001 => 1011
[b] Sub-blocks with non-neutral disparity must alternate polarity (i.e., + → -). In this case, rd does not alternate (it stays positive for two sub-blocks in a row). The resulting encoded character does not exist in the rd+ column in the data or control code table, and so an invalid encoded character is recognized.
[c] Running disparity must be computed on the received character regardless of the validity of the encoded character.

## 7.3 Transmission overview

The information on the serial line is a sequence of 8b/10b encoded characters. The smallest unit of communication is a Dword. The contents of each Dword are grouped to provide low-level control information or to transfer information between a host and an attached device.

The two types of structures are primitives and frames.

A primitive consists of a single Dword and is the simplest unit of information that may be exchanged between a host and a device. When the bytes of a primitive are encoded the resulting pattern is difficult to misinterpret as any other primitive or random pattern. Primitives are used primarily to convey real-time state information, to control the transfer of information and coordinate host / device communication. All bytes in a primitive are constants and the first byte is always a special character.

Since all of the bytes are constants, a primitive cannot be used to convey variable information. Later sections will describe the exact contents of the primitives used by Serial ATA.

A frame consists of multiple Dwords, and always starts with an SOF primitive, followed by a user payload called a Frame Information Structure (FIS), a CRC, and ends with an EOF primitive. The CRC is defined to be the last non-primitive Dword immediately preceding the EOF primitive. Some number of flow control primitives (HOLD or HOLDA, or a CONT stream to sustain a HOLD or HOLDA state) are allowed between the SOF and EOF primitives to throttle data flow for speed matching purposes.

The following figure shows an example of a sequence of transmissions.

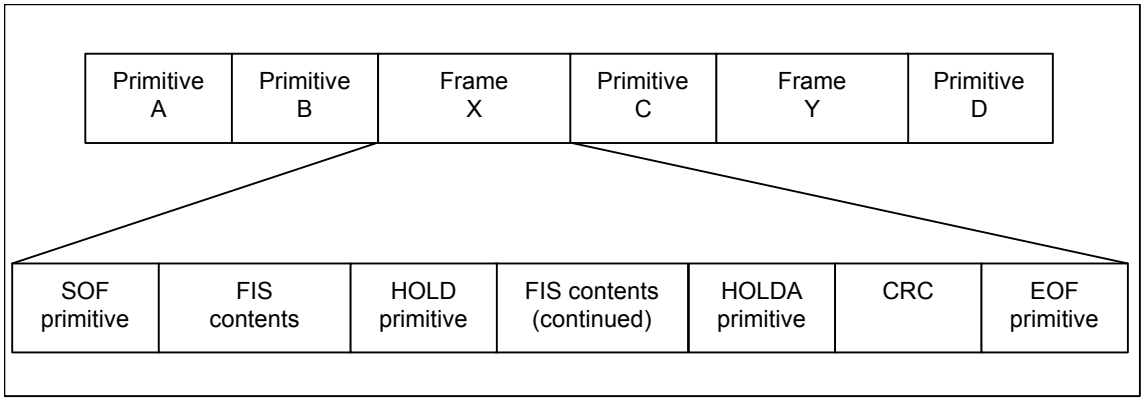



**Figure 57 – Transmission structures**

## 7.4 Primitives

### 7.4.1 Overview

Primitives are Dword entities that are used to control and provide status of the serial line.

Primitives always begin with a control character; all primitives use the K28.3 control character to signify the beginning of a primitive except for the ALIGN primitive which begins with the K28.5 control character. ALIGN thus represents the only primitive that contains the comma character. Following the control character, three additional characters are encoded to complete the Dword. Table 20 is a summary of the character combinations that make up each primitive.

#### 7.4.1.1 Primitive disparity

Primitives can begin with either positive or negative disparity and end in either positive or negative disparity Normal 8b/10b encoding disparity rules are applied when encoding primitives.

The ALIGN primitive is chosen to have neutral disparity so that it can be inserted into the stream without affecting the disparity of previously encoded characters. Disparity at the end of ALIGN is the same as the ending disparity of the last character transmitted before the ALIGN primitive.

Each primitive is described and the encoding defined in the following sections.

#### 7.4.1.2 Primitive handshakes

Some primitives are transmitted in response to receipt of other primitives to acknowledge receipt. For example, the HOLDA primitive is transmitted in response to the receipt of HOLD primitives and R_OK or R_ERR is transmitted in response to WTRM primitives. Due to the different clock domains between to two ends of the cable, the number of response primitives may not match the number of primitives to which they are responding. For example, a device may send five HOLD primitives but receive six HOLDA primitives in response. Neither the transmitter nor receiver of these primitives need count the number of primitives or match the number sent and received. There are boundary cases were a zero number of response primitives such as HOLDA may be sent.

### 7.4.2 Primitive descriptions

The following table contains the primitive mnemonics and a brief description of each.

**Table 20. – Description of primitives**

| Primitive | Name | Description |
|---|---|---|
| ALIGN | Physical layer control | Upon receipt of an ALIGN, the physical layer readjusts internal operations as necessary to perform its functions correctly. This primitive is always sent in pairs - there is no condition where an odd number of ALIGN primitives shall be sent (except as noted for retimed loopback). |
| CONT | Continue repeating previous primitive | The CONT primitive allows long strings of repeated primitives to be eliminated. The CONT primitive implies that the previously received primitive be repeated as long as another primitive is not received. |
| DMAT | DMA terminate | This primitive is sent as a request to the transmitter to terminate a DMA data transmission early by computing a CRC on the data sent and ending with a EOF primitive. The transmitter context is assumed to remain stable after the EOF primitive has been sent. |
| EOF | End of frame | EOF marks the end of a frame. The previous non-primitive Dword is the CRC for the frame. |
| HOLD | Hold data transmission | HOLD is transmitted in place of payload data within a frame when the transmitter does not have the next payload data ready for transmission. HOLD is also transmitted on the backchannel when a receiver is not ready to receive additional payload data. |
| HOLDA | Hold acknowledge | This primitive is sent by a transmitter as long the HOLD primitive is received by its companion receiver. |
| PMACK | Power management acknowledge | Sent in response to a PMREQ_S or PMREQ_P when a receiving node is prepared to enter a power mode state. |
| PMNAK | Power management denial | Sent in response to a PMREQ_S or PMREQ_P when a receiving node is not prepared to enter a power mode state or when power management is not supported. |
| PMREQ_P | Power management request to partial | This primitive is sent continuously until PMACK or PMNAK is received. When PMACK is received, current node (host or device) will stop PMREQ_P and enters the Partial power management state. |
| PMREQ_S | Power management request to Slumber | This primitive is sent continuously until PMACK or PMNAK is received. When PMACK is received, current node (host or device) will stop PMREQ_S and enters the Slumber power management state. |
| R_ERR | Reception error | Current node (host or device) detected error in received payload. |
| R_IP | Reception in Progress | Current node (host or device) is receiving payload. |
| R_OK | Reception with no error | Current node (host or device) detected no error in received payload. |

*(continued)*

**Table 20 – Description of primitives** *(concluded)*

| Primitive | Name | Description |
|---|---|---|
| R_RDY | Receiver ready | Current node (host or device) is ready to receive payload. |
| SOF | Start of frame | Start of a frame. Payload and CRC follow to EOF. |
| SYNC | Synchronization | Synchronizing primitive – always idle. |
| WTRM | Wait for frame termination | After transmission of any of the EOF, the transmitter will transmit WTRM while waiting for reception status from receiver. |
| X_RDY | Transmission data ready | Current node (host or device) has payload ready for transmission |

*(concluded)*

### 7.4.3 Primitive encoding

The following table defines the encoding for each primitive.

**Table 21 – Primitive encoding**

| Primitive name | Byte 3 contents | Byte 2 contents | Byte 1 contents | Byte 0 contents |
|---|---|---|---|---|
| ALIGN | D27.3 | D10.2 | D10.2. | K28.5 |
| CONT | D25.4 | D25.4 | D10.5 | K28.3 |
| DMAT | D22.1 | D22.1 | D21.5 | K28.3 |
| EOF | D21.6 | D21.6 | D21.5 | K28.3 |
| HOLD | D21.6 | D21.6 | D10.5 | K28.3 |
| HOLDA | D21.4 | D21.4 | D10.5 | K28.3 |
| PMACK | D21.4 | D21.4 | D21.4 | K28.3 |
| PMNAK | D21.7 | D21.7 | D21.4 | K28.3 |
| PMREQ_P | D23.0 | D23.0 | D21.5 | K28.3 |
| PMREQ_S | D21.3 | D21.3 | D21.4 | K28.3 |
| R_ERR | D22.2 | D22.2 | D21.5 | K28.3 |
| R_IP | D21.2 | D21.2 | D21.5 | K28.3 |
| R_OK | D21.1 | D21.1 | D21.5 | K28.3 |
| R_RDY | D10.2 | D10.2 | D21.4 | K28.3 |
| SOF | D23.1 | D23.1 | D21.5 | K28.3 |
| SYNC | D21.5 | D21.5 | D21.4 | K28.3 |
| WTRM | D24.2 | D24.2 | D21.5 | K28.3 |
| X_RDY | D23.2 | D23.2 | D21.5 | K28.3 |

### 7.4.4 Abort primitive

Because of its unique qualities, the abort primitive - DMAT - will be discussed in detail in this section.

Parallel ATA devices can abort a DMA transfer and post an error, by negating DMARequest, updating the Error and Status registers, and possibly setting the INTRQ line. This can be performed for both reads from and writes to the device.

For example, in the case of a DMA read from device, a Serial ATA device may terminate the transfer with an EOF, and send a Register Device to Host FIS to the host, with Error and Status registers updated appropriately. In the case of a DMA write to device, the device sends a DMA Activate FIS to the host, and then after receiving an SOF, will have to accept all data until receiving an EOF from the host. Since the device cannot terminate such a transfer once started, a special abort primitive is used.

The DMA Terminate (DMAT) primitive may be sent on the back channel during transmission of a Data FIS to signal the transmitter to terminate the transfer in progress. It may be used for both host to device transfers and for device to host transfers. Reception of the DMAT signal shall cause the recipient to close the current frame by inserting the CRC and EOF, and return to the idle state.

For host to device data transfers, upon receiving the DMAT signal the host shall terminate the transfer in progress by deactivating its DMA engine and closing the frame with valid CRC and EOF. The host DMA engine shall preserve its state at the point it was deactivated so that the device may resume the transmission at a later time by transmitting another DMA Active FIS to re-activate the DMA engine. The device is responsible for either subsequently resuming the terminated transfer by transmitting another DMA Activate FIS or closing the affected command with appropriate status.

For device to host transfers, receipt of DMAT signal by the device results in permanent termination of the transfer and is not resumable. The device shall terminate the transmission in progress and close the frame with a valid CRC and EOF, and shall thereafter clean up the affected command by indicating appropriate status for that command. No facility for resuming a device to host transfer terminated with the DMAT signal is provided.

Some implementations may have an implementation-dependent latency associated with closing the affected Data FIS in response to the DMAT signal. For example, a host controller may have a small transmit FIFO, and in order for the DMA engine to accurately reflect a resumable state, the data already transferred by the DMA engine to the transmit FIFO may have to be transmitted prior to closing the affected Data FIS. Conservative designs should minimize the DMAT response latency while being tolerant of other devices having a long latency.

### 7.4.5 Continue primitive

In order to accommodate EMI reductions, scrambling of data is incorporated in Serial ATA as described in Section 7.5. The scrambling of data is simple, with a linear feedback shift register (LFSR) used in generating the scrambling pattern being reset at each SOF primitive, or rolling over every 2048 Dwords. However, the scrambling of primitives is not as effective or simple because of the small number of control characters available. In order to accommodate EMI reductions, repeated primitives are eliminated through the use of the CONT primitive.

Any repetitive primitive may be implied to continue repeating through the use of the CONT primitive. The recipient of the CONT primitive shall ignore all data received after the CONT primitive until the reception of any primitive, excluding ALIGN. After transmitting the CONT character, the transmitter may send any sequence of data characters to the recipient provided that no primitives are included. The reception of a CONT primitive shall cause the last valid primitive to be implied as repeated until the reception of the next valid primitive.

To improve overall protocol robustness and avoid potential timeout situations caused by a reception error in a primitive, all repeated primitives shall be transmitted a minimum of twice before a CONT primitive is transmitted. The first primitive correctly received is the initiator of any action within the receiver. This avoids scenarios, for example, where X_RDY is sent from the host, followed by a CONT, and the X_RDY is received improperly resulting in the device not returning an R_RDY and causing the system to deadlock until a timeout/reset condition occurs.

The transmission of a CONT primitive is optional, but the ability to receive and properly process the CONT primitive is required. The insertion of a single, or two repetitive primitives not followed by a CONT primitive is valid (i.e. data, data, HOLD, data).

The following primitives may be followed by a CONT: HOLD, HOLDA, PMREQ_P, PMREQ_S, R_ERR, R_IP, R_OK, R_RDY, SYNC, WTRM and X_RDY.

The host PHY initialization state machine consumes the first few received primitives before communications between the host and device have been established (see state HP7:HR_SendAlign in section 6.8.1.1.1). In order to ensure proper synchronization between the host and device after entry into the L1:L_IDLE state from the LS3:L_SendAlign state or the LPM8:L_WakeUp2 state (see section 7.6.1.2 and section 7.6.1.5), the use of the CONT primitive is not allowed after a transition from the LS3:L_SendAlign state or the LPM8:L_WakeUp2 state to the L1:L_IDLE state until either a minimum of 10 non-ALIGN primitives have been transmitted or until receipt of a primitive other than SYNC or ALIGN has been detected.

Figure 58 illustrates use of the CONT primitive in the transmission of an FIS.

| Transmitter | | Receiver |
|---|---|---|
| XXXX | | XXXX |
| XXXX | | XXXX |
| X_RDY | | XXXX |
| X_RDY | | XXXX |
| CONT | | XXXX |
| XXXX | | XXXX |
| XXXX | | R_RDY |
| XXXX | | R_RDY |
| XXXX | | CONT |
| SOF | | XXXX |
| TYPE | | XXXX |
| DATA | | XXXX |
| DATA | | R_IP |
| DATA | | R_IP |
| DATA | | CONT |
| HOLD | | XXXX |
| HOLD | | XXXX |
| CONT | | XXXX |
| XXXX | | XXXX |
| XXXX | | HOLDA |
| XXXX | | HOLDA |
| HOLD | | CONT |
| DATA | | XXXX |
| DATA | | XXXX |
| DATA | | XXXX |
| CRC | | XXXX |
| EOF | | R_IP |
| WTRM | | R_IP |
| WTRM | | CONT |
| WTRM | | XXXX |
| CONT | | XXXX |
| XXXX | | R_OK |
| XXXX | | R_OK |
| XXXX | | CONT |
| XXXX | | XXXX |
| SYNC | | XXXX |
| SYNC | | XXXX |
| CONT | | XXXX |
| XXXX | | XXXX |
| XXXX | | SYNC |
| XXXX | | SYNC |
| XXXX | | CONT |
| XXXX | | XXXX |
| XXXX | | XXXX |

NOTE –
XXXX Scrambled data values (non-primitives)
DATA FIS payload data

**Figure 58 – CONT usage example**

#### 7.4.5.1 Scrambling of data following the continue primitive

The data following the CONT shall be the output of an LFSR which implements the same polynomial as is used to scramble FIS contents. That polynomial is defined in section 7.5.

The resulting LFSR value shall be encoded using the 8b/10b rules for encoding data characters before transmission by the Link layer.

The LFSR used to supply data after the CONT primitive shall be reset to the initial value upon detection of a COMINIT or COMRESET event.

Since the data following a CONT primitive is discarded by the Link layer, the value of the LFSR is undefined between CONT primitives. That is, the LFSR result used for CONT sequence N is not required to be continuous from the last LFSR result of CONT sequence N-1.

The sequence of LFSR values used to scramble the payload contents of an FIS shall not be affected by the scrambling of data used during repeated primitive suppression. That is, the data payload LFSR shall not be advanced during repeated primitive suppression and shall only be advanced for each data payload character that is scrambled using the data payload LFSR. See section 7.5 for additional information on scrambling and repeated primitive suppression.

### 7.4.6 ALIGN primitive

The Link layer shall be agnostic to reception of ALIGN primitives. The Phy layer is free to consume received ALIGN primitives. Implementations where the Phy does not consume received ALIGN primitives shall effectively drop received ALIGN primitives at the input to the Link layer or shall include Link layer processing that yields behavior equivalent to the behavior produced if all received ALIGN primitives are consumed by the Phy and not presented to the Link.

### 7.4.7 Flow control signaling latency

There is a finite pipeline latency in a round-trip handshake across the Serial ATA interface, In order to accommodate efficient system design with sufficient buffering headroom to avoid buffer overflow in flow control situations, the maximum tolerable latency from when a receiver issues a HOLD signal until it receives the HOLDA signal from the transmitter must be specified. This allows the high-water mark to be set in the receive FIFO so as to avoid buffer overflow while avoiding excessive buffering/FIFO space.

In the case where the receiver wants to flow control the incoming data, it transmits HOLD characters on the back channel. Some number of received Dwords later, valid data ceases, and HOLDA characters are received. The larger the latency between transmitting HOLD until receiving HOLDA, the larger the receive FIFO needs to be. The maximum allowed latency from the time the MSB of the HOLD primitive is on the wire, until the MSB of the HOLDA is on the wire shall be no more than 20 Dword symbol times. The LSB is transmitted first. A receiver shall be able to accommodate reception of 20 Dwords of additional data after the time it transmits the HOLD flow control character to the transmitter, and the transmitter shall respond with a HOLDA in response to receiving a HOLD character within 20 Dword symbol times.

There is no reference design in the Serial ATA standard. The specified maximum latency figure is based on the layers and states described throughout this document. It is recognized that the Link Layer may have two separate clock domains -- transmit clock domain, and the receive clock domain. It is also recognized that a Link state machine could run at the Dword clock rate, implying synchronizers between three potential clock domains. In practice more efficient implementations would be pursued and the actual latencies may be less than indicated here. The figures represent an almost literal interpretation of the spec into logic design. A synchronizer is assumed to be a worst case of 2.99 clocks of any clock domain and is rounded to three whole clocks. The Serial ATA cable contains less than ½ Dword in generations 1 and 2, and is therefore rounded to 0. Two Dwords of

pipeline delay are assumed for the Phy, and the FIFO is assumed to run at the Link state machine rate. No synchronization is needed between the two.

The following figure outlines the origin of the 20 Dword latency specification. The example illustrates the components of a round trip delay when the receiver places a HOLD on the bus until reception of the HOLDA from the transmitter. This corresponds to the number of Dwords that the receiver must be able to accept after transmitting a HOLD character.

Receiver sends HOLD:

| | |
|---|---|
| 1 Dword | Convert to 40 bit data. |
| 1 Dword | 10b/8b conversion. |
| 1 Dword | De scrambling. |
| 3 Dwords | Synchronization between receive clock, and Link state machine clock. |
| 1 Dword | Link state machine is notified that primitive has been received. |
| 1 Dword | Link state machine takes action. |
| 1 Dword | FIFO is notified of primitive reception. |
| 1 Dword | FIFO stops sending data to Link layer. |
| 1 Dword | Link is notified to insert HOLDA. |
| 1 Dword | Link acts on notification and inserts HOLDA into data stream. |
| 1 Dword | Scrambling. |
| 1 Dword | 8b/10b conversion. |
| 1 Dword | Synchronize to transmit clock (3 transmit clocks, which are four times the Link state machine rate). |
| 1 Dword | Convert to 10 bit data. |
| 2 Dwords | Phy, transmit side. |

HOLDA on the cable.

**Table 22 – SRST write from host to device transmission breaking through a device to host Data FIS**

| Host driver | Device driver | Description |
|---|---|---|
| … | … | Previous activity abbreviated for clarity |
| R_IP | Data n | Device transmitting data |
| R_IP | Data n+1 | |
| R_IP | HOLD | Device transmit FIFO empty, and flow control applied |
| R_IP | HOLD | Host receives and decodes HOLD flow control |
| HOLDA | HOLD | Host acknowledges flow control. Device internally deadlocked and no more data forthcoming (drive hung) |
| HOLDA | HOLD | |
| … | … | System in this state until host decides to reset drive |
| HOLDA | HOLD | Host detects SRST write to control register, needs to break deadlock |
| SYNC | HOLD | Host transmits SYNC primitive to abort current transmission |
| SYNC | HOLD | Device receives and decodes SYNC, abandons transmission in progress |
| SYNC | SYNC | Host sends SYNC / Device sends SYNC (both returned to idle state) |
| SYNC | SYNC | Host receives and decodes SYNC, can now initiate new FIS transmission |
| X_RDY | SYNC | Host ready to send Shadow register block registers for SRST write |
| X_RDY | SYNC | Device decodes X_RDY |
| X_RDY | R_RDY | Device indicates ready to receive |
| X_RDY | R_RDY | Host decodes R_RDY |

| | | |
|---|---|---|
| SOF | R_RDY | Host starts a frame |
| etc | etc | etc |

### 7.4.8 Examples

The following examples illustrate basic primitive usage. They do not show detailed lengthy sequences which will invoke the use of the CONT primitive.

**Table 23. – Command Shadow Register Block register transmission example**

| Host driver | Device driver | Description |
|---|---|---|
| SYNC | SYNC | Idle condition |
| SYNC | SYNC | Idle condition |
| X_RDY | SYNC | Host ready to send Shadow register block registers |
| X_RDY | SYNC | Device decodes X_RDY |
| X_RDY | R_RDY | Device indicates ready to receive |
| X_RDY | R_RDY | Host decodes R_RDY |
| SOF | R_RDY | Host starts a frame |
| Hdr 0 | R_RDY | Host sends Register FIS Dword 0 / device decodes SOF |
| Hdr 1 | R_IP | Host sends Register FIS Dword 1 / device stores Hdr 0 |
| … | … | … |
| Hdr n | R_IP | Host sends Register FIS Dword n / device stores Hdr n-1 |
| CRC | R_IP | Host sends CRC / device stores Hdr n |
| EOF | R_IP | Host sends EOF / device stores CRC |
| WTRM | R_IP | Device decodes EOF |
| WTRM | R_IP | Device computes good CRC and releases TF contents |
| WTRM | R_OK | Device sends good end |
| WTRM | R_OK | Host decodes R_OK as good results |
| SYNC | R_OK | Host releases interface |
| SYNC | R_OK | Device decodes release by host - is allowed to release |
| SYNC | SYNC | Idle condition |

**Table 24. – Data from host to device transmission example**

| Host driver | Device driver | Description |
|---|---|---|
| SYNC | SYNC | Idle condition |
| SYNC | SYNC | Idle condition |
| X_RDY | SYNC | Host ready to send Shadow register block registers |
| X_RDY | SYNC | Device decodes X_RDY |
| X_RDY | R_RDY | Device indicates ready to receive |
| X_RDY | R_RDY | Host decodes R_RDY |
| SOF | R_RDY | Host starts a frame |
| Hdr 0 | R_RDY | Host sends header Dword 0 / device decodes SOF |
| Hdr 1 | R_IP | Host sends header Dword 1 / device stores header Dword 0 |
| ... | .. | ... |
| Dat x | R_IP | Host sends data Dword x / device stores data Dword (x-1) |
| HOLD | R_IP | Host sends HOLD / device stores data Dword (x) and decodes HOLD |
| HOLD | HOLDA | Device acknowledges HOLD |
| HOLD | HOLDA | Host decodes HOLDA – host may release HOLD at any time |
| Dat(n-3) | HOLDA | Host sends (n-2)th data Dword / device decodes data Dword |
| Dat(n-2) | R_IP | Host sends (n-1)th data Dword / device Stores (n-2)th data Dword |
| Dat(n-1) | R_IP | Host sends nth data Dword / device ores (n-1)th data Dword |
| CRC | R_IP | Host sends CRC / device stores nth data Dword |
| EOF | R_IP | Host sends EOF / device stores CRC |
| WTRM | R_IP | Device decodes EOF |
| WTRM | R_IP | Device computes good CRC and releases data contents |
| WTRM | R_OK | Device sends good end |
| WTRM | R_OK | Host decodes R_OK as good results |
| SYNC | R_OK | Host releases interface |
| SYNC | R_OK | Device decodes release by host - is allowed to release |
| SYNC | SYNC | Idle condition |

**Table 25. – DMA data from host to device, device terminates transmission example**

| Host driver | Device driver | Description |
|---|---|---|
| SYNC | SYNC | Idle condition |
| SYNC | SYNC | Idle condition |
| X_RDY | SYNC | Host ready to send data |
| X_RDY | SYNC | Device decodes X_RDY |
| X_RDY | R_RDY | Device indicates ready to receive |
| X_RDY | R_RDY | Host decodes R_RDY |
| SOF | R_RDY | Host starts a frame |
| Hdr 0 | R_RDY | Host sends header Dword 0 / device decodes SOF |
| Dat 0 | R_IP | Host sends data Dword 0 |
| ... | .. | Host sends data Dword x / device stores data Dword |
| Dat x | DMAT | Device decides to terminate, sends DMAT |
| Dat x | R_IP | Host decodes DMAT – host prepares to terminate |
| CRC | R_IP | Host sends current CRC value |
| EOF | R_IP | Host sends EOF / device stores CRC |
| WTRM | R_IP | Device decodes EOF |
| WTRM | R_IP | Device computes good CRC and releases data contents |
| WTRM | R_OK | Device sends good end |
| WTRM | R_OK | Host decodes R_OK as good results |
| SYNC | R_OK | Host releases interface |
| SYNC | R_OK | Device decodes release by host - is allowed to release |
| SYNC | SYNC | Idle condition |

## 7.5 CRC calculation and scrambling of FIS contents

### 7.5.1 CRC

The CRC (Cyclic Redundancy Check) of a frame is a Dword (32-bit) field that shall follow the last Dword of the contents of an FIS and precede the EOF primitive. The CRC calculation covers all of the FIS transport data between the SOF and EOF primitives, and excludes any intervening primitives and CONT stream contents. The CRC value shall be computed on the contents of the FIS before encoding for transmission (scrambling) and after decoding upon reception.

The CRC shall be calculated on Dword quantities. If an FIS contains an odd number of words the last word of the FIS shall be padded with zeros to a full Dword before the Dword is used in the calculation of the CRC.

The CRC shall be aligned on a Dword boundary.

The CRC shall be calculated using the following 32-bit generator polynomial:

$$G(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$$

The CRC value shall be initialized with a value of 52325032h before the calculation begins.

The maximum number of Dwords between the SOF primitive to the EOF primitive shall not exceed 2064 Dwords including the FIS type and CRC..

The contents of a frame shall be scrambled before transmission by the physical layer.

Scrambling shall be performed on Dword quantities by XORing the data to be transmitted with the output of a linear feedback shift register (LFSR). The shift register shall implement the following polynomial:

$$G(X) = X^{16} + X^{15} + X^{13} + X^{4} + 1$$

The serial shift register shall be initialized with a value of FFFFh before the first shifted output. The shift register shall be initialized to the seed value before the SOF primitive. All data words between the SOF and EOF shall be scrambled, including the CRC.

#### 7.5.1.1 Relationship between scrambling of FIS data and repeated primitives

There are two separate scramblers used in Serial ATA. One scrambler is used for the data payload encoding and a separate scrambler is used for repeated primitive suppression. The scrambler used for data payload encoding shall maintain consistent and contiguous context over the scrambled payload characters of a frame (between SOF and EOF), and shall not have its context affected by the scrambling of data used for repeated primitive suppression.

Scrambling is applied to all data (non-primitive) Dwords. Primitives, including ALIGN, do not get scrambled and shall not advance the data payload LFSR register. Similarly, the data payload LFSR shall not be advanced during transmission of Dwords during repeated primitive suppression (i.e. after a CONT primitive). Since it is possible for a repeated primitive stream to occur in the middle of a data frame – multiple HOLD/HOLDA primitives are likely – care must be taken to ensure that the data payload LFSR is only advanced for each data payload character that it scrambles and that it is not advanced for primitives or for data characters transmitted as part of repeated primitive suppression which uses a separate scrambler.

### 7.5.1.2 Relationship between scrambling and CRC

The order of application of scrambling shall be as follows. For a Dword of data following the SOF primitive the Dword shall be used in the calculation of the CRC. The same Dword value shall be XORed with the scrambler output, and the resulting Dword submitted to the 8b/10b encoder for transmission. Similarly, on reception, the Dword shall be decoded using a 10b/8b decoder, the scrambler output shall be XORed with the resulting Dword, and the resulting Dword presented to the Link layer and subsequently used in calculating the CRC. The CRC Dword shall be scrambled according to the same rules.

## 7.6 Link layer state diagrams

#### 7.6.1.1 Terms used in Link layer transition tables

1. LRESET: Link layer COMRESET signal
2. PhyNRdy: Phy status as defined in section 6.4.2
3. PhyRdy: Phy status as defined in section 6.4.2
4. DecErr: Bad decode of a 32 bit Dword transferred from Phy to Link
   - Invalid 10b pattern
   - Disparity error
   - Primitive with a control character in the first byte but not an allowed control character
   - Any control character in other than the first byte of the Dword
5. DatDword: A 32 bit pattern that is formed correctly, but does not have the primitive leading 10b pattern (K28.5 or K28.3).
6. COMWAKE: Signal from the out of band detector in the Phy indicating that the COMWAKE out of band signal is being detected.
7. AnyDword: A 32 bit pattern of any type - even one with DecErr received from Phy
8. The “p” subscript used in the tables designate the name immediately preceding to represent a primitive

### 7.6.1.2 Link idle state diagram

| L1: L_IDLE[4] | Transmit $SYNC_P$ | |
|---|---|---|
| 1. Transport layer requests frame transmission and PhyRdy[2] | → | HL_SendChkRdy or DL_SendChkRdy[1] |
| 2. Transport layer requests transition to Partial and PhyRdy[2] | → | L_TPMPartial |
| 3. Transport layer requests transition to Slumber and PhyRdy[2] | → | L_TPMSlumber |
| 4. $X\_RDY_P$ received from Phy | → | L_RcvWaitFifo |
| 5. Phy layer forwards ( $PMREQ\_P_P$ or $PMREQ\_S_P$ ) and power modes are enabled | → | L_PMOff |
| 6. Phy layer forwards ( $PMREQ\_P_P$ or $PMREQ\_S_P$ ) and power modes are disabled | → | L_PMDeny |
| 7. Phy layer forwards AnyDword other than ( $X\_RDY_P$ or $PMREQ\_P_P$ or $PMREQ\_S_P$ ) and no transmit request from Transport layer[2,3] | → | L_IDLE |
| 8. PhyNRdy | → | L_NoCommErr |

NOTE –
1. The host Link layer makes a transition to the HL_SendChkRdy state; the device Link layer makes a transition to the DL_SendChkRdy state.
2. This transition is taken even if errors such as 10b decoding errors are detected.
3. This statement also ignores any unrecognized sequences or commands not defined in this specification.
4. Upon entry to this state from the LS3:L_SendAlign state or the LPM8:L_WakeUp2 state, use of the CONT primitive is subject to restrictions as outlined in section 7.4.5.

| LS1: L_NoCommErr | Post Phy not ready error to Transport layer | |
|---|---|---|
| 1. Unconditional | → | L_NoComm |

| LS2: L_NoComm | Transmit $ALIGN_P$[1] | |
|---|---|---|
| 1. PhyNRdy | → | L_NoComm |
| 2. PhyRdy | → | L_SendAlign |
| NOTE –<br>1. Also deactivate any signal for Phy layer to abort operation | | |

| LS3: L_SendAlign | Transmit $ALIGN_P$ | |
|---|---|---|
| 1. PhyNRdy | → | L_NoCommErr |
| 2. PhyRdy | → | L_IDLE |

| LS4: L_RESET | Reset Link state to initial conditions | |
|---|---|---|
| 1. LRESET Link reset signal asserted | → | L_RESET |
| 2. LRESET Link reset signal deasserted | → | L_NoComm |
| NOTE –<br>1. This state is entered ANYTIME the link reset control is active. | | |

**L1: L_IDLE state:** This state is entered when a frame transmission has been completed by the Link layer.

When in this state, the Link layer transmits the SYNC primitive and waits for an X_RDY primitive from the Phy layer or a frame transmission request from the Transport layer.

**Transition L1:1a:** When the host Link layer receives a request to transmit a frame from the Transport layer and the Phy layer is ready, the Link layer shall make a transition to the LT1: HL_SendChkRdy state.

**Transition L1:1b:** When the device Link layer receives a request to transmit a frame from the Transport layer and the Phy layer is ready, the Link layer shall make a transition to the LT2: DL_SendChkRdy state.

**Transition L1:2:** When the Link layer receives a request to enter the Partial power mode from the Transport layer and the Phy layer is ready, the Link layer shall make a transition to the L_TPMPartial state.

**Transition L1:3:** When the Link layer receives a request to enter the Slumber power mode from the Transport layer and the Phy layer is ready, the Link layer shall make a transition to the L_TPMSlumber state.

**Transition L1:4:** When the Link layer receives an X_RDY from the Phy layer, the Link layer shall make a transition to the LR2: L_RcvWaitFifo state.

**Transition L1:5:** When the Link layer receives a PMREQ_P or PMREQ_S from the Phy layer and is enabled to perform power management modes, the Link layer shall make a transition to the LPM3: L_PMOff state.

**Transition L1:6:** When the Link layer receives a PMREQ_P or a PMREQ_S from the Phy layer and is not enabled to perform power management modes, the Link layer shall make a transition to the LR0: L_PMDeny state.

**Transition L1:7:** When the Link layer does not receive a request to transmit a frame from the Transport layer, does not receive a request to go to a power mode from the Transport layer, does not receive an X_RDY from the Phy layer or does not receive a PMREQ_x from the Phy layer the Link layer shall make a transition to the L1: L_IDLE state.

**Transition L1:8:** If the Phy layer becomes not ready even if the Transport layer is requesting an operation, the Link layer transitions to the L_NoCommErr state.

**LS1: L_NoCommErr state:** This state is entered upon detection of a non ready condition of the Phy layer while attempting to process another state. The entry into this state heralds a relatively serious error condition in the Link layer. This state is executed only once so as to pass on the error condition up to the Transport layer.

**Transition LS1:1:** The transition is made to LS1:L_NoComm unconditionally.

**LS2: L_NoComm state:** This state is entered directly from the LS1:L_NoCommErr state or the LS4:L_RESET State. The Link Layer remains in this state until the Phy signals that it has established communications and is ready.

**Transition LS2:1:** For as long as the Phy layer stays not ready, the transition is made to LS2: L_NoComm.

**Transition LS2:2:** When the Phy layer signals it is ready, a transition is made to LS3: L_SendAlign.

**LS3: L_SendAlign state:** This state is entered whenever an ALIGN needs to be sent to the Phy layer.

**Transition LS3:1:** If the Phy layer becomes not ready, then a transition is made to LS1: L_NoCommErr.

**Transition LS3:2:** If the Phy indicates that it is ready, a transition is made to the L1: L_IDLE state.

**LS4: L_RESET state:** This state is entered whenever the Link LRESET control is active. All Link layer hardware is initialized to and held at a known state/value. While in this state all requests or triggers from other layers are ignored. While in this state, the Phy reset signal is also asserted.

**Transition LS4:1:** While the RESET control is active a transition is made back to the LS4: L_RESET state.

**Transition LS4:2:** When the RESET control goes inactive a transition is made to the LS2: L_NoComm state.

### 7.6.1.3 Link transmit state diagram

| LT1: HL_SendChkRdy | Transmit $X\_RDY_P$ | |
|---|---|---|
| 1. $R\_RDY_P$ received from Phy | → | L_SendSOF |
| 2. $X\_RDY_P$ received from Phy | → | L_RcvWaitFifo |
| 3. AnyDword other than ($R\_RDY_P$ or $X\_RDY_P$)[1] received from Phy layer | → | HL_SendChkRdy |
| 4. PhyNRdy | → | L_NoCommErr[2] |
| 1. NOTE –Any received errors such as 10b decoding errors and invalid primitives are ignored<br>2. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer | | |

| LT2: DL_SendChkRdy | Transmit $X\_RDY_P$ | |
|---|---|---|
| 1. $R\_RDY_P$ received from Phy | → | L_SendSOF |
| 2. AnyDword other than $R\_RDY_P$ received from Phy | → | DL_SendChkRdy |
| 3. PhyNRdy | → | L_NoCommErr |

| LT3: L_SendSOF | Transmit $SOF_P$ | |
|---|---|---|
| 1. PhyRdy[1] | → | L_SendData |
| 2. PhyNRdy | → | L_NoCommErr[2] |
| 3. $SYNC_P$ received from Phy | → | L_IDLE[2] |
| NOTE –<br>1. Any received errors such as 10b decoding errors and invalid primitives are ignored<br>2. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer | | |

| LT4: L_SendData | Transmit data $Dword_W$ | |
|---|---|---|
| 1. More data to transmit and AnyDword other than ($HOLD_P$ or $DMAT_p$ or $SYNC_P$ ) received from Phy[1,2] | → | L_SendData |
| 2. More data to transmit and $HOLD_P$ received from Phy | → | L_RcvrHold |
| 3. Data transmit not complete and data not ready to transmit and AnyDword other than $SYNC_P$ received from Phy | → | L_SendHold |
| 4. $DMAT_p$ received from Phy or data transmit complete and AnyDword other than $SYNC_P$ received from Phy | → | L_SendCRC |
| 5. $SYNC_P$ received from Phy | → | L_IDLE[3] |
| 6. PhyNRdy | → | L_NoCommErr[3] |
| 7. Host Transport Layer indicates request to transmit Control Register frame[4] | → | L_IDLE |

NOTE –
1. Any received errors such as 10b decoding errors and invalid primitives are ignored
2. This makes possible a back channel during this time
3. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer.
4. When this condition is true, the associated transition has priority over all other transitions exiting this state.

| LT5: L_RcvrHold | Transmit $HOLDA_P$ | |
|---|---|---|
| 1. More data to transmit and AnyDword other than ($HOLD_p$ or $SYNC_p$ or $DMAT_p$) received from Phy | → | L_SendData |
| 2. More data to transmit and $HOLD_P$ received from Phy or DecErr | → | L_RcvrHold |
| 3. More data to transmit and $SYNC_P$ received from Phy | → | L_IDLE[1] |
| 4. More data to transmit and $DMAT_p$ received from Phy | → | L_SendCRC |
| 5. PhyNRdy | → | L_NoCommErr[1] |
| 6. Host Transport Layer indicates request to transmit Control Register frame[2] | → | L_IDLE |
| 7. $SYNC_P$ received from Phy | → | L_IDLE[1] |

NOTE –
1. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer
2. When this condition is true, the associated transition has priority over all other transitions exiting this state.

| LT6: L_SendHold | Transmit $HOLD_P$ | |
|---|---|---|
| 1. More data ready to transmit and AnyDword other than ($HOLD_P$ or $SYNC_P$) received from Phy | → | L_SendData |
| 2. More data ready to transmit and $HOLD_P$ received from Phy | → | L_RcvrHold |
| 3. Data transmit not complete and data not ready to transmit and AnyDword other than ($SYNC_P$ or $DMAT_p$) received from Phy | → | L_SendHold |
| 4. DMATp received from Phy | → | L_SendCRC |
| 5. $SYNC_P$ received from Phy | → | L_IDLE[1] |
| 6. PhyNRdy | → | L_NoCommErr[1] |
| 7. Host Transport Layer indicates request to transmit Control Register frame[2] | → | L_IDLE |

NOTE –
1. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer.
2. When this condition is true, the associated transition has priority over all other transitions exiting this state.

| LT7: L_SendCRC | Transmit $CRC_W$ | |
|---|---|---|
| 1. PhyRdy and $SYNC_P$ not received from Phy | → | L_SendEOF |
| 2. PhyNRdy | → | L_NoCommErr[1] |
| 3. PhyRdy and $SYNC_P$ received from Phy | → | L_IDLE[1] |

NOTE –
1. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer.

| LT8: L_SendEOF | Transmit $EOF_P$ | |
|---|---|---|
| 1. PhyRdy and $SYNC_P$ not received from Phy | → | L_Wait |
| 2. PhyNRdy | → | L_NoCommErr[1] |
| 3. PhyRdy and $SYNC_P$ received from Phy | → | L_IDLE[1] |

NOTE –
1. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer.

| LT9: L_Wait | Transmit $WTRM_P$ | |
|---|---|---|
| 1. $R\_OK_P$ received from Phy | → | L_IDLE (good status) |
| 2. $R\_ERR_P$ received from Phy | → | L_IDLE (bad status) |
| 3. $SYNC_P$ received from Phy | → | L_IDLE[1] |
| 4. AnyDword other than ($R\_OK_P$ or $R\_ERR_P$ , or $SYNC_P$ ) received from Phy | → | L_Wait |
| 5. PhyNRdy | → | L_NoCommErr[1] |

NOTE –
1. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer.

**LT1: HL_SendChkRdy state:** This state is entered when a frame transmission has been requested by the host Transport layer.

When in this state, the Link layer transmits an X_RDY primitive and waits for an X_RDY primitive or R_RDY primitive from the Phy layer.

> NOTE – It is possible that both the host and the device simultaneously request frame transmission by transmitting X_RDY. If the host receives X_RDY while transmitting X_RDY, the host shall back off and enter the L_RcvChkRdy state, postponing its desired frame transmission until the device has completed its frame transmission and the bus is idle.

**Transition LT1:1:** When the host Link layer receives an R_RDY primitive from the Phy layer, the Link layer shall make a transition to the LT3: L_SendSOF state.

**Transition LT1:2:** When the host Link layer receives an X_RDY primitive from the Phy layer, the Link layer shall make a transition to the LR2: L_RcvWaitFifo state.

**Transition LT1:3:** When the host Link layer receives any Dword other than an R_RDY or an X_RDY primitive from the Phy layer, the Link layer shall make a transition to the LT1: HL_SendChkRdy state.

**Transition LT1:4:** When the host Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**LT2: DL_SendChkRdy state:** This state is entered when a frame transmission has been requested by the device Transport layer.

When in this state, the Link layer transmits an X_RDY primitive and waits for an R_RDY primitive from the Phy layer.

**Transition LT2:1:** When the device Link layer receives an R_RDY primitive from the Phy layer, the Link layer shall make a transition to the LT3: L_SendSOF state.

**Transition LT2:2:** When the device Link layer does not receive an R_RDY primitive from the Phy layer, the Link layer shall make a transition to the LT2: DL_SendChkRdy state.

**Transition LT2:3:** When the device Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**LT3: L_SendSOF state:** This state is entered an R_RDY primitive has been received from the Phy layer.

When in this state, the Link layer transmits an SOF primitive.

**Transition LT3:1:** When the device Link layer has transmitted an SOF primitive, the Link layer shall make a transition to the LT4: L_SendDATA state.

**Transition LT3:2:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**Transition LT3:3:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall notify the Transport layer of the illegal transition error condition and shall make a transition to the L1:L_IDLE state.

**LT4: L_SendData state:** This state is entered when an SOF primitive has been transmitted.

When in this state, the Link layer takes a data Dword from the Transport layer, encodes the Dword, and transmits it. The Dword is also entered into the CRC calculation before encoding.

**Transition LT4:1:** When the Link layer receives any Dword other than a HOLD, DMAT, or SYNC primitive from the Phy layer and the Transport layer indicates a Dword is available for transfer, the Link layer shall make a transition to the LT4: L_SendData state. The DMAT signal is advisory and data transmission should be halted at the earliest opportunity but is not required to cease immediately. It is therefore allowable to stay in the LT4: L_SendData state when there is more data to transmit and DMAT is received.

**Transition LT4:2:** When the device Link layer receives a HOLD primitive from the Phy layer, the Link layer shall make a transition to the LT5: L_RcvrHold state.

**Transition LT4:3:** When the Transport layer indicates that the next Dword is not available to transfer and any Dword other than a SYNC primitive has been received from the Phy layer, the Link layer shall make a transition to the LT6: L_SendHold state.

**Transition LT4:4:** When the Transport layer indicates that all data for the frame has been transferred and any Dword other than a SYNC primitive has been received from the Phy layer, the Link layer shall make a transition to the LT7: L_SendCRC state. When the Link layer receives a DMAT primitive from the Phy layer, it shall notify the Transport layer and terminate the transmission in progress as described in section 7.4.4 and shall transition to the LT7: L_SendCRC state.

**Transition LT4:5:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall notify the Transport layer of the illegal transition error condition and shall make a transition to the L1:L_IDLE state
**Transition LT4:6:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**Transition LT4:7:** When the host Link layer receives notification from the host Transport layer that a Control Register FIS is pending for transmission (typically due to SRST being toggled), the current transfer shall be aborted and a transition to the L_IDLE state shall be made. The return to the L_IDLE state results in the transmission of SYNC primitives, which causes the receiving Link layer to also transition to the L_IDLE state. The Control Register FIS may then be transmitted as a Register FIS Host to Device.

**LT5: L_RcvrHold state:** This state is entered when a HOLD primitive has been received from the Phy layer.

When in this state, the Link layer shall transmit the HOLDA primitive.

**Transition LT5:1:** When the Link layer receives any Dword other than a HOLD, SYNC, or a DMAT primitive from the Phy layer and the Transport layer indicates that a Dword is available for transfer, the Link layer shall make a transition to the LT4: L_SendData state.

**Transition LT5:2:** When the device Link layer receives a HOLD primitive from the Phy layer or a decoding error was detected, the Link layer shall make a transition to the LT5: L_RcvrHold state.

**Transition LT5:3:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall make a transition to the L1: L_IDLE state. The Transport layer shall be notified of the illegal transition error condition.

**Transition LT5:4:** When the Link layer receives a DMAT primitive from the Phy layer, it shall notify the Transport layer and terminate the transmission in progress as described in section 7.4.4 and shall transition to the LT7: L_SendCRC state.

**Transition LT5:5:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**Transition LT5:6:** When the host Link layer receives notification from the host Transport layer that a Control Register FIS is pending for transmission (typically due to SRST being toggled), the current transfer shall be aborted and a transition to the L_IDLE state shall be made. The return to the L_IDLE state results in the transmission of SYNC primitives, which causes the receiving Link layer to also transition to the L_IDLE state. The Control Register FIS may then be transmitted as a Register FIS Host to Device.

**Transition LT5:7:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall notify the Transport layer of the illegal transition error condition and shall make a transition to the L1:L_IDLE state.

**LT6: L_SendHold state:** This state is entered when the Transport layer indicates a Dword is not available for transfer and a HOLD primitive has not been received from the Phy layer.

When in this state, the Link layer shall transmit the HOLD primitive.

**Transition LT6:1:** When the Link layer receives any Dword other than a HOLD or SYNC primitive from the Phy layer and the Transport layer indicates that a Dword is available for transfer, the Link layer shall make a transition to the LT4: L_SendData state.

**Transition LT6:2:** When the Link layer receives a HOLD primitive from the Phy layer and the Transport layer indicates a Dword is available for transfer, the Link layer shall make a transition to the LT5: L_RcvrHold state.

**Transition LT6:3:** When the Transport layer indicates that a Dword is not available for transfer and any Dword other than a SYNC primitive is received from the Phy layer, the Link layer shall make a transition to the LT6: L_SendHold state.

**Transition LT6:4:** When the Link layer receives a DMAT primitive from the Phy layer, it shall notify the Transport layer and terminate the transmission in progress as described in section 7.4.4 and shall transition to the LT7:L_SendCRC state.

**Transition LT6:5:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall make a transition to the L1: L_IDLE state. The Transport layer shall be notified of the illegal transition error condition.

**Transition LT6:6:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**Transition LT6:7:** When the host Link layer receives notification from the host Transport layer that a Control Register FIS is pending for transmission (typically due to SRST being toggled), the current transfer shall be aborted and a transition to the L_IDLE state shall be made. The return to the L_IDLE state results in the transmission of SYNC primitives, which causes the receiving Link layer to also transition to the L_IDLE state. The Control Register FIS may then be transmitted as a Register FIS Host to Device.

**LT7: L_SendCRC state:** This state is entered when the Transport layer indicates that all data Dwords have been transferred for this frame.

When in this state, the Link layer shall transmit the calculated CRC for the frame.

**Transition LT7:1:** When the CRC has been transmitted, the Link layer shall make a transition to the LT8: L_SendEOF state.

**Transition LT7:2:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**Transition LT7:3:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall notify the Transport layer of the illegal transition error condition and shall make a transition to the L1:L_IDLE state.

**LT8: L_SendEOF state:** This state is entered when the CRC for the frame has been transmitted.

When in this state, the Link layer shall transmit the EOF primitive.

**Transition LT8:1:** When the EOF primitive has been transmitted, the Link layer shall make a transition to the LT9: L_Wait state.

**Transition LT8:2:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**Transition LT8:3:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall notify the Transport layer of the illegal transition error condition and shall make a transition to the L1:L_IDLE state.

**LT9: L_Wait state:** This state is entered when the EOF primitive has been transmitted.

When in this state, the Link layer shall transmit the WTRM primitive.

**Transition LT9:1:** When the Link layer receives a R_OK primitive from the Phy layer, the Link layer shall notify the Transport layer and make a transition to the L1: L_IDLE state.

**Transition LT9:2:** When the Link layer receives a R_ERR primitive from the Phy layer, the Link layer shall notify the Transport layer and make a transition to the L1: L_IDLE state.

**Transition LT9:3:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall notify the Transport layer and make a transition to the L1: L_IDLE state.

**Transition LT9:4:** When the Link layer receives any Dword other than an R_OK, R_ERR, or SYNC primitive from the Phy layer, the Link layer shall make a transition to the LT9: L_Wait state.

**Transition LT9:5:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

### 7.6.1.4 Link receive state diagram

| LR1: L_RcvChkRdy | Transmit $R\_RDY_P$ | |
|---|---|---|
| 1. $X\_RDY_P$ received from Phy | → | L_RcvChkRdy |
| 2. $SOF_P$ received from Phy | → | L_RcvData |
| 3. Any Dword other than ( $X\_RDY_P$ or $SOF_P$ ) received from Phy | → | L_IDLE |
| 4. PhyNRdy | → | L_NoCommErr[1] |
| NOTE –<br>1. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer. | | |

| LR2: L_ RcvWaitFifo | Transmit $SYNC_P$ | |
|---|---|---|
| 1. $X\_RDY_P$ received from Phy and Fifo space available | → | L_RcvChkRdy |
| 2. $X\_RDY_P$ received from Phy and Fifo space not available | → | L_RcvWaitFifo |
| 3. Any Dword other than $X\_RDY_P$ received from Phy | → | L_IDLE |
| 4. PhyNRdy | → | L_NoCommErr[1] |
| NOTE –<br>1. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer. | | |

| LR3: L_RcvData | Transmit $R\_IP_P$ or $DMAT_p$[1] | |
|---|---|---|
| 1. ( $DatDword_W$ received from Phy and FIFO space ) or $HOLDA_P$ received from Phy | → | L_RcvData |
| 2. $DatDword_W$ received from Phy and insufficient FIFO space | → | L_Hold |
| 3. $HOLD_P$ received from Phy | → | L_SendHold |
| 4. $EOF_P$ received from Phy | → | L_RcvEOF |
| 5. $WTRM_P$ received from Phy | → | L_BadEnd |
| 6. $SYNC_P$ received from Phy | → | L_IDLE |
| 7. AnyDword other than ($HOLD_P$ or $EOF_P$ or $HOLDA_P$ or $SYNC_P$ or $WTRM_P$ ) received from Phy | → | L_RcvData |
| 8. PhyNRdy | → | L_NoCommErr[2] |
| 9. Host Transport Layer indicates request to transmit Control Register frame | → | L_IDLE |
| NOTE –<br>1. If the Transport layer signals that it wishes to terminate the transfer, DMAT is transmitted in place of R_IP.<br>2. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer. | | |

| LR4: L_Hold | Transmit $HOLD_P$ | |
|---|---|---|
| 1. FIFO space available and AnyDword other than $HOLD_P$ or $EOF_p$ received from Phy | → | L_RcvData |
| 2. FIFO space available and $HOLD_P$ received from Phy | → | L_SendHold |
| 3. $EOF_p$ received from Phy | → | L_RcvEOF |
| 4. No FIFO space available and $EOF_p$ not received from Phy and $SYNC_P$ not received from Phy and PhyRdy | → | L_Hold |
| 5. PhyNRdy | → | L_NoCommErr[1] |
| 6. $SYNC_P$ received from Phy | → | L_IDLE |
| 7. Host Transport Layer indicates request to transmit Control Register frame | → | L_IDLE |
| NOTE –<br>1. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer. | | |

| LR5: L_SendHold | Transmit $HOLDA_P$ or $DMAT_p$[1] | |
|---|---|---|
| 1. AnyDword other than ( $HOLD_P$ or $EOF_P$ or $SYNC_P$ ) received from Phy | → | L_RcvData |
| 2. $HOLD_P$ received from Phy | → | L_SendHold |
| 3. $EOF_P$ received from Phy | → | L_RcvEOF |
| 4. $SYNC_P$ received from Phy | → | L_IDLE |
| 5. PhyNRdy | → | L_NoCommErr[2] |
| 6. Host Transport Layer indicates request to transmit Control Register frame | → | L_IDLE |
| NOTE –<br>1. If the Transport layer signals that it wishes to terminate the transfer, DMAT is transmitted in place of HOLDA.<br>2. The Link layer shall notify the Transport layer of the condition and fail the attempted transfer. | | |

| LR6: L_RcvEOF | Transmit $R\_IP_P$ | |
|---|---|---|
| 1. CRC check not complete | → | L_RcvEOF |
| 2. CRC good | → | L_GoodCRC |
| 3. CRC bad | → | L_BadEnd |
| 4. PhyNRdy | → | L_NoCommErr |

| LR7: L_GoodCRC | Transmit $R\_IP_P$ | |
|---|---|---|
| 1. Transport Layer indicated good result | → | L_GoodEnd |
| 2. Transport Layer indicates unrecognized FIS | → | L_BadEnd |
| 3. Transport Layer has yet to respond | → | L_GoodCRC |
| 4. PhyNRdy | → | L_NoCommErr |
| 5. Transport or Link Layer indicated error detected during reception of recognized FIS | → | L_BadEnd |
| 6. $SYNC_P$ received from Phy | → | L_IDLE |
| NOTE<br>1. Upon entering this state for the first time, the Link Layer shall notify the Transport Layer that the CRC for this frame is valid. | | |

| LR8: L_GoodEnd | Transmit $R\_OK_P$ | |
|---|---|---|
| 1. $SYNC_P$ received from Phy | → | L_IDLE |
| 2. AnyDword other than $SYNC_P$ received from Phy | → | L_GoodEnd |
| 3. PhyNRdy | → | L_NoCommErr |

| LR9: L_BadEnd | Transmit $R\_ERR_P$ | |
|---|---|---|
| 1. $SYNC_P$ received from Phy | → | L_IDLE |
| 2. AnyDword other than $SYNC_P$ received from Phy | → | L_BadEnd |
| 3. PhyNRdy | → | L_NoCommErr |

**LR1: L_RcvChkRdy state:** This state is entered when an X_RDY primitive has been received from the Phy layer.

When in this state, the Link layer shall transmit an R_RDY primitive and wait for an SOF primitive from the Phy layer.

**Transition LR1:1:** When the Link layer receives an X_RDY primitive from the Phy layer, the Link layer shall make a transition to the LR1: L_RcvChkRdy state.

**Transition LR1:2:** When the Link layer receives an SOF primitive from the Phy layer, the Link layer shall make a transition to the LR2: L_RcvData state.

**Transition LR1:3:** When the Link layer receives any Dword other than an X_RDY or SOF primitive from the Phy layer, the Link layer shall notify the Transport layer of the condition and make a transition to the L1: L_IDLE state.

**Transition LR1:4:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**LR2: L_RcvWaitFifo state:** This state is entered when an X_RDY has been received, and the FIFO is not ready to receive an FIS.

When in this state, the Link layer shall transmit the SYNC primitive.

**Transition LR2:1:** When the Link layer receives a X_RDY primitive from the Phy layer and the FIFO is ready to accept data, the Link layer shall make a transition to the LR1: L_RcvChkRdy state.

**Transition LR2:2:** When the Link layer receives a X_RDY primitive from the Phy layer and the FIFO is not ready to accept data, the Link layer shall make a transition to the LR2: L_RcvWaitFifo state.

**Transition LR2:3:** When the Link layer receives any Dword other than an X_RDY primitive from the Phy layer, the Link layer shall notify the Transport layer of the condition and make a transition to the L1: L_IDLE state.

**Transition LR2:4:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**LR3: L_RcvData state:** This state is entered when an SOF primitive has been received from the Phy layer.

When in this state, the Link layer receives an encoded character sequence from the Phy layer, decodes it into a Dword, and passes the Dword to the Transport layer. The Dword is also entered into the CRC calculation. When in this state the Link layer either transmits a R_IP primitive to signal transmission to continue or transmits a DMAT primitive to signal the transmitter to terminate the transmission.

**Transition LR3:1:** When the Transport layer indicates that space is available in its FIFO, the Link layer shall make a transition to the LR3: L_RcvData state.

**Transition LR3:2:** When the Transport layer indicates that sufficient space is not available in its FIFO, the Link layer shall make a transition to the LR4: L_Hold state.

**Transition LR3:3:** When the Link layer receives a HOLD primitive from the Phy layer, the Link layer shall make a transition to the LR5: L_SendHold state.

**Transition LR3:4:** When the Link layer receives an EOF primitive from the Phy layer, the Link layer shall make a transition to the LR6: L_RcvEOF state.

**Transition LR3:5:** When the Link layer receives a WTRM primitive from the Phy layer, the Link layer shall make a transition to the LR8: L_BadEnd state.

**Transition LR3:6:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall notify the Transport layer that reception was aborted and shall make a transition to the L1: L_IDLE state.

**Transition LR3:7:** When the Link layer receives any Dword other than a HOLD, HOLDA, EOF, or SYNC primitive from the Phy layer, the Link layer shall make a transition to the LR3: L_RcvData state.

**Transition LR3:8:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**Transition LR3:9:** When the host Link layer receives notification from the host Transport layer that a Control Register Frame is pending for transmission (typically due to SRST being toggled), a transition to the L_IDLE state shall be made. The return to the L_IDLE state results in the transmission of SYNC primitives, which causes the transmitting Link layer to also transition to the L_IDLE state. The Control Register FIS may then be transmitted as a Register FIS from host to device.

**LR4: L_Hold state:** This state is entered when the Transport layer indicates that sufficient space is not available in its receive FIFO.

When in this state, the Link layer shall transmit the HOLD primitive and may receive an encoded character from the Phy layer.

**Transition LR4:1:** When the Link layer receives any Dword other than a HOLD primitive from the Phy layer and the Transport layer indicates that sufficient space is now available in its receive FIFO, the Link layer shall make a transition to the LR3: L_RcvData state.

**Transition LR4:2:** When the Link layer receives a HOLD primitive from the Phy layer and the Transport layer indicates that space is now available in its FIFO, the Link layer shall make a transition to the LR5: L_SendHold state.

**Transition LR4:3:** When the Link layer receives a EOF primitive from the Phy layer, the Link layer shall make a transition to the LR6: L_RcvEOF state. Note that due to pipeline latency, an EOF may be received when in the L_Hold state in which case the receiving Link shall use its FIFO headroom to receive the EOF and close the frame reception.

**Transition LR4:4:** When the Transport layer indicates that there is not sufficient space available in its FIFO and the Phy layer is ready, the Link layer shall make a transition to the LR4: L_Hold state.

**Transition LR4:5:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**Transition LR4:6:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall notify the Transport layer of the illegal transition error condition and shall make a transition to the L1: L_IDLE state.

**Transition LR4:7:** When the host Link layer receives notification from the host Transport layer that a Control Register Frame is pending for transmission (typically due to SRST being toggled), a transition to the L_IDLE state shall be made. The return to the L_IDLE state results in the transmission of SYNC primitives, which causes the transmitting Link layer to also transition to the L_IDLE state. The Control Register FIS may then be transmitted as a Register FIS from host to device.

**LR5: L_SendHold state:** This state is entered when a HOLD primitive has been received from the Phy layer.

When in this state, the Link layer shall either transmit the HOLDA primitive to signal transmission to proceed when the transmitter becomes ready or transmit a DMAT primitive to signal the transmitter to terminate the transmission.

**Transition LR5:1:** When the Link layer receives any Dword other than a HOLD or SYNC primitive from the Phy layer, the Link layer shall make a transition to the LR3: L_RcvData state.

**Transition LR5:2:** When the Link layer receives a HOLD primitive from the Phy layer, the Link layer shall make a transition to the LR5: L_SendHold state.

**Transition LR5:3:** When the Link layer receives a EOF primitive from the Phy layer, the Link layer shall make a transition to the LR6: L_RcvEOF state.

**Transition LR5:4:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall make a transition to the L1: L_IDLE state. The Transport layer shall be notified of the illegal transition error condition.

**Transition LR5:5:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**Transition LR5:6:** When the host Link layer receives notification from the host Transport layer that a Control Register Frame is pending for transmission (typically due to SRST being toggled), a transition to the L_IDLE state shall be made. The return to the L_IDLE state results in the transmission of SYNC primitives, which causes the transmitting Link layer to also transition to the L_IDLE state. The Control Register FIS may then be transmitted as a Register FIS from host to device.

**LR6: L_RcvEOF state:** This state is entered when the Link layer has received an EOF primitive from the Phy layer.

When in this state, the Link layer shall check the calculated CRC for the frame and transmit one or more R_IP primitives.

**Transition LR6:1:** If the CRC calculation and check is not yet completed, the Link layer shall make a transition to the LR6: L_RcvEOF state.

**Transition LR6:2:** When the CRC indicates no error, the Link layer shall notify the Transport layer and make a transition to the LR7: L_GoodCRC state.

**Transition LR6:3:** When the CRC indicates an error has occurred, the Link layer shall notify the Transport layer and make a transition to the LR8: L_BadEnd state.

**Transition LR6:4:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**LR7: L_GoodCRC state:** This state is entered when the CRC for the frame has been checked and determined to be good.

When in this state, the Link layer shall wait for the Transport Layer to check the frame and transmit one or more R_IP primitives.

**Transition LR7:1:** When the Transport Layer indicates a good result, the Link Layer shall transition to the LR8: L_GoodEnd state.

**Transition LR7:2:** When the Transport Layer indicates an unrecognized FIS, the Link Layer shall transition to the LR9: L_BadEnd state.

**Transition LR7:3:** If the Transport Layer has not supplied status, then the Link Layer shall transition to the LR7: L_GoodCRC state.

**Transition LR7:4:** When the Link layer detects that the Phy layer is not ready, the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**Transition LR7:5:** When the Transport layer or Link layer indicates an error was encountered during the reception of the recognized FIS, the Link layer shall transition to the LR9: L_BadEnd state.

**Transition LR7:6:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall notify the Transport layer of the illegal transition error condition and shall make a transition to the L1: L_IDLE state.

**LR8: L_GoodEnd state:** This state is entered when the CRC for the frame has been checked and determined to be good.

When in this state, the Link layer shall transmit the R_OK primitive.

**Transition LR8:1:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall make a transition to the L1: L_IDLE state.

**Transition LR8:2:** When the Link layer receives any Dword other than a SYNC primitive from the Phy layer, the Link layer shall make a transition to the LR7: L_GoodEnd state.

**Transition LR8:3:** When the Link layer detects that the Phy layer is not ready, the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

**LR9: L_BadEnd state:** This state is entered when the CRC for the frame has been checked and determined to be bad or when the Transport layer has notified the Link layer that the received FIS is invalid.

When in this state, the Link layer shall transmit the R_ERR primitive.

**Transition LR9:1:** When the Link layer receives a SYNC primitive from the Phy layer, the Link layer shall make a transition to the L1: L_IDLE state.

**Transition LR9:2:** When the Link layer receives any Dword other than a SYNC primitive from the Phy layer, the Link layer shall make a transition to the LR8:BadEnd state.

**Transition LR9:3:** When the Link layer detects that the Phy layer is not ready the Link layer shall notify the Transport layer of the condition and make a transition to the LS1: L_NoCommErr state.

### 7.6.1.5 Link power mode state diagram

| LPM1: L_TPMPartial | Transmit $PMREQ\_P_P$ | |
|---|---|---|
| 1. $PMACK_P$ received from Phy | → | L_ChkPhyRdy |
| 2. $X\_RDY_P$ received from Phy | → | L_RcvWaitFifo[1] |
| 3. $SYNC_P$ or $R\_OK_P$ received from Phy | → | L_TPMPartial |
| 4. AnyDword other than ($PMACK_P$ or $X\_RDY_P$ or $SYNC_P$ or $R\_OK_P$ or $PMREQ\_P_p$[3] or $PMREQ\_S_p$[3])[1] received from Phy layer | → | L_IDLE |
| 5. $PMREQ\_P_p$ or $PMREQ\_S_p$ received from Phy | → | L_TPMPartial[3] |
| 6. PhyNRdy | → | L_NoCommErr[2] |

NOTE –
1. This transition aborts the request from the Transport layer to enter a power mode. A status indication to the Transport layer of this event is required
2. This is an unexpected transition and constitutes an error condition. An error condition needs to be sent to the Transport layer as a result.
3. If $PMREQ\_P_p$ or $PMREQ\_S_p$ is received, the host shall make a transition to the L_IDLE state, but the device shall make a transition to the L_TPMPartial state.

| LPM2: L_TPMSlumber | Transmit $PMREQ\_S_P$ | |
|---|---|---|
| 1. $PMACK_P$ received from Phy | → | L_ChkPhyRdy |
| 2. $X\_RDY_P$ received from Phy | → | L_RcvWaitFifo[1] |
| 3. $SYNC_P$ or $R\_OK_P$ received from Phy | → | L_TPMSlumber |
| 4. AnyDword other than ($PMACK_P$ or $X\_RDY_P$ or $SYNC_P$ or $R\_OK_P$ or $PMREQ\_P_p$[3] or $PMREQ\_S_p$[3])[1] received from Phy layer | → | L_IDLE |
| 5. $PMREQ\_P_p$ or $PMREQ\_S_p$ received from Phy | → | L_TPMSlumber[3] |
| 6. PhyNRdy | → | L_NoCommErr[2] |

NOTE –
1. This transition aborts the request from the Transport layer to enter a power mode. A status indication to the Transport layer of this event is required
2. This is an unexpected transition and constitutes an error condition. An error condition needs to be sent to the Transport layer as a result.
3. If $PMREQ\_P_p$ or $PMREQ\_S_p$ is received, the host shall make a transition to the L_IDLE state, but the device shall make a transition to the L_TPMSlumber state.

| LPM3: L_PMOff | Transmit $PMACK_P$[1] | |
|---|---|---|
| 1. 4 PMACK primitives sent | → | L_ChkPhyRdy |
| 2. Less than four PMACK primitives sent | → | L_PMOff |

NOTE –
1. A flag is set according to whether a $PMREQ\_P_P$ or $PMREQ\_S_P$ was received from the Phy layer

| LPM4: L_PMDeny | Transmit $PMNACK_P$ | | |
|---|---|---|---|
| | 1. $PMREQ\_P_P$ or $PMREQ\_S_P$ received from Phy | → | L_PMDeny |
| | 2. AnyDword other than ($PMREQ\_P_P$ or $PMREQ\_S_P$) received from Phy layer | → | L_IDLE |
| | 3. PhyNRdy | → | L_NoCommErr |

| LPM5: L_ChkPhyRdy | Assert Partial/Slumber to Phy (as appropriate) | | |
|---|---|---|---|
| | 1. PhyRdy | → | L_ChkPhyRdy |
| | 2. PhyNRdy | → | L_NoCommPower |

| LPM6: L_NoCommPower | Maintain Partial/Slumber assertion (as appropriate) | | |
|---|---|---|---|
| | 1. Transport layer requests a wakeup or COMWAKE detected | → | L_WakeUp1 |
| | 2. Transport layer not requesting wakeup and COMWAKE not detected | → | L_NoCommPower |

| LPM7: L_WakeUp1 | Deassert both Partial and Slumber | | |
|---|---|---|---|
| | 1. PhyRdy | → | L_WakeUp2 |
| | 2. PhyNRdy | → | L_WakeUp1 |

| LPM8: L_WakeUp2 | Transmit $ALIGN_P$ | | |
|---|---|---|---|
| | 1. PhyRdy | → | L_IDLE |
| | 2. PhyNRdy | → | L_NoCommErr |

**LPM1: L_TPMPartial state:** This state is entered when the Transport layer has indicated that a transition to the Partial power state is desired.

**Transition LPM1:1:** When in this state a PMREQ_P primitive shall be transmitted. When the Link layer receives a PMACK primitive a transition to the LPM5: L_ChkPhyRdy state shall be made.

**Transition LPM1:2:** If the Link layer receives an X_RDY primitive a transition shall be made to the LR2: L_RcvWaitFifo state, effectively aborting the request to a power mode state.

**Transition LPM1:3:** If the Link layer receives a SYNC or R_OK primitive, then it is assumed that the opposite side has not yet processed the PMREQ_P primitive yet and time is needed. A transition to the LPM1: L_TPMPartial state shall be made.

**Transition LPM1:4:** If the host Link layer receives any Dword from the Phy layer other than a PMACK, X_RDY, SYNC or R_OK primitive, then the request to enter the partial state is aborted and a transition to L1: L_IDLE shall be made. This transition includes the case where a PMNACK is received. The device link layer shall not make this transition if it receives a PMREQ_P or PMREQ_S primitive while in this state.

**Transition LPM1:5:** The host Link layer shall not make this transition as it applies only to the device Link layer. If the device Link layer receives a PMREQ_P or PMREQ_S primitive from the host, it shall remain in this state by transitioning back to LPM1: L_TPMPartial.

**Transition LPM1:6:** If the Link layer detects that the Phy layer has become not ready, this is interpreted as an error condition. The Transport layer shall be notified of the condition and a transition shall be made to the LS1: L_NoCommErr state.

**LPM2: L_TPMSlumber state:** This state is entered when the Transport layer has indicated that a transition to the Slumber power state is desired.

**Transition LPM2:1:** When in this state a PMREQ_S primitive shall be transmitted. When the Link layer receives a PMACK primitive, a transition to the LPM5: L_ChkPhyRdy state shall be made.

**Transition LPM2:2:** If the Link layer receives an X_RDY primitive, a transition to the LR2: L_RcvWaitFifo state shall be made, effectively aborting the request to a power mode state.

**Transition LPM2:3:** If the Link layer receives a SYNC or R_OK primitive, then it is assumed that the opposite side has not yet processed the PMREQ_S primitive yet and time is needed. The transition to the LPM2: L_TPMSlumber state shall be made.

**Transition LPM2:4:** If the host Link layer receives any Dword from the Phy layer other than a PMACK, X_RDY, SYNC, or R_OK primitive, then the request to enter the slumber state is aborted and a transition to L1: L_IDLE shall be made. This transition includes the case where a PMNACK is received.. The device link layer shall not make this transition if it receives a PMREQ_P or PMREQ_S primitive while in this state.

**Transition LPM2:5:** The host Link layer shall not make this transition as it applies only to the device Link layer. If the device Link layer receives a PMREQ_P or PMREQ_S primitive from the host, it shall remain in this state by transitioning back to LPM2: L_TPMSlumber.

**Transition LPM2:6:** If the Link layer detects that the Phy layer has become not ready, this is interpreted as an error condition. The Transport layer shall be notified of the condition and a transition shall be made to the L_NoCommErr state.

**LPM3: L_PMOff state:** This state is entered when either a PMREQ_S or PMREQ_P primitive was received by the Link layer. The Link layer transmits a PMACK primitive for each execution of this state.

**Transition LPM3:1:** If four PMACK primitives have been transmitted, a transition shall be made to the L_ChkPhyRdy state.

**Transition LPM3:2:** If less than four PMACK primitives have been transmitted, a transition shall be made to L_PMOff state.

**LPM4: L_PMDeny state:** This state is entered when any primitive is received by the Link layer to enter a power mode and power modes are currently disabled. The Link layer shall transmit a PMNACK primitive to inform the opposite end that a power mode is not allowed.

**Transition LPM4:1:** If the Link layer continues to receive a request to enter any power mode than a transition back to the same LPM4: L_PMDeny state shall be made.

**Transition LPM4:2:** If the Link layer receives any Dword other than a power mode request primitive, then the Link layer assumes that the power mode request has been removed and shall make a transition to the L1: L_IDLE state.

**Transition LPM4:3:** If the Link layer detects that the Phy layer has become not ready, this is interpreted as an error condition. The Transport layer shall be notified of the condition and a transition shall be made to the LS1: L_NoCommErr state.

**LPM5: L_ChkPhyRdy state:** This state is entered whenever it is desired for the Phy layer to enter a low power condition. For each execution in this state a request is made to the Phy layer to enter the state and deactivate the PhyRdy signal. Partial or Slumber is asserted to the Phy as appropriate.

**Transition LPM6:1:** If the Phy layer has not yet processed the request to enter the power saving state and not deactivated the PhyRdy signal, then the Link layer shall remain in the LPM5: L_ChkPhyRdy state and continue to request the Phy layer to enter the power mode state.

**Transition LPM6:2:** When the Phy layer has processed the power mode request and has deactivated the PhyRdy signal, then a transition shall be made to the LPM6: L_NoCommPower state.

**LPM6: L_NoCommPower state:** This state is entered when the Phy has deasserted its PhyRdy signal indicating that it is in either Partial or Slumber state. In this state, the Link layer waits for the out of band detector to signal reception of the COMWAKE signal ( for a wakeup initiated by the other device ), or for the Transport layer to request a wakeup.

**Transition LPM6:1**: If the Transport layer requests a wakeup or the out of band signal detector indicates reception of the COMWAKE signal, then a transition shall be made to LPM7: L_WakeUp1

**Transition LPM6:2**: If the Transport layer does not request a wakeup and the out of band detector does not indicate reception of the COMWAKE signal, then a transition shall be made to LPM6: L_NoCommPower.

**LPM7: L_WakeUp1 state:** This state is entered when the Transport layer has initiated a wakeup. In this state, the Link layer shall deassert both Partial and Slumber to the Phy, and wait for the PhyRdy signal from the Phy to be asserted. While in this state the Phy is performing the wakeup sequence.

**Transition LPM7:1** When the Phy asserts its PhyRdy signal, a transition shall be made to LPM8: L_WakeUp2.

**Transition LPM7:2:** When the Phy remains not ready, a transition shall be made to LPM7: L_WakeUp1.

**LPM8: L_WakeUp2 state:** This state is entered when the Phy has acknowledged an initiated wakeup request by asserting its PhyRdy signal. In this state, the Link layer shall transmit the ALIGN sequence, and transition to the L1: L_IDLE state.

**Transition LPM8:1** If the Phy keeps PhyRdy asserted, a transition shall be made to the L1: L_IDLE state.

**Transition LPM8:2** If the Phy deasserts PhyRdy, this is an error condition. The Transport layer shall be notified of the condition and a transition shall be made to the LS1: L_NoCommErr state.

# 8 Transport layer

## 8.1 Transport layer overview

The Transport layer need not be cognizant of how frames are transmitted and received. The Transport layer simply constructs Frame Information Structures (FIS's) for transmission and decomposes received Frame Information Structures. Host and device Transport layer state differ in that the source of the FIS content differs. The Transport layer maintains no context in terms of ATA commands or previous FIS content.

### 8.1.1 FIS construction

When requested to construct an FIS by a higher layer, the Transport layer provides the following services:

- Gathers FIS content based on the type of FIS requested.
- Places FIS content in the proper order.
- Notifies the Link layer of required frame transmission and passes FIS content to Link.
- Manages Buffer/FIFO flow, notifies Link of required flow control.
- Receives frame receipt acknowledge from Link layer.
- Reports good transmission or errors to requesting higher layer.

### 8.1.2 FIS decomposition

When an FIS is received from the Link layer, the Transport layer provides the following services:

- Receives the FIS from the Link layer.
- Determines FIS type.
- Distributes the FIS content to the locations indicated by the FIS type.
- For the host Transport layer, receipt of an FIS may also cause the construction of an FIS to be returned to the device.
- Reports good reception or errors to higher layer

## 8.2 Frame Information Structure ( FIS)

## 8.3 Overview

A frame is a group of Dwords that convey information between host and device as described previously. Primitives are used to define the boundaries of the frame and may be inserted to control the rate of the information flow. This section will focus on and describe the actual information content of the frame - hereto referred as payload - and assumes the reader is aware of the Primitives that are needed to support the information content.

The contents of the info field is divided into three categories: (1) register type, (2) setup type, and (3) data type. For each category the organization of each frame is defined in the following section.

## 8.4 Payload content

The type and layout of the payload is indicated by the Frame Information Type field located in byte 0 of the first Dword of the payload. See Figure 59 – Register - Host to Device FIS layout as an example. This example type is used primarily to transfer the contents of the Shadow Register Block Registers from the host to the device. Table 26 can be referenced to refresh the reader's memory of a simplified version of the Shadow Register Block organization of a standard ATA adapter.

**Table 26. – Simplified Shadow Register Block register numbering**

| | | | | Register access operation | |
|---|---|---|---|---|---|
| | **A2** | **A1** | **A0** | **Read** | **Write** |
| CS 0 Active | 0 | 0 | 0 | Data Port | |
| | 0 | 0 | 1 | Error | Features |
| | 0 | 1 | 0 | Sector Count [15:8], [7:0] | |
| | 0 | 1 | 1 | Sector Number [31:24], [7:0] | |
| | 1 | 0 | 0 | Cylinder Low [39:32], [15:8] | |
| | 1 | 0 | 1 | Cylinder High [47:40], [23:16] | |
| | 1 | 1 | 0 | Device / Head | |
| | 1 | 1 | 1 | Status | Command |
| | | | | | |
| CS 1 active | 1 | 1 | 0 | Alternate Status | Device Control |

The following sections will detail the types of payloads that are possible. The SOF, EOF and HOLD primitives have been removed for clarity.

## 8.5 FIS types

The following sections define the structure of each individual FIS.

### 8.5.1 All FIS types

In all of the following FIS structures the following rules shall apply:

1. All reserved fields shall be written or transmitted as all 0's
2. All reserved fields shall be ignored during the reading or reception process.

### 8.5.2 Register - Host to Device

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | Features | Command | C | R | R | Reserved (0) | FIS Type (27h) |
| 1 | Dev / Head | Cyl High | Cyl Low | | | | Sector Number |
| 2 | Features (exp) | Cyl High (exp) | Cyl Low (exp) | | | | Sector Num (exp) |
| 3 | Control | Reserved (0) | Sector Count (exp) | | | | Sector Count |
| 4 | Reserved (0) | Reserved (0) | Reserved (0) | | | | Reserved (0) |

**Figure 59 – Register - Host to Device FIS layout**

Field Definitions

FIS Type - Set to a value of 27h. Defines the rest of the FIS fields. Defines the length of the FIS as five Dwords.

C - This bit is set to one when the register transfer is due to an update of the Command register. The bit is set to zero when the register transfer is due to an update of the Device Control register.

Command - Contains the contents of the Command register of the Shadow Register Block.

Sector Number - Contains the contents of the Sector Number register of the Shadow Register Block.

Control - Contains the contents of the Device Control register of the Shadow Register Block.

Cyl Low - Contains the contents of the Cylinder Low register of the Shadow Register Block.

Cyl Low (exp) – Contains the contents of the expanded address field of the Shadow Register Block

Cyl High - Contains the contents of the Cylinder High register of the Shadow Register Block.

Cyl High (exp) – Contains the contents of the expanded address field of the Shadow Register Block

Dev / Head - Contains the contents of the Device / Head register of the Shadow Register Block.

Features - Contains the contents of the Features register of the Shadow Register Block.

Features (exp) – Contains the contents of the expanded address field of the Shadow Register Block

R – Reserved bits – shall be set to 0.

Sector Count - Contains the contents of the Sector Count register of the Shadow Register Block.

Sector Count (exp) – Contains the contents of the expanded address field of the Shadow Register Block

Sector Number - Contains the contents of the Sector Number register of the Shadow Register Block.

Sector Num (exp) – Contains the contents of the expanded address field of the Shadow Register Block

#### 8.5.2.1 Description

The Register – Host to Device FIS is used to transfer the contents of the Shadow Register Block from the host to the device. This is the mechanism for issuing legacy ATA commands to the device.

#### 8.5.2.2 Transmission

Transmission of a Register – Host to Device FIS is initiated by a write operation to either the command register, or a write to the Control register with a value different than is currently in the Control register * in the host adapter's Shadow Register Block. Upon initiating transmission, the current contents of the Shadow Register Block are transmitted and the C bit in the FIS is set according to whether the transmission was a result of the Command register being written or the Control register being written. The host adapter shall set the BSY bit in the shadow Status register to one within 400 ns of the write operation to the Command register that initiated the transmission. The host adapter shall set the BSY bit in the shadow Status register to one within 400 ns of a write operation to the Control register if the write to the Control register changes the state of the SRST bit from 0 to 1 but shall not set the BSY bit in the shadow Status register for writes to the Control register that do not change the state of the SRST bit from 0 to 1.

It is important to note that Serial ATA host adapters enforce the same access control to the Shadow Register Block as legacy (parallel) ATA devices enforce to the Command Block Registers. Specifically, the host is prohibited from writing the Features, Sector Count, Sector Number, Cylinder Low, Cylinder High, or Device/Head registers when either BSY or DRQ is set in the Status Register. Any write to the Command Register when BSY or DRQ is set is ignored unless the write is to issue a Device Reset command.

#### 8.5.2.3 Reception

Upon reception of a valid Register - Host to Device FIS the device updates its local copy of the Command and Control Block Register contents and either initiates execution of the command indicated in the command register or initiates execution of the control request indicated in the Control register, depending on the state of the C bit in the FIS.

* There are legacy BIOS and drivers that will write the Control register to enable the interrupt just prior to issuing a command. To avoid unnecessary overhead, this FIS is transmitted to the device only upon a change of state from the previous value.

### 8.5.3 Register - Device to Host

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | Error | Status | R | I | R | Reserved (0) | FIS Type (34h) |
| 1 | Dev / Head | Cyl High | Cyl Low | | | | Sector Number |
| 2 | Reserved (0) | Cyl High (exp) | Cyl Low (exp) | | | | Sector Num (exp) (0) |
| 3 | Reserved (0) | Reserved (0) | Sector Count (exp) | | | | Sector Count |
| 4 | Reserved (0) | Reserved (0) | Reserved (0) | | | | Reserved (0) |

**Figure 60 – Register - Device to Host FIS layout**

Field Definitions

FIS Type - Set to a value of 34h. Defines the rest of the FIS fields. Defines the length of the FIS as five Dwords.

Cyl Low - Contains the new value of the cylinder low register of the Shadow Register Block.

Cyl Low (exp) – Contains the contents of the expanded address field of the Shadow Register Block

Cyl High - Contains the new value of the cylinder high register of the Shadow Register Block.

Cyl High (exp) – Contains the contents of the expanded address field of the Shadow Register Block

Dev / Head - Contains the new value of the device / head register of the Shadow Register Block.

Error - Contains the new value of the Error register of the Shadow Register Block.

I - Interrupt bit. This bit reflects the interrupt bit line of the device

R - Reserved (0)

Sector Count - Contains the new value of the sector count register of the Shadow Register Block.

Sector Count (exp) – Contains the contents of the expanded address field of the Shadow Register Block

Sector Number - Contains the new value of the sector number register of the Shadow Register Block.

Sector Num (exp) – Contains the contents of the expanded address field of the Shadow Register Block

Status - Contains the new value of the Status (and Alternate status) register of the Shadow Register Block.

#### 8.5.3.1 Description

The Register – Device to Host FIS is used to by the device to update the contents of the host adapter's Shadow Register Block. This is the mechanism by which devices indicate command completion status or otherwise change the contents of the host adapter's Shadow Register Block.

#### 8.5.3.2 Transmission

Transmission of a Register - Device to Host FIS is initiated by the device in order to update the contents of the host adapter's Shadow Register Block. Transmission of the Register - Device to Host FIS is typically as a result of command completion by the device.

The Register - Device to Host FIS should not be used to set the SERV bit in the Status Register to request service for a bus released command; the Set Device Bits - Device to Host FIS should be used instead. The SERV bit transmitted with the Register - Device to Host FIS will be written to the shadow Status Register and so the bit should accurately reflect the state of pending service requests when the FIS is transmitted as a result of a command completion by the device.

#### 8.5.3.3 Reception

Upon reception of a valid Register - Device to Host FIS the received register contents are transferred to the host adapter's Shadow Register Block.

If the BSY bit and DRQ bit in the shadow Status Register are both cleared when a Register - Device to Host FIS is received by the host adapter, then the host adapter shall discard the contents of the received FIS and not update the contents of any shadow register.

### 8.5.4 Set Device Bits - Device to Host

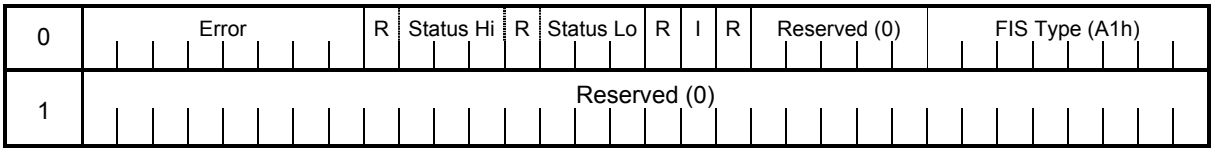



**Figure 61 - Set Device Bit - Device to Host FIS layout**

Field Definitions

FIS Type – Set to a value of A1h. Defines the rest of the FIS fields. Defines the length of the FIS as two Dwords.

I – Interrupt Bit. This bit signals the host adapter to enter an interrupt pending state if both the BSY bit and the DRQ bit in the shadow Status register are zero when the frame is received.

Error – Contains the new value of the Error register of the Shadow Register Block.

Status-Hi – Contains the new value of bits 6, 5, and 4 of the Status register of the Shadow Register Block.

Status-Lo – Contains the new value of bits 2,1, and 0 of the Status register of the Shadow Register Block.

R – Reserved (0)

#### 8.5.4.1 Description

The Set Device Bits - Device to Host FIS is used by the device to load Shadow Register Block bits for which the device has exclusive write access. These bits are the eight bits of the Error register and six of the eight bits of the Status register. This FIS does not alter bit 7, BSY, or bit 3, DRQ, of the Status register.

The FIS includes a bit to signal the host adapter to generate an interrupt if the BSY bit and the DRQ bit in the shadow Status Register are both cleared to zero when this FIS is received.

Some Serial ATA to parallel ATA bridge solutions may elect to not support this FIS based on the requirements of their target markets. Upon reception, such devices will process this FIS as if it were an invalid FIS type and return the R_ERR end of frame handshake.

#### 8.5.4.2 Transmission

The device transmits a Set Device Bits - Device to Host to alter one or more bits in the Error register or in the Status register in the Shadow Register Block. This FIS should be used by the device to set the SERV bit in the Status register to request service for a bus released command. When used for this purpose the device shall set the Interrupt bit to one.

#### 8.5.4.3 Reception

Upon receiving a Set Device Bits - Device to Host, the host adapter shall load the data from the Error field into the shadow Error register, the data from the Status-Hi field into bits 6, 5, and 4, of the shadow Status register, and the data from the Status-Lo field into bits 2, 1, and 0 of the shadow Status register. Bit 7, BSY, and bit 3, DRQ, of the shadow Status register shall not be changed. If the I bit in the FIS is set to a one, and if both the BSY bit and the DRQ bit in the Shadow status register are cleared to zero when this FIS is received, then the host adapter shall enter an interrupt pending state.

### 8.5.5 DMA Activate - Device to Host

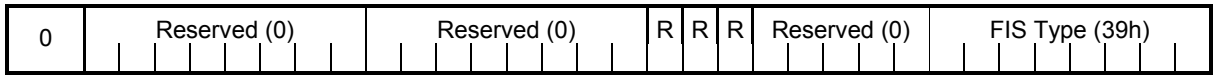



**Figure 62 – DMA Activate - Device to Host FIS layout**

Field Definitions

FIS Type - Set to a value of 39h. Defines the rest of the FIS fields. Defines the length of the FIS as one Dword.

R - Reserved (0)

#### 8.5.5.1 Description

The DMA Activate – Device to Host FIS is used by the device to signal the host to proceed with a DMA data transfer of data from the host to the device. This is the mechanism by which a legacy device signals its readiness to receive DMA data from the host.

A situation may arise where the host needs to send multiple Data FIS's in order to complete the overall data transfer request. The host shall wait for a successful reception of a DMA Activate FIS before sending each of the Data FIS's that are needed.

#### 8.5.5.2 Transmission

The device transmits a DMA Activate – Device to Host to the host in order to initiate the flow of DMA data from the host to the device as part of the data transfer portion of a corresponding DMA write command. When transmitting this FIS, the device shall be prepared to subsequently receive a Data - Host to Device FIS from the host with the DMA data for the corresponding command.

#### 8.5.5.3 Reception

Upon receiving a DMA Activate – Device to Host, if the host adapter's DMA controller has been programmed and armed, the host adapter shall initiate the transmission of a Data FIS and shall transmit in this FIS the data corresponding to the host memory regions indicated by the DMA controller's context. If the host adapter's DMA controller has not yet been programmed and armed, the host adapter shall set an internal state indicating that the DMA controller has been activated by the device, and as soon as the DMA controller has been programmed and armed, a Data FIS shall be transmitted to the device with the data corresponding to the host memory regions indicated by the DMA controller context.

### 8.5.6 DMA Setup – Device to Host or Host to Device (Bidirectional)

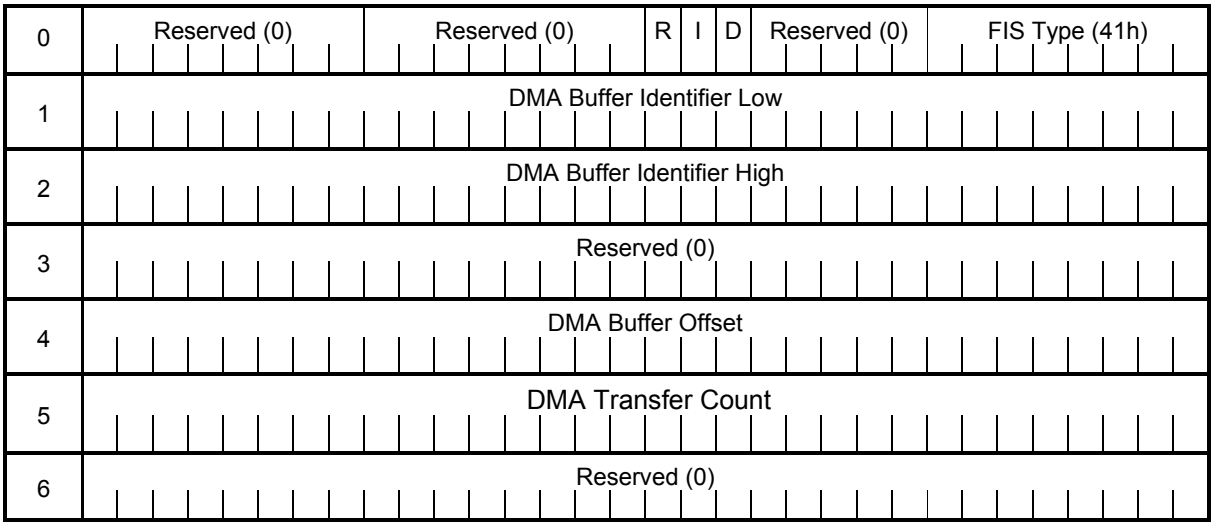



**Figure 63 –DMA Setup – Device to Host FIS layout**

Field Definitions

FIS Type - Set to a value of 41h. Defines the rest of the FIS fields. Defines the total length of the FIS as seven Dwords.

D - Indicates whether subsequent data transferred after this FIS is from transmiter to receiver or from receiver to transmitter 1 = transmitter to receiver, 0 = receiver to transmitter.

DMA Buffer Identifier Low/High - This field is used to identify a DMA buffer region in host memory. The contents are not described in this specification and are host dependent. The buffer identifier is supplied by the host to the device and the device echoes it back to the host. This allows the implementation to pass a physical address or or, in more complex implementations, the buffer identifier could be a scatter gather list or other information that can identify a DMA “channel”.

DMA Buffer Offset - This is the byte offset into the buffer. Bits <1:0> must be zero

DMA Transfer Count: This is the number of bytes that will be read or written by the device. Bit zero must be zero.

I Interrupt - If the Interrupt bit is set to 1 an interrupt pending shall be generated when the DMA transfer count is exhausted.

R - Reserved (0)

#### 8.5.6.1 Description

The DMA Setup – Device to Host or Host to Device FIS is the mechanism by which first-party DMA access to host memory is initiated. This FIS is used to request the host or device to program its DMA controller before transferring data. The FIS allows the actual host memory regions to be abstracted (depending on implementation) by having memory regions referenced via a base memory descriptor representing a memory region that the host has granted the device access to. The specific implementation for the memory descriptor abstraction is not defined.

The device or host is informed of the 64-bit DMA buffer identifier/descriptor at some previous time by an implementation specific mechanism such as a command issued to or as defined in a specification. Random access within a buffer is accomplished by using the buffer offset.

First party DMA is a superset capability not necessarily supported by legacy devices or legacy device drivers but essential for accommodating future capabilities.

#### 8.5.6.2 Transmission

A device or host transmits a DMA Setup – Device to Host or Host to Device FIS as the first step in performing a DMA access. The purpose of the DMA Setup – Device to Host or Host to Device is to establish DMA hardware context for one or more data transfers.

A DMA Setup – Device to Host or Host to Device is required only when the DMA context is to be changed. Multiple Data – Host to Device or Device to Host FIS's can follow in either direction, for example, if the transfer count exceeds the maximum Data – Host to Device or Device to Host transfer length or when a data transfer is interrupted. When multiple Data – Host to Device or Device to Host FIS's follow a DMA Setup – Device to Host or Host to Device FIS, the device or host shall place the data contained in the FIS in sequential addresses; that is, if the last Dword of an FIS is placed in (or obtained from) address N, the first Dword of a subsequent Data – Host to Device or Device to Host shall be placed in (or obtained from) address N+4 unless an intervening DMA Setup – Device to Host or Host to Device FIS is used to alter the DMA context. This mechanism allows for the efficient streaming of data into a buffer.

#### 8.5.6.3 Reception

Upon receiving a DMA Setup – Device to Host or Host to Device FIS, the receiver of the FIS shall validate the received DMA Setup request, and provided that the buffer identifier and the specified offset/count are valid, program and arm the adapter's DMA controller using the information in the FIS. The specific implementation of the buffer identifier and buffer/address validation is not specified. After a valid DMA Setup – Device to Host or Host to Device FIS with the D bit set to 0, the receiver of the DMA Setup – Device to Host or Host to Device FIS responds with one or more Data – Host to Device or Device to Host FIS's until the DMA count is exhausted. After a valid DMA Setup – Device to Host or Host to Device FIS with the D bit set to 1, the receiver of the FIS must be prepared to accept one or more Data – Host to Device or Device to Host FIS's until the DMA count is exhausted.

An interrupt pending condition shall be generated upon the completion of the DMA transfer if the I bit is set to 1. The definition DMA transfer completion is system dependent but typically includes the exhaustion of the transfer count or the detection of an error by the DMA controller.

NOTE – First-party DMA accesses are categorized in two groups: command/status transfers and user-data transfers. Interrupts would not typically be generated on user-data transfers. The optimal interrupt scheme for command/status transfers is not defined in this specification.

### 8.5.7 BIST Activate - Bidirectional

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Reserved (0) | | | | | | | | Pattern Definition | | | | | | | | R | R | R | Reserved (0) | | | | | | | | FIS Type (58h) | | | | | | | |
| | | | | | | | | | T | A | S | L | F | P | R | V | | | | | | | | | | | | | | | | |
| 1 | Data [31:24] | | | | | | | | Data [23:16] | | | | | | | | Data [15:8] | | | | | | | | Data [7:0] | | | | | | | |
| 2 | Data [31:24] | | | | | | | | Data [23:16] | | | | | | | | Data [15:8] | | | | | | | | Data [7:0] | | | | | | | |

**Figure 64 – BIST Activate - Bidirectional**

Field Definitions

FIS Type - Set to a value of 58h. Defines the rest of the FIS fields.
F – Far End Analog (AFE) Loopback (Optional)
L - Far End Retimed Loopback* Transmitter must insert additional ALIGNS
R - Reserved (0)
T - Far end transmit only mode
A - ALIGN Bypass ( Do not Transmit Align Primitives ) (valid only in combination with T Bit) (optional behavior)
S - Bypass Scrambling (valid only in combination with T Bit) (optional behavior)
P - Primitive bit. (valid only in combination with the T- Bit) (optional behavior)

V - Vendor Unique Test Mode. Causes all other bits to be ignored

#### 8.5.7.1 Description

The BIST Activate FIS shall be used to place the receiver in one of “n” loopback modes.

The BIST Activate FIS is a bi-directional request in that it can be sent by either the host or the device. The sender and receiver have distinct responsibilities in order to insure proper cooperation between the two parties. The state machines for transmission and reception of the FIS are symmetrical.

The state machines for the transmission of the FIS do not attempt to specify the actions the sender takes once successful transmission of the request has been performed. After the Application layer is notified of the successful transmission of the FIS the sender’s Application layer will prepare its own Application, Transport and Physical layers into the appropriate states that support the transmission of a stream of data. The FIS shall not be considered successfully transmitted until the receiver has acknowledged reception of the FIS as per normal FIS transfers documented in various sections of this specification. The transmitter of the BIST Activate FIS should transmit continuous SYNC primitives after reception of the R_OK primitive until such a time that it is ready to interact with the receiver in the BIST exchange.

Similarly, the state machines for the reception of the FIS do not specify the actions of the receiver’s application layer. Once the FIS has been received, the receiver’s application layer must place its own Application, Transport and Physical layers into states that will perform the appropriate retransmission of the sender’s data. The receiver shall not enter the BIST state until after it has properly received a good BIST Activate FIS (good CRC), indicated a successful transfer of the FIS to the transmitting side via the R_OK primitive and has received at least one good SYNC primitive. Once in the self-test mode, a receiver shall continue to allow processing of the COMINIT or COMRESET signals in order to exit from the self-test mode.

**F**: The Far End Analog ( Analog Front End - AFE ) Loopback, shall be defined as a vendor optional mode where the raw data is received, and retransmitted, without any retiming or re-synchronization, etc. The implementation of Far End AFE Loopback is optional due to the round-trip characteristics of the test as well as the lack of retiming. This mode is intended to give a quick indication of connectivity, and test failure is not an indication of system failure.

**L:** The Far End Retimed Loopback, shall be defined as a mode where the receiver will retime the data, and retransmit the retimed data. The initiator of the retimed loopback mode must account for the loopback device consuming up to two ALIGN primitives (one ALIGN sequence) every 256 Dwords transmitted and, if it requires any ALIGN primitives to be present in the returned data stream, it must insert additional ALIGN's in the transmitted stream. The initiator shall transmit additional ALIGN sequences in a single burst at the normal interval of every 256 Dwords transmitted (as opposed to inserting ALIGN sequences at half the interval).

The loopback device may remove zero, one, or two ALIGN primitives from the received data. It may insert one or more ALIGN primitives if they are directly preceded or followed by the initiator inserted ALIGN primitives (resulting in ALIGN sequences consisting of at least two ALIGN primitives) or it may insert two or more ALIGN primitives if not preceded or followed by the initiator's ALIGN primitives. One side effect of the loopback retiming is that the returned data stream may have instances of an odd number of ALIGN primitives, however, returned ALIGN's are always in bursts and if the initiator transmitted dual ALIGN sequences (four consecutive ALIGN's), then the returned data stream shall include ALIGN bursts that are no shorter than two ALIGN primitives long (although the length of the ALIGN burst may be odd). The initiator of the retimed loopback mode shall not assume any relationship between the relative position of the ALIGN's returned by the loopback device and the relative position of the ALIGN's sent by the initiator.

In retimed loopback mode, the initiator shall transmit only valid 8b/10b characters so the loopback device may 10b/8b decode it and re-encode it before retransmission. If the loopback device descrambles incoming data it is responsible for rescrambling it with the same sequence of scrambling syndromes in order to ensure the returned data is unchanged from the received data. The loopback device's running disparity for its transmitter and receiver are not guaranteed to be the same and thus the loopback initiator shall 10b/8b decode the returned data rather than use the raw 10b returned stream for the purpose of data comparison. The loopback device shall return all received data unaltered and shall disregard protocol processing of primitives. Only the OOB signals and ALIGN processing is acted on by the loopback device, while all other data is retransmitted without interpretation.

**T:** The Far-End Transmit Mode shall be defined as a mode that can be used to invoke the Far-End Interface to send data patterns, upon receipt of the FIS BIST Activate, as defined by Pattern DWords #1, and #2. Note that Pattern D-Words #1, and #2 shall be applicable only when the T bit is active, indicating "Far-End Transmit Mode".
This data is modified by the following bits.

Note, that this mode is intended for Inspection/Observation Testing, as well as support for conventional laboratory equipment, rather than for in-system automated testing.

**P**: The transmit primitives bit. When this bit is set in far end transmit mode, the lowest order byte of the two following Dwords will be treated as K Characters. It is the responsibility of the sender of the BIST FIS to ensure that the data in byte 0 of the Dwords are valid D character versions of the K character (i.e. BCh for K28.5). Applicable only when the T bit is set.

**A:** The ALIGN primitive sequence bypass mode. When set to 1, no ALIGN primitives are sent. When the A-bit is not asserted, ALIGN Primitives are sent normally as defined in this document. Applicable only when the T bit is set.

**S:** The Bypass Scrambling shall be defined as a mode that can be used to send data or patterns, during BIST activation, that are not scrambled, however are encoded and decoded to normal and legal 8b/10b values. Applicable only when the T bit is set.

**V**: The vendor unique mode is implementation specific and shall be reserved for individual vendor use. All other bits are ignored in this mode.

#### 8.5.7.2 Transmission

The initiator transmits a BIST Activate to the recipient in order to initiate the BIST mode of operation.

#### 8.5.7.3 Reception

Upon receiving a BIST Activate, the recipient shall begin operations as per the BIST Activate FIS, described in this document above.

### 8.5.8 PIO Setup – Device to Host

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | Error | Status | R | I | D | Reserved (0) | FIS Type (5Fh) |
| 1 | Dev / Head | Cyl High | Cyl Low | | | | Sector Number |
| 2 | Reserved (0) | Cyl High (exp) | Cyl Low (exp) | | | | Sector Num (exp) (0) |
| 3 | E_Status | Reserved (0) | Sector Count (exp) | | | | Sector Count |
| 4 | Reserved (0) | | Transfer Count | | | | |

**Figure 65 – PIO Setup - Device to Host FIS layout**

Field Definitions

FIS Type - Set to a value of 5Fh. Defines the rest of the FIS fields. Defines the length of the FIS as five Dwords.

Cyl Low - Holds the contents of the cylinder low register of the Commanf Block.

Cyl Low (exp) – Contains the contents of the expanded address field of the Shadow Register Block

Cyl High - Holds the contents of the cylinder high register of the Command Block.

Cyl High (exp) – Contains the contents of the expanded address field of the Shadow Register Block

D - Indicates whether host memory is being written or read by the device. 1 = write (device to host), 0 = read (host to device).

Dev / Head - Holds the contents of the device / head register of the Command Block.

Status - Contains the new value of the Status register of the Command Block for initiation of host data transfer.

Error - Contains the new value of the Error register of the Command Block at the conclusion of all subsequent Data to Device frames.

I - Interrupt bit. This bit reflects the interrupt bit line of the device

R - Reserved (0)

Sector Count - Holds the contents of the sector count register of the Command Block.

Sector Count (exp) – Contains the contents of the expanded address field of the Shadow Register Block

Sector Number - Holds the contents of the sector number register of the Command Block.

Sector Num (exp) – Contains the contents of the expanded address field of the Shadow Register Block

E_Status - Contains the new value of the Status register of the Command Block at the conclusion of all subsequent Data frames.

Transfer Count – Holds the number of bytes to be transferred in the subsequent data FIS. The Transfer Count value shall be nonzero and the low order bit shall be zero (even number of bytes transferred).

#### 8.5.8.1 Description

The PIO Setup – Device to Host FIS is used by the device to provide the host adapter with sufficient information regarding a PIO data phase to allow the host adapter to efficiently handle PIO data transfers. For PIO data transfers, the device shall send to the host a PIO Setup – Device to Host FIS just before each and every data transfer FIS that is required to complete the data transfer. Data transfers from Host to Device as well as data transfers from Device to Host shall follow this algorithm. Because of the stringent timing constraints in the ATA standard, the PIO Setup FIS includes both the starting and ending status values. These are used by the host adapter to first signal to host software readiness for PIO write data (BSY deasserted and DRQ asserted), and following the PIO write burst to properly signal host software by deasserting DRQ and possibly raising BSY.

#### 8.5.8.2 Transmission of PIO Setup by Device Prior to a Data Transfer from Host to Device

The device transmits a PIO Setup – Device to Host FIS to the host in preparation for a PIO data payload transfer just before each and every PIO data payload transfer required to complete the total data transfer for a command. The device includes in the FIS the values to be placed in the Status shadow register at the beginning of the PIO data payload transfer and the value to be placed in the Status shadow register at the end of the data payload transfer. The device must be prepared to receive a Data FIS in response to transmitting a PIO Setup FIS.

#### 8.5.8.3 Reception of PIO Setup by Host Prior to a Data Transfer from Host to Device

Upon receiving a PIO Setup – Device to Host FIS, the host shall transfer the Status and Error values into the Status and Error Shadow registers and shall hold the E_Status value in a temporary register. The Transfer Length value shall be loaded into a countdown register. Upon detecting the change in the Status shadow register, host software proceeds to perform a series of write operations to the Data shadow register, which the host adapter shall collect to produce a Data FIS to the device. Each write of the Data shadow register results in another word of data being concatenated into the Data FIS, and the countdown register being decremented accordingly. The E_Status value shall be transferred to the Status shadow register within 400 ns of the countdown register reaching terminal count. In the case that the transfer length represents an odd number of words, the last word shall be placed in the low order (word 0) of the final Dword and the high order word (word 1) of the final Dword shall be padded with zeros before transmission. This process is repeated for each and every data FIS needed to complete the overall data transfer of a command.

#### 8.5.8.4 Transmission of PIO Setup by Device Prior to a Data Transfer from Device to Host

The device transmits a PIO Setup – Device to Host FIS to the host in preparation for a PIO data payload transfer just before each and every PIO data payload transfer required to complete the total data transfer for a command. The device includes in the FIS the values to be placed in the Status shadow register at the beginning of the PIO data payload transfer and the value to be placed in the Status shadow register at the end of the data payload transfer. The device must be prepared to transmit a Data FIS in following the transmittal of a PIO Setup FIS.

#### 8.5.8.5 Reception of PIO Setup by Host Prior to a Data Transfer from Device to Host

Upon receiving a PIO Setup – Device to Host FIS for a device to host transfer, the host shall hold the Status, Error, and E_Status values in temporary registers. The Transfer Length value shall be loaded into a countdown register. Upon reception of a Data FIS from the device, the Status and Error values are loaded into the Status and Error Shadow registers and host software proceeds to perform a series of read operations from the Data shadow register. Each read of the Data shadow register results in a countdown register being decremented accordingly. The E_Status value shall be transferred to the Status shadow register within 400 ns of the countdown register reaching terminal count. This process is repeated for each and every data FIS needed to complete the overall data transfer of a command.

### 8.5.9 Data - Host to Device or Device to Host (Bidirectional)

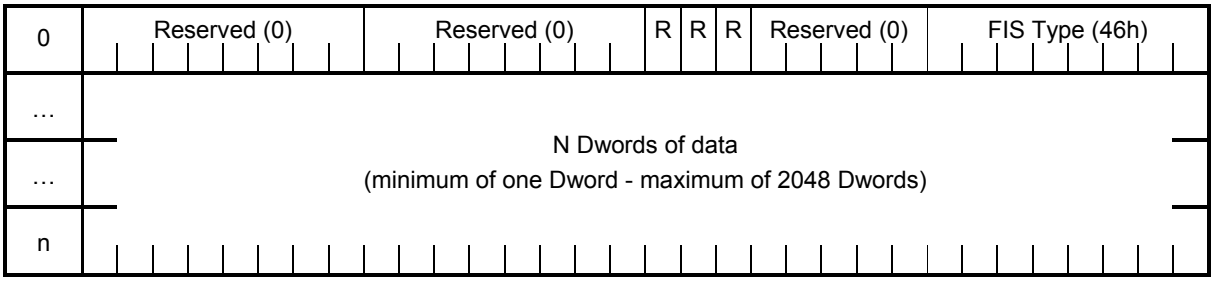



**Figure 66 – Data – Host to Device or Device to Host FIS layout**

Field Definitions

FIS Type - Set to a value of 46h. Defines the rest of the FIS fields. Defines the length of the FIS as n+1 Dwords.

Dwords of data - Contain the actual data to transfer. Only 32 bit fields are transferred. The last Dword is padded with zeros when only a partial Dword is to be transmitted.

NOTE –The maximum amount of user data that can be sent in a single Data – Host to Device or Data – Device to Host FIS is limited. See description.

R – Reserved bit – shall be set to 0.

#### 8.5.9.1 Description

The Data – Host to Device and the Data – Device to Host FIS's are used for transporting payload data, such as the data read from or written to a number of sectors on a hard drive. The FIS may either be generated by the device to transmit data to the host or may be generated by the host to transmit data to the device. This FIS is generally only one element of a sequence of transactions leading up to a data transmission and the transactions leading up to and following the Data FIS establish the proper context for both the host and device.

The byte count of the payload is not an explicit parameter, rather it is inferred by counting the number of Dwords between the SOF and EOF primitives, and discounting the FIS type and CRC Dwords. The payload size shall be no more than 2048 Dwords (8192 bytes) for native implementations. Parallel ATA to Serial ATA bridge implementations should not transmit more than 2048 Dwords (8192 bytes) in any one Data FIS. However, separate bridge hardware is not required to break up PIO or DMA transfers from parallel ATA into more than one Serial ATA Data FIS and may therefore produce payloads longer than 2048 Dwords. Receivers shall tolerate reception of data payloads longer than

2048 Dwords without error. Non-packet devices, with or without bridges, should report a SET MULTIPLE limit of 16 sectors or less in word 47 of their IDENTIFY DEVICE information.

In the case that the transfer length represents an odd number of words, the last word shall be placed in the low order (word 0) of the final Dword and the high order word (word 1) of the final Dword shall be padded with zeros before transmission.

#### 8.5.9.2 Transmission

The device transmits a Data – Device to Host FIS to the host during the data transfer phase of legacy mode PIO reads, DMA reads, and First-party DMA writes to host memory. The device shall precede a Data FIS with any necessary context-setting transactions as appropriate for the particular command sequence. For example, a First-party DMA host memory write must be preceded by a First-party DMA Setup – Device to Host FIS to establish proper context for the Data FIS that follows.

The host transmits a Data – Host to Device FIS to the device during the data transfer phase of PIO writes, DMA writes, and First-party DMA reads of host memory. The FIS shall be preceded with any necessary context-setting transactions as appropriate for the particular command sequence. For example, a legacy mode DMA write to the device is preceded by a DMA Activate – Device to Host FIS with the DMA context having been pre-established by the host.

When used for transferring data for DMA operations multiple Data – Host to Device or Device to Host FIS's can follow in either direction. Segmentation can occur when the transfer count exceeds the maximum Data – Host to Device or Device to Host transfer length or if a data transfer is interrupted.

When used for transferring data in response to a PIO Setup all of the data must be transmitted in a single Data FIS.

In the event that a transfer is broken into multiple FIS's, all intermediate FIS's must contain an integral number of full Dwords. If the total data transfer is for an odd number of words, then the high order word (word 1) of the last Dword of the last FIS shall be padded with zeros before transmission and discarded on reception.

The Serial ATA protocol does not permit for the transfer of an odd number of bytes.

#### 8.5.9.3 Reception

Neither the host nor device is expected to buffer an entire Data FIS in order to check the CRC of the FIS before processing the data. Incorrect data reception for a Data FIS should be reflected in the overall command completion status.

## 8.6 Host transport states

### 8.6.1 Host transport idle state diagram

| HTI1: HT_HostIdle | Host adapter waits for frame or frame request. | |
|---|---|---|
| 1. Shadow Command register written | → | HT_CmdFIS |
| 2. Shadow Device Control register written | → | HT_CntrlFIS [1] |
| 3. Frame receipt indicated by Link layer | → | HT_ChkTyp |
| 4. First party DMA host to device communication indicated by application layer | → | HT_DMASTUPFIS |
| 5. BIST FIS Transmission requested | → | HT_XmitBIST |
| 6. Previous FIS was PIO Setup and Application layer indicates data direction is host to device | → | HT_PIOOTrans2 [2] |
| NOTE:<br>1. The transition to this state is mandatory if the state of the SRST bit in the Device Control Shadow Register is changed, and is optional if the state of the SRST bit is not changed.<br>2. The PIO Setup FIS shall set an indication that PIO Setup was the last FIS received. Indication from the application layer that it is transmitting data to the device can be determined from the application layer performing write operations to the Data register in the Shadow Register Block. | | |

| HTI2: HT_ChkTyp | Received FIS type checked. | |
|---|---|---|
| 1. Register FIS type detected | → | HT_RegFIS |
| 2. Set Device Bits FIS type detected | → | HT_DB_FIS |
| 3. DMA Activate FIS type detected | → | HT_DMA_FIS |
| 4. PIO Setup FIS type detected) | → | HT_PS_FIS |
| 5. First Party DMA Setup FIS type detected | → | HT_DS_FIS |
| 6. BIST FIS type detected | → | HT RcvBIST |
| 7. Data FIS type detected and previous FIS was not PIO Setup | → | HT_DMAITrans [1] |
| 8. Data FIS type detected and previous FIS was PIO Setup | → | HT_PIOITrans1 [1] |
| 9. Unrecognized FIS received | → | HT_HostIdle |
| 10. Notification of illegal transition error received from Link layer | → | HT_HostIdle |
| NOTE:<br>1. The PIO Setup FIS shall set an indication that PIO Setup was the last FIS sent, so that this state can determine whether to transition to DMA data transfer, or PIO data transfer. | | |

**HTI1: HT_HostIdle state:** This state is entered when a Frame Information Structure (FIS) transaction has been completed by the Transport layer.

When in this state, the Transport layer waits for the shadow Command register to be written, the shadow Device Control register to be written, or the Link layer to indicate that an FIS is being received.

**TransitionHTI1:1:** When the shadow Command register is written, the Transport layer shall make a transition to the HTCM1: HT_CmdFIS state.

**Transition HTI1:2:** When the shadow Control register is written, and the state of the SRST bit is changed from its previous state, the Transport layer shall make a transition to the HTCR1: HT_CntrlFIS state. This transition may optionally be made upon a write operation to the Control shadow register, or upon a write operation to the Control shadow register that changes its value from the previous value.

**Transition HTI1:3:** When the Link layer indicates that an FIS is being received, the Transport layer shall make a transition to the HTI2: HT_ChkTyp state.

**Transition HTI1:4:** When the Application layer indicates that a DMA Setup FIS is to be sent, the Transport layer shall make a transition to the HT_DMASTUP0:HT_DMASTUPFIS state.

**Transition HTI1:5:** When the application layer requests the transmission of a BIST request to the device the Transport layer shall make a transition to the HTXBIST1:HT state.

**Transition HTI1:6:** When the application layer requests the transmission of data to the device and the previous FIS was a PIOSetup type, the Transport layer shall make a transition to the HTPS3:HT_PIOOTrans2 state. The application layer signals transmission of PIO data to the device by performing writes to the Data register in the Shadow Register Block.

**HTI2: HT_ChkTyp state:** This state is entered when the Link layer indicates that an FIS is being received.

When in this state, the Transport layer checks the FIS type of the incoming FIS.

**Transition HTI2:1:** When the incoming FIS is a register type, the Transport layer shall notify the Link Layer that it has received a valid FIS, and make a transition to the HTR1: HT_RegFIS state.

**Transition HTI2:2:** When the incoming FIS is a Set Device Bits type, the transport layer shall notify the Link Layer that it has received a valid FIS and make a transition to the HTDB0:HT_DB_FIS state.

**Transition HTI2:3:** When the incoming FIS is a DMA Activate type, the Transport layer shall notify the Link Layer that it has received a valid FIS, and make a transition to the HTDA1: HT_DMA_FIS state.

**Transition HTI2:4:** When the incoming FIS is a PIO Setup type, the Transport layer shall notify the Link Layer that it has received a valid FIS, and make a transition to the HTPS1: HT_PS_FIS state.

**Transition HTI2:5:** When the incoming FIS is a First-party DMA Setup type, the Transport layer shall notify the Link Layer that it has received a valid FIS, and make a transition to the HTDS1: HT_DS_FIS state.

**Transition HTI2:6:** When the incoming FIS is a BIST Activate type, the Transport layer shall notify the Link Layer that it has received a valid FIS, and make a transition to the HTRBIST1:HT_RcvBIST state.

**Transition HTI2:7:** When the incoming FIS is a Data type, and the previous FIS was not a PIO Setup type, the Transport layer shall notify the Link Layer that it has received a valid FIS, and make a transition to the HTDA6:HT_DMAITrans state.

**Transition HTI2:8:** When the incoming FIS is a Data type, and the previous FIS was a PIO Setup type, the Transport layer shall notify the Link Layer that it has received a valid FIS, and make a transition to the HTPS6:HT_PIOITrans state.

**Transition HTI2:9:** When the received FIS is of an unrecognized, or unsupported type, the Transport layer shall notify the Link Layer that it has received an unrecognized FIS, and make a transition to the HTI1: HT_HostIdle state.

**Transition HTI2:10:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1: HT_HostIdle state.

### 8.6.2 Host Transport transmit command FIS diagram

This protocol builds an FIS that contains the host adapter shadow register content and sends it to the device when the software driver or BIOS writes the host adapter shadow Command register.

| HTCM1: HT_CmdFIS | Construct Cmd type FIS from the content of the shadow registers and notify Link to transfer. | |
|---|---|---|
| 1. FIS transfer complete | → | HT_TransStatus |
| 2. Notification of illegal transition error received from Link layer | → | HT_HostIdle |
| 3. Frame receipt indicated by Link layer | → | HT_HostIdle |

| HTCM2: HT_TransStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. | |
|---|---|---|
| 1. Status checked, and no error detected | → | HT_HostIdle |
| 2. Status checked, and error detected | → | HT_CmdFIS |
| 3. Status checked, error detected, and SRST asserted | → | HT_HostIdle |
| 4. Notification of illegal transition error received from Link layer | → | HT_HostIdle |

**HTCM1: HT_CmdFIS state:** This state is entered when the shadow Command register is written.

When in this state, the Transport layer shall construct a register FIS, notify the Link layer that the FIS is to be transmitted, and pass the FIS to the Link layer. The Transport layer shall not perform flow control on a register FIS.

**Transition HTCM1:1:** When the entire FIS has been passed to the Link layer, the Transport layer shall indicate to the Link layer that the FIS transmit is complete and make a transition to the HTCM2: HT_TransStatus state.

**Transition HTCM1:2:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1: HT_HostIdle state.

**Transition HTCM1:3:** When the Link layer indicates that an FIS is being received, the Transport Layer shall make a transition to the HTI1:HT_HostIdle state.

**HTCM2: HT_TransStatus state:** This state is when the entire FIS has been passed to the Link layer.

When in this state, the Transport layer shall wait for the Link and Phy ending status for the FIS and take appropriate error handling action if required.

**Transition HTCM2:1:** When the FIS status has been handled and no errors detected, the Transport layer shall transition to the HTI1: HT_HostIdle state.

**Transition HTCM2:2:** When the FIS status has been handled and an error has been detected, and the host has not asserted the SRST by writing to the Device Control register, the Transport layer shall transition to the HTCM1: HT_CmdFIS state.

**Transition HTCM2:3:** When the FIS status has been handled, an error has been detected, and the host has asserted the SRST by writing to the Device Control register, or a device reset command has

been written to an ATAPI device, the Transport layer shall inform the Link layer to send a SYNC primitive, and transition to the HTCM1: HT_HostIdle state.

**Transition HTCM2:4:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1: HT_HostIdle state.

### 8.6.3 Host Transport transmit control FIS diagram

This protocol builds an FIS that contains the host adapter shadow register content and sends it to the device when the software driver or BIOS writes the host adapter shadow Device Control register.

| HTCR1: HT_CntrlFIS | Construct Cntrl type FIS from the content of the shadow registers and notify Link to transfer. | |
|---|---|---|
| 1. FIS transfer complete | → | HT_TransStatus |
| 2. Notification of illegal transition error received from Link layer | → | HT_HostIdle |
| 3. Frame receipt indicated by Link layer | → | HT_HostIdle |

| HTCR2: HT_TransStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. | |
|---|---|---|
| 1. Status checked and no errors detected | → | HT_HostIdle |
| 2. Status check and at least one error detected | → | HT_Cntrl_FIS |

**HTCR1: HT_Cntrl_FIS state:** This state is entered when the shadow Device Control register is written.

When in this state, the Transport layer shall construct a register FIS, notify the Link layer that the FIS is to be transmitted, and pass the FIS to the Link layer. The Transport layer shall not perform flow control on a register FIS.

**Transition HTCR1:1:** When the entire FIS has been passed to the Link layer, the Transport layer shall indicate to the Link layer that the FIS transmit is complete and make a transition to the HTCR2: HT_TransStatus state.

**Transition HTCR1:2:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1: HT_HostIdle state.

**Transition HTCR1:3:** When the Link layer indicates that an FIS is being received, the Transport Layer shall make a transition to the HTI1:HT_HostIdle state.

**HTCR2: HT_TransStatus state:** This state is when the entire FIS has been passed to the Link layer.

When in this state, the Transport layer shall wait for the Link and Phy ending status for the FIS and take appropriate error handling action if required.

**Transition HTCR2:1:** When the FIS status has been handled and no errors have been detected, the Transport layer shall transition to the HTI1: HT_HostIdle state.

**Transition HTCR2:2:** When the FIS status has been handled and at least one error has been detected, the Transport layer shall transition to the HTCR1: HT_Cntrl_FIS.

### 8.6.4 Host Transport transmit First-party DMA Setup – Device to Host or Host to Device FIS state diagram

This protocol transmits a DMA Setup – Device to Host or Host to Device FIS to a receiver. This FIS is a request by a transmitter for the receiver to program its DMA controller for a First-party DMA transfer and is followed by one or more Data FIS's that transfer data. The DMA Setup – Device to Host or Host to Device FIS request includes the transfer direction indicator, the host buffer identifier, the host buffer offset, the byte count, and the interrupt flag. The Transport layer shall not perform flow control on the DMA Setup – Host to Device or Device to Host FIS.

| HTDMASTUP0: HT_DMASTUPFIS | Construct the DMA Setup – Host to Device or Device to Host FIS from the content provided by the Application layer and notifies Link to transfer. |
|---|---|

| | | |
|---|---|---|
| 1. FIS transfer complete | → | HT_TransStatus |
| 2. Notification of illegal transition error received from Link layer | → | HT_HostIdle |
| 3. Frame receipt indicated by Link layer | → | HT_HostIdle |

| HTPDMASTUP1: HT_TransStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. |
|---|---|

| | | |
|---|---|---|
| 1. Status checked, and no error detected. | → | HT_HostIdle |
| 2. Status checked, error detected, and no SRST. | → | HT_DMASTUPFIS |
| 3. Status checked, error detected, and SRST. | → | HT_HostIdle |

**HTDMASTUP0: HT_DMASTUPFIS state:** This state is entered when the Application requests the transmission of a DMA Setup – Host to Device or Device to Host FIS.

When in this state, the Transport layer shall construct a DMA Setup – Host to Device or Device to Host FIS, notify the Link layer that the FIS is to be transmitted, and pass the FIS to the Link layer. The Transport layer shall not perform flow control on a DMA Setup – Host to Device or Device to Host FIS.

**Transition HTDMASTUP0:1:** When the entire FIS has been passed to the Link layer, the Transport layer shall indicate to the Link layer that the FIS transmission is complete and make a transition to the HTDMASTUP1: HT_TransStatus state.

**Transition HTDMASTUP0:2:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1: HT_HostIdle state.

**Transition HTDMASTUP0:3:** When the Link layer indicates that an FIS is being received, the Transport Layer shall make a transition to the HTI1:HT_HostIdle state.

**HTPDMASTUP1: HT_TransStatus state:** This state is entered when the entire FIS has been passed to the Link layer.

When in this state, the Transport layer shall wait for the Link and Phy ending status for the FIS and take appropriate error handling action if required.

**Transition HTDMASTUP1:1:** When the FIS status has been handled, and no error detected, the Transport layer shall transition to the HTI1: HT_HostIdle state.

**Transition HTDMASTUP1:2**: When the FIS status has been handled, an error detected and the host has not asserted the SRST by writing to the Device Control register, the Transport layer shall report status to the Link layer, and retry this transfer by transitioning to the HTDMASTUP0: HT_DMASTUPFIS state.

**Transition HTDMASTUP1:3:** When the host has asserted the SRST bit by writing to the Device Control register, or the ATAPI Device Reset command is issued, the Transport layer shall inform the Link layer to send SYNC primitives, and the Transport layer shall transition to the HTI1: HT_HostIdle state.

### 8.6.5 Host Transport transmit BIST Activate FIS

This protocol builds a BIST Activate FIS that tells the device to prepare to enter the appropriate Built-in Self-test mode. After successful transmission, the host Transport layer enters the idle state. The application layer, upon detecting successful transmission to the device shall then cause the host's Transport layer, Link layer and Physical layer to enter the appropriate mode for the transmission of the Built-in Test data defined by the FIS. The means by which the Transport, Link and Physical layers are placed into self-test mode are not defined by this specification.

| HTXBIST1: HT_XmitBIST | Construct the BIST Activate FIS from the content provided by the Application layer and notify Link to transfer. | |
|---|---|---|
| 1. FIS transfer complete | → | HT_TransBISTStatus |
| 2. Notification of illegal transition error received from Link layer | → | HT_HostIdle |
| 3. Frame receipt indicated by Link layer | → | HT_HostIdle |

| HTXBIST2: HT_TransBISTStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. | |
|---|---|---|
| 1. Status checked and no errors detected | → | HT_HostIdle |
| 2. Status check and at least one error detected | → | HT_XmitBIST |

**HTXBIST1: HT_XmitBIST state:** This state is entered to send a BIST FIS to the device.

**Transition HTXBIST1:1:** When the entire FIS has been passed to the Link layer, the Transport layer shall indicate to the Link layer that the FIS transmission is complete and make a transition to the HTXBIST2:HT_TransBISTStatus state.

**Transition HTXBIST1:2:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1: HT_HostIdle state.

**Transition HTXBIST1:3:** When the Link layer indicates that an FIS is being received, the Transport Layer shall make a transition to the HTI1:HT_HostIdle state.

**HTXBIST2: HT_TransBISTStatus state:** This state is entered when the entire FIS has been passed to the Link layer.

**Transition HTXBIST2:1:** When the FIS transmission is completed and no errors have been detected the Transport layer shall transition to the HTI1:HT_HostIdle state.

**Transition HTXBIST2:2:** When the FIS transmission is completed and at least one error is detected the Transport layer shall transition to the HTXBIST1: HT_XmitBIST state.

### 8.6.6 Host Transport decompose Register FIS diagram

This protocol receives an FIS from the device containing new shadow register content and places that content into the shadow registers.

| HTR1: HT_RegFIS | Place FIS contents from device into appropriate holding registers. | | |
|---|---|---|---|
| | 1. FIS transfer complete | → | HT_TransStatus |
| | 2. Notification of illegal transition error received from Link layer | → | HT_HostIdle |

| HTR2: HT_TransStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. | | |
|---|---|---|---|
| | 1. Status checked | → | HT_HostIdle |

**HTR1: HT_RegFIS state:** This state is entered the Link layer has indicated that an FIS is being received and that the FIS is of the register type.

When in this state, the Transport layer shall decompose the register FIS and place the contents into the appropriate holding registers. The Transport layer shall not perform flow control on a register FIS.

**Transition HTR1:1:** When the entire FIS has been placed into the holding registers, the Transport layer shall make a transition to the HTR2: HT_TRANS_STATUS state.

**Transition HTR1:2:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1: HT_HostIdle state.

**HTR2: HT_TransStatus state:** This state is when the entire FIS has been placed into the holding registers.

When in this state, the Transport layer shall wait for the Link and Phy ending status for the FIS and take appropriate error handling action if required.

**Transition HTR2:1:** When the FIS status has been handled and no errors detected, the contents of the holding registers shall be placed in the shadow registers and if the interrupt bit is set, the Transport layer shall set the interrupt pending flag. The Transport layer shall transition to the HTI1: HT_HostIdle state. When the FIS status has been handled and at least one error detected, the contents of the holding registers shall not be transferred to the shadow registers, error status shall be returned to the device, and the Transport layer shall transition to the HTI1: HT_HostIdle state.

### 8.6.7 Host Transport decompose a Set Device Bits FIS state diagram

This protocol receives an FIS from the device containing new Error and Status shadow register content and places that content into the Error and Status Shadow registers.

| HTDB0:HT_DB_FIS | Receive Set Device Bits FIS | |
|---|---|---|
| 1. FIS status checked and no error detected | → | HT_Dev_Bits |
| 2. FIS status checked and error detected. | → | HT_HostIdle |

| HTDB1:HT_Dev_Bits | Load Error register and bits of the Status register | |
|---|---|---|
| 1. Register bits loaded | → | HT_HostIdle |

**HTDB0:HT_DB_FIS state**: This state is entered when the Link layer has indicated that an FIS being received and that the FIS is a Set Device Bits type.

When in this state, the Transport layer shall wait for the FIS reception to complete and for Link and Phy ending status to be posted.

**Transition HTDB0:1**: When the FIS reception is complete with no errors detected, the Transport layer shall transition to the HTDB1:HT_Dev_Bits state.

**Transition HTDB0:2**: When the FIS reception is complete with errors detected, the Transport layer shall return error status to the device and transition to the HTI1:HT_HostIdle state.

**HTDB1:HT_Dev_Bits state**: This state is entered when a Set Device Bits FIS has been received with no errors.

When in this state, the data in the Error field of the received FIS shall be loaded into the host adapter's shadow Error register. The data in the Status-Hi field of the received FIS shall be loaded into bits 6, 5, and 4 of the shadow Status register. The data in the Status-Lo field of the received FIS shall be loaded into bits 2, 1, and 0 of the shadow Status register. Bit 7, BSY, and bit 3, DRQ, in the shadow Status register shall not be changed. If the I bit in the FIS is set to one and if both the BSY bit and the DRQ bit in the shadow Status register are cleared to zero, then the host adapter shall enter an interrupt pending state.

**Transition HTDB1:1:** The Transport layer shall transition to the HTI1:HT_HostIdle state.

### 8.6.8 Host Transport decompose a DMA Activate FIS diagram

This protocol receives an FIS that requests a legacy DMA data out transfer. The data transfer is from the host to the device and the DMA Activate FIS causes the host adapter to transmit the data in a subsequent Data FIS.

| HTDA1: HT_DMA_FIS | | |
|---|---|---|
| 1. Status checked and no error detected. | → | HT_DMAOTrans1 |
| 2. Status checked and error detected | → | HT_HostIdle |
| 3. Notification of illegal transition error received from Link layer | → | HT_HostIdle |

| HTDA2: HT_DMAOTrans1 | DMA controller initialized? | |
|---|---|---|
| 1. DMA controller not initialized. | → | HT_DMAOTrans1 |
| 2. DMA controller initialized. | → | HT_DMAOTrans2 |
| 3. Notification of software reset or device reset command from the host application | → | HT_HostIdle |

| HTDA3: HT_DMAOTrans2 | Activate DMA controller | |
|---|---|---|
| 1. Transfer not complete and <8KB transmitted | → | HT_DMAOTrans2 |
| 2. Transfer not complete and 8KB transmitted | → | HT_DMAEnd |
| 3. Abort notification from Link layer | → | HT_DMAEnd |
| 4. Transfer complete | → | HT_DMAEnd |
| 5. Notification of illegal transition error received from Link layer | → | HT_HostIdle |
| 6. Notification of software reset or device reset command from the host application | → | HT_HostIdle |

| HTDA4: HT_DMAEnd | Check DMA Controller completion | |
|---|---|---|
| 1. DMA controller actions completed, no error detected | → | HT_HostIdle |
| 2. DMA controller actions completed, and error detected. | → | HT_HostIdle |
| 3. Abort notification from Link layer, no error detected | → | HT_HostIdle |
| 4. Abort notification from Link layer, error detected. | → | HT_HostIdle |

| HTDA5: HT_DMAITrans | Activate DMA controller if initialized, receive Data FIS. | |
|---|---|---|
| 1. Transfer not complete | → | HT_DMAITrans |
| 2. SRST asserted, or device reset command issued | → | HT_HostIdle |
| 3. Transfer complete | → | HT_DMAEnd |
| 4. Notification of illegal transition error received from Link layer | → | HT_HostIdle |

**HTDA1: HT_DMA_FIS state:** This state is entered when the Link layer has indicated that an FIS is being received and the Transport layer has determined that a DMA Activate FIS is being received.

When in this state, the Transport layer shall determine the direction of the DMA transfer being activated.

**Transition HTDA1:1:** The Transport layer shall make a transition to the HTDA2: HT_DMAOTrans1 state. This transition occurs if no error is detected.

**Transition HTDA1:2:** When an error is detected, status is conveyed to the Link layer and to the application layer. The Transport layer shall make a transition to the HTI1:HT_HostIdle state.

**Transition HTDA1:3:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1: HT_HostIdle state.

**HTDA2: HT_DMAOTrans1 state:** This state is entered when it is determined that the DMA transfer that is being activated is a transfer from host to device..

When in this state, the Transport layer shall determine if the DMA controller has been initialized.

**Transition HTDA2:1:** When the DMA controller has not yet been initialized, the Transport layer shall transition to the HTDA2: HT_DMAOTrans1 state.

**Transition HTDA2:2:** When the DMA controller has been initialized, the Transport layer shall transition to the HTDA3: HT_DMAOTrans2 state.

**Transition HTDA2:3:** When the host has asserted the SRST bit by writing to the Device Control register, or the ATAPI Device Reset command is written to the Command register, the Transport layer shall inform the Link layer to send SYNC primitives, and the Transport layer shall transition to the HTI1:HT_HostIdle state.

**HTDA3: HT_DMAOTrans2 state:** This state is entered when the DMA controller has been initialized.

When in this state, the Transport layer shall activate the DMA controller and pass data to the Link layer.

**Transition HTDA3:1:** When the transfer is not complete and less than 8KB of payload data has been transmitted, the Transport layer shall transition to the HTDA3: HT_DMAOTrans2 state.

**Transition HTDA3:2:** When the transfer is not complete but 8KB of payload data has been transmitted, the Link layer shall be notified to close the current frame and the Transport layer shall deactivate the DMA engine and transition to the HTDA4: HT_DMAEnd state.

**Transition HTDA3:3:** When notified by the Link layer that the DMA Abort primitive was received, the Transport layer shall transition to the HTDA4: HT_DMAEnd state.

**Transition HTDA3:4:** When the requested DMA transfer is complete, the Transport layer shall transition to the HTDA4: HT_DMAEnd state.

**Transition HTDA3:5:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1: HT_HostIdle state.

**Transition HTDA3:6:** When the host has asserted the SRST bit by writing to the Device Control register, or the ATAPI Device Reset command is written to the Command register, the Transport layer shall inform the Link layer to send SYNC primitives, and the Transport layer shall transition to the HTI1:HT_HostIdle state.

**HTDA4: HT_DMAEnd state:** This state is entered when the DMA data transfer is complete.

When in this state, the Transport layer shall ensure that the activities of the DMA controller have completed.

**Transition HTDA4:1:** When the DMA controller has completed its activities, whether it has exhausted its transfer count or has been deactivated as a result of reaching the 8KB data payload limit, the Transport layer shall transition to the HTI1: HT_HostIdle state. This transition occurs if no error is detected.

**Transition HTDA4:2:** When an error is detected, status shall be reported to the Link and application layers. The Transport layer shall transition to the HTI1:HT_HostIdle state.

**Transition HTDA4:3:** When notified by the Link layer that a DMA Abort primitive was received, the transfer shall be truncated, and the Link layer notified to append CRC and end the frame. When it is determined that the transfer is completed with no error, the Transport layer shall make a transition to the HTI1:HT_HostIdle state.

**Transition HTDA4:4:** When notified by the Link layer that a DMA Abort primitive was received, the transfer shall be truncated, and the Link layer notified to append CRC and end the frame. When it is determined that the transfer is completed with an error, the Transport layer shall report status to the host and make a transition to the HTI1:HT_HostIdle state.

**HTDA5: HT_DMAITrans state:** This state is entered when the Transport layer has determined that the DMA transfer being activated is from device to host.

When in this state, the Transport layer shall activate the DMA controller if the DMA controller is initialized. A data frame will be received from the device and a received data Dword shall be placed in the data FIFO.

When in this state, the Transport layer shall wait until the Link layer has begun to receive the DMA data frame and data is available to be read by the host.

**Transition HTDA5:1:** When the transfer is not complete, the Transport layer shall transition to the HTDA5: HT_DMAITrans state. This includes the condition where the host DMA engine has not yet been programmed and the transfer is therefore held up until the DMA engine is prepared to transfer the received data to the destination memory locations.

**Transition HTDA5:2:** When the SRST bit is asserted by the host writing the Device Control register, or a device reset command has been written to an ATAPI device, the Link layer shall be informed to send a SYNC primitive, and the Transport layer shall transition to the HTI1:HT_Idle state.

**Transition HTDA5:3:** When the requested DMA transfer is complete, the Transport layer shall transition to the HTDA4: HT_DMAEnd state.

**Transition HTDA5:4:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1: HT_HostIdle state.

### 8.6.9 Host Transport decompose a PIO Setup FIS state diagram

This protocol receives a PIO Setup FIS that requests a PIO data transfer. If the direction is from host to device, the Transport layer transmits a Data FIS to the device containing the PIO data. If the direction of transfer is from device to host, the Transport layer receives a Data FIS from the device. The PIO data must be sent in a single Data FIS.

| HTPS1: HT_PS_FIS | Determine the direction of the requested PIO transfer. | | |
|---|---|---|---|
| | 1. Transfer host to device, no error detected. D bit = 0. | → | HT_PIOOTrans1 |
| | 2. Transfer device to host, no error detected. D bit = 1. | → | HT_HostIdle |
| | 3. Error detected. | → | HT_HostIdle |
| | 4. Notification of illegal transition error received from Link layer | → | HT_HostIdle |

| HTPS2: HT_PIOOTrans1 | Place initial register content received from FIS into shadow registers. | | |
|---|---|---|---|
| | 1. Unconditional | → | HT_ HostIdle |

| HTPS3: HT_PIOOTrans2 | Wait for Link layer to indicate data transfer complete | | |
|---|---|---|---|
| | 1. Transfer not complete | → | HT_PIOOTrans2 |
| | 2. Transfer complete | → | HT_PIOEnd |
| | 3. Abort notification from Link layer | → | HT_PIOEnd |
| | 4. Notification of illegal transition error received from Link layer | → | HT_HostIdle |
| | 5. Notification of software reset or device reset command from the host application | → | HT_HostIdle |

| HTPS4: HT_PIOEnd | Place ending register content from PIO REQ FIS into shadow registers. | | |
|---|---|---|---|
| | 1. No error detected. | → | HT_HostIdle |
| | 2. Error detected | → | HT_HostIdle |

| HTPS5: HT_PIOITrans1 | Wait until initial PIO data received in data frame. | | |
|---|---|---|---|
| | 1. Received PIO data available. | → | HT_PIOITrans2[1] |
| | 2. SRST asserted, or device reset command issued | → | HT_HostIdle |
| | 3. Notification of illegal transition error received from Link layer | → | HT_HostIdle |
| | NOTE:<br>1. When transitioning to the HT_PIOITrans2 state, the starting status and I bit value from the PIO Setup FIS shall be transferred to the status shadow register and interrupt signal shall then reflect the value of the I bit. | | |

| HTPS6: HT_PIOITrans2 | Wait for Link layer to indicate data transfer complete. | | |
|---|---|---|---|
| | 1. Transfer not complete | → | HT_PIOITrans2 |
| | 2. Transfer complete | → | HT_PIOEnd |
| | 3. Abort notification from Link layer | → | HT_PIOEnd |
| | 4. Notification of illegal transition error received from Link layer | → | HT_HostIdle |

**HTPS1: HT_PS_FIS state:** This state is entered when the Link layer has indicated that an FIS is being received and that the Transport layer has determined a PIO Setup FIS is being received.

When in this state, the Transport layer shall determine the direction of the requested PIO transfer and indicate that the last FIS sent was a PIO Setup.

**Transition HTPS1:1:** When the direction of transfer requested is from host to device (D=0), the Transport layer shall make a transition to the HTPS2: HT_PIOOTrans1 state. This transition occurs if no error is detected.

**Transition HTPS1:2:** When the direction of transfer requested is from device to host (D=1), the Transport layer shall make a transition to the HTI1:HT_HostIdle state. This transition occurs if no error is detected.

**Transition HTPS1:3:** When an error is detected, status shall be reported to the Link layer. The Transport layer shall make a transition to the HTI1:HT_HostIdle state.

**Transition HTPS1:4:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1:HT_HostIdle state.

**HTPS2: HT_PIOOTrans1 state:** This state is entered when the direction of the requested PIO data transfer is from host to device.

When in this state, the Transport layer shall place the FIS initial register content into the shadow registers, the FIS byte count, and set the interrupt pending flag if the FIS indicates to do so.

**Transition HTPS2:1:** When the FIS initial register content has been placed into the shadow registers, interrupt pending set if requested, and the Transport layer is ready to begin transmitting the requested PIO data FIS, the Transport layer shall make a transition to the HTI1:HT_HostIdle state.

**HTPS3: HT_PIOOTrans2 state:** This state is entered when PIO data is available in the PIO FIFO to be passed the Link layer.

When in this state, the Transport layer shall wait for the Link layer to indicate that all data has been transferred.

NOTE – Since the software driver or BIOS sees DRQ set and BSY cleared, it continues writing the Data register filling the PIO FIFO.

**Transition HTPS3:1:** When the transfer is not complete, the Transport layer shall transition to the HTPS3: HT_PIOOTrans2 state.

**Transition HTPS3:2:** When the byte count for this DRQ data block is reached, the Transport layer shall transition to the HTPS4: HT_PIOEnd state.

**Transition HTPS3:3:** When notified by the Link layer that the DMA Abort primitive was received, the Transport layer shall transition to the HTPS4: HT_PIOEnd state.

**Transition HTPS3:4:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1:HT_HostIdle state.

**Transition HTPS3:5:** When the host has asserted the SRST bit by writing to the Device Control register, or the ATAPI Device Reset command is written to the Command register, the Transport layer shall inform the Link layer to send SYNC primitives, and the Transport layer shall transition to the HTI1:HT_HostIdle state.

**HTPS4: HT_PIOEnd state:** This state is entered when the PIO data transfer is complete.

When in this state, the Transport layer shall place the ending register content from the received PIO request FIS into the shadow registers.

**Transition HTPS4:1:** When the ending register content for the PIO request FIS has been placed into the shadow registers and there were no errors detected with the transfer, the Transport layer shall transition to the HTI1: HT_HostIdle state.

**Transition HTPS4:2:** When the ending register content from the previous PIO Setup FIS has been placed into the shadow registers, the Transport layer shall transition to the HTI1:HT_HostIdle state. For data in transfers, the Transport layer shall notify the Link layer of any error encountered during the transfer, and the error shall be reflected in the end of frame handshake. If the transfer was not the final transfer for the PIO data in command, the device shall reflect the error status by transmitting an appropriate Register FIS to the host. If the transfer was the final transfer for the associated PIO data in command, the error condition is not detectable. For data out transfers, errors detected by the device shall be reflected in the end of frame handshake. The device shall reflect the error status by transmitting an appropriate Register FIS to the host.

**HTPS5: HT_PIOITrans1 state:** This state is entered when the direction of the PIO data transfer is device to host.

When in this state, the Transport layer shall wait until the Link layer has begun to receive the PIO data frame and data is available to be read by the host.

**Transition HTPS5:1:** When data is available for the host to read in the shadow Data register, the Transport layer shall place the initial register content received in the PIO Setup frame into the shadow registers and transition to the HTPS6: HT_PIOITrans2 state.

**Transition HTPS5:2:** When the host has asserted the SRST bit by writing to the Device Control register, or the ATAPI Device Reset command is issued, the Transport layer shall inform the Link layer to send SYNC primitives, and the Transport layer shall transition to the HTI1: HT_HostIdle state.

**Transition HTPS5:3:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1:HT_HostIdle state.

**HTPS6: HT_PIOITrans2 state:** This state is entered when PIO data is available in the PIO FIFO to be read by the host and the initial shadow register content has been set.

When in this state, the Transport layer shall wait for the Link layer to indicate that the data transfer is complete

**Transition HTPS6:1:** When the transfer is not complete, the Transport layer shall transition to the HTPS6: HT_PIOITrans2 state.

**Transition HTPS6:2:** When the byte count for this DRQ data block is reached, the Transport layer shall transition to the HTPS5: HT_PIOEnd state.

**Transition HTPS6:3:** When notified by the Link layer that the DMA Abort primitive was received, the Transport layer shall transition to the HTPS5: HT_PIOEnd state.

**Transition HTPS6:4:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1:HT_HostIdle state.

### 8.6.10 Host Transport decompose a First-party DMA Setup FIS state diagram

This protocol receives an FIS that sets up the host adapter DMA controller to allow the transfer of a first party DMA data FIS to the host. The actual DMA Activate FIS will be transmitted as a separate DMA Activate FIS protocol following the completion of this protocol.

| HTDS1: HT_DS_FIS | Initialize the DMA controller for a first party DMA with the content of the request FIS. |
|---|---|

| | | |
|---|---|---|
| 1. No error detected. | → | HT_HostIdle |
| 2. Error detected. | → | HT_HostIdle |
| 3. Notification of illegal transition error received from Link layer | → | HT_HostIdle |

**HTDS1: HT_DR_FIS state:** This state is entered when the Link layer has indicated that an FIS is being received and that the Transport layer has determined the FIS is of the first party DMA in request type.

When in this state, the Transport layer shall initialize the DMA controller with content from the FIS.

**Transition HTDS1:1:** When the DMA controller has been initialized, the Transport layer shall transition to the HTI1: HT_HostIdle state. This transition is made if no error is detected.

**Transition HTDS1:2:** When an error is detected, status shall be reported to the Link layer. The Transport layer shall transition to the HTI1:HI_HostIdle state.

**Transition HTDS1:3:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1:HT_HostIdle state.

### 8.6.11 Host transport decompose a BIST Activate FIS state diagram

This protocol receives an FIS that instructs the host to enter one of several Built-in Self-test modes that cause the host to retransmit the data it receives. If the mode is supported the Host's application layer will place both the transmit and receive portions of the Transport, Link and/or Physical layers into appropriate state to perform the loopback operation.

| HTRBIST1: HT_RcvBIST | Determine validity of loopback mode requested. | |
|---|---|---|
| 1. Status checked, no error detected and Loopback mode valid | → | HT_BISTTrans1 |
| 2. Status checked, no error detected and Loopback mode is invalid or not supported. | → | HT_HostIdle |
| 3. Status checked and error detected | → | HT_HostIdle |
| 4. Notification of illegal transition error received from Link layer | → | HT_HostIdle |

| HTRBIST2: HT_BISTTrans1 | Notify application layer of desired BIST modes | |
|---|---|---|
| 1. Unconditional | → | HT_HostIdle |

**HTRBIST1: HT_RcvBIST state**: This state is entered when the link layer has indicated that an FIS is being received and the Transport layer has determined that a BIST Activate FIS is being received.

When in this state, the Transport layer shall determine the validity of the loopback request.

**Transition HTRBIST1:1:** If no reception error is detected and the FIS contents indicate a form of loopback request that is supported by the host the Transport layer shall make a transition to the HTRBIST2: HT_BISTTrans1 state.

**Transition HTRBIST1:2:** If no reception error is detected and the FIS contents indicate a form of loopback request that is not supported by the host the Transport layer shall make a transition to the HTI1:HT_HostIdle state.

**Transition HTRBIST1:3:** If a reception error is indicated the Transport layer shall make a transition to the HTI1:HT_HostIdle state.

**Transition HTRBIST1:4:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the HTI1:HT_HostIdle state.

**HTRBIST2: HT_BISTTrans1 state:** This state is entered when the Transport layer has determined that a valid BIST Activate FIS has been received.

Having received a valid FIS, the Transport layer informs the application layer that it should place the Transport, Link and Physical layers into the appropriate modes to loop the received data back to the transmitter. The method by which this is performed is not defined by this specification.

**Transition HTRBIST2:1:** When the Application layer has been notified the Transport layer shall transition to the HTI1:HostIdle state.

## 8.7 Device transport states

### 8.7.1 Device transport idle state diagram

| DTI0: DT_DeviceIdle | Device waits for FIS or FIS request. | | |
|---|---|---|---|
| | 1. Transmit of RegDHFIS requested by Application layer | → | DT_RegDHFIS |
| | 2. Transmission of Set Device Bits requested by Application | → | DT_DB_FIS |
| | 3. Transmit of PIOSTUPFIS requested by Application layer | → | DT_PIOSTUPFIS |
| | 4. Transmit of DMAACTFIS requested by Application layer | → | DT_DMAACTFIS |
| | 5. Transmit of DMASTUPFIS requested by Application layer | → | DT_DMASTUPDHFIS |
| | 6. Transmit of DATAFIS requested by Application layer | → | DT_DATAIFIS |
| | 7. Transmit of BIST FIS requested by Application layer | → | DT_XmitBIST |
| | 8. Frame receipt indicated by Link layer | → | DT_ChkTyp |

| DTI1: DT_ChkTyp | Received FIS type checked. | | |
|---|---|---|---|
| | 1. Register FIS type detected | → | DT_RegHDFIS |
| | 2. Data FIS type detected | → | DT_DATAOFIS |
| | 3. DMA Setup FIS type detected | → | DT_DMASTUPHDFIS |
| | 4. BIST Activate FIS type detected | → | DT_rcvBIST |
| | 5. Notification of illegal transition error received from Link layer or unrecognized FIS type | → | DT_DeviceIdle |

**DTI0: DT_DeviceIdle state:** This state is entered when a Frame Information Structure (FIS) transaction has been completed by the Transport layer.

When in this state, the Transport layer waits for the Application layer to indicate that an FIS is to be transmitted or the Link layer to indicate that an FIS is being received.

**Transition DTI0:1:** When the Application layer indicates that a register FIS is to be transmitted, the Transport layer shall make a transition to the DTR0: DT_RegDHFIS state.

**Transition DTI0:2:** When the Application layer indicates that a Set Device Bits FIS is to be transmitted, the Transport layer shall make a transition to the DTDB0:DT_DB_FIS state.

**Transition DTI0:3:** When the Application layer indicates that a PIO Setup FIS is to be transmitted, the Transport layer shall make a transition to the DTPIOSTUP0: DT_PIOSTUPFIS state.

**Transition DTI0:4:** When the Application layer indicates that a DMA Activate FIS is to be transmitted, the Transport layer shall make a transition to the DTDMAACT0: DT_DMAACTFIS state.

**Transition DTI0:5:** When the Application layer indicates that a First-party DMA Setup FIS is to be transmitted, the Transport layer shall make a transition to the DTDMASTUP0: DT_DMASTUPDHFIS state.

**Transition DTI0:6:** When the Application layer indicates that a Data FIS is to be transmitted, the Transport layer shall make a transition to the DTDATAI0: DT_DATAIFIS state.

**Transition DTI0:7**: When the Application layer indicates that a BIST Activate FIS is to be transmitted, the Transport layer shall make a transition to the DTXBIST1:DT XmitBIST state.

**Transition DTI0:8:** When the Link layer indicates that an FIS is being received, the Transport layer shall make a transition to the DTI1: DT_ChkTyp state.

**DTI1: DT_ChkTyp state:** This state is entered when the Transport layer is idle and Link layer indicates that an FIS is being received.

When in this state, the Transport layer checks the FIS type of the incoming FIS.

**Transition DTI1:1:** When the incoming FIS is a Register – Host to Device FIS type, the Transport layer shall make a transition to the DTCMD0: DT_RegHDFIS state.

**Transition DTI1:2:** When the incoming FIS is a Data - Host to Device FIS type, the Transport layer shall make a transition to the DTDATAO0: DT_DATAOFIS state.

**Transition DTI1:3:** When the incoming FIS is a DMA Setup FIS type, the Transport layer shall make a transition to the DTSTP0: DT_ DMASTUPHDFIS state.

**Transition DTI1:4:** When the incoming FIS is a BIST Activate FIS type, the Transport layer shall make a transition to the DTRBIST1:DT Rcv BIST state.

**Transition DTI1:5:** When the Transport layer receives notification from the Link layer of an illegal state transition or the FIS type is not recognized, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

### 8.7.2 Device Transport send Register – Device to Host state diagram

This protocol builds a Register – Device to Host FIS that contains the register content and sends it to the host when the Application layer requests the transmission.

| DTR0: DT_RegDHFIS | Construct Register – Device to Host FIS from the content of the registers and notify Link to transfer. | | |
|---|---|---|---|
| | 1. FIS transfer complete | → | DT_TransStatus |
| | 2. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

| DTR1: DT_TransStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. | | |
|---|---|---|---|
| | 1. Status checked, and no error detected. | → | DT_DeviceIdle |
| | 2. Status checked, and error detected. | → | DT_RegDHFIS |

**DTR0: DT_RegDHFIS state:** This state is entered the Application requests the transmission of a Register – Device to Host FIS.

When in this state, the Transport layer shall construct a Register – Device to Host FIS, notify the Link layer that the FIS is to be transmitted, and pass the FIS to the Link layer. The Transport layer shall not perform flow control on the FIS transmission.

**Transition DTR0:1:** When the entire FIS has been passed to the Link layer, the Transport layer shall indicate to the Link layer that the FIS transmission is complete and make a transition to the DTR1: DT_TransStatus state.

**Transition DTR0:2:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

**DTR1: DT_TransStatus state:** This state is when the entire FIS has been passed to the Link layer.

When in this state, the Transport layer shall wait for the Link and Phy ending status for the FIS and take appropriate error handling action if required.

**Transition DTR1:1:** When the FIS status has been handled, and no error detected, the Transport layer shall transition to the DTI0: DT_DeviceIdle state.

**Transition DTR1:2**: When the FIS status has been handled, and an error detected, the Transport layer shall report status to the Link layer, and retry this transfer by transitioning to the DT_RegDHFISstate.

### 8.7.3 Device Transport send Set Device Bits FIS state diagram

This protocol sends a Set Device Bits FIS to the host adapter when the Application layer requests the transmission.

| DTDB0: DT_DB_FIS | Inform Link to transmit Set Device Bits FIS | | |
|---|---|---|---|
| | 1. FIS transfer complete | → | DT_TransStatus |

| DTDB1: DT_TransStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. | | |
|---|---|---|---|
| | 1. Status checked and no error detected | → | DT_DeviceIdle |
| | 2. Status checked and error detected. | → | DT_DB_FIS |

**DTDB0:DT_DB_FIS state:** This state is entered when the Application layer requests the transmission of a Set Device Bits FIS.

When in this state, the Transport layer shall construct a Set Device Bits FIS, notify the Link layer that the FIS is to be transmitted, and pass the FIS to the Link layer. The Transport layer shall not perform flow control on the FIS transmission.

**Transition DTDB0:1:** When the entire FIS has been passed to the Link layer, the Transport layer shall indicate to the Link layer that the FIS transmission is complete and make a transition to the DTDB1:DT_TransStatus state.

**DTDB1:DT_TransStatus state**: This state is entered when the entire FIS has been passed to the Link layer.

When in this state, the Transport layer shall wait for the Link and Phy ending status for the FIS and take appropriate error handling action if required.

**Transition DTDB1:1:** When the FIS status has been handled and no error detected, the Transport layer shall transition to the DTI0:DT_DeviceIdle state.

**Transition DTDB1:2:** When the FIS status has been handled and an error detected, the Transport layer shall report status to the Link layer and retry this transfer by transitioning to the DTDB0:DT_DB_FIS state.

### 8.7.4 Device Transport transmit PIO Setup – Device to Host FIS state diagram

This protocol transmits a PIO Setup – Device to Host FIS to the host. Following this PIO Setup frame, a single data frame containing PIO data shall be transmitted or received depending on the state of the D bit in the PIO Setup frame. The Transport layer shall not perform flow control on the PIO Setup FIS.

| DTPIOSTUP0: DT_PIOSTUPFIS | Construct PIO Setup – Device to Host FIS from the content provided by the Application layer and notify Link to transfer. This FIS shall include the beginning and ending register content, the byte count, and the interrupt flag. |
|---|---|

| | | |
|---|---|---|
| 1. FIS transfer complete | → | DT_TransStatus |
| 2. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

| DTPIOSTUP1: DT_TransStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. |
|---|---|

| | | |
|---|---|---|
| 1. Status checked, and no error detected. | → | DT_Device Idle |
| 2. Status checked, and error detected. | → | DT_PIOSTUPFIS |

**DTPIOSTUP0: DT_PIOSTUPFIS state:** This state is entered the Application layer requests the transmission of a PIO Setup – Device to Host FIS.

When in this state, the Transport layer shall construct a PIO Setup – Device to Host FIS, notify the Link layer that the FIS is to be transmitted, and pass the FIS to the Link layer. The Transport layer shall not perform flow control on a register FIS.

**Transition DTPIOSTUP0:1:** When the entire FIS has been passed to the Link layer, the Transport layer shall indicate to the Link layer that the FIS transmission is complete and make a transition to the DTPIOSTUP1: DT_TransStatus state.

**Transition DTPIOSTUP0:2:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

**DTPIOSTUP1: DT_TransStatus state:** This state is when the entire FIS has been passed to the Link layer.

When in this state, the Transport layer shall wait for the Link and Phy ending status for the FIS and take appropriate error handling action if required.

**Transition DTPIOSTUP1:1:** When the FIS status has been handled and no error detected, the Transport layer shall transition to the DTI0: DT_DeviceIdle state.

**Transition DTPIOSTUP1**:**2**: When the FIS status has been handled and an error detected, the Transport layer shall report status to the Link layer and retry this transfer by transitioning to the DT_PIOSTUPFIS state.

### 8.7.5 Device Transport transmit Legacy DMA Activate FIS state diagram

This protocol transmits a Legacy DMA Activate FIS to the host adapter. Following this Legacy DMA Activate frame, a data frame of DMA data shall be sent from the host to the device. The Transport layer shall not perform flow control on the DMA Activate FIS.

| DTDMAACT0: DT_DMAACTFIS | Construct Legacy DMA Activate FIS from the content provided by the Application layer and notify Link to transfer. | |
|---|---|---|
| 1. FIS transfer complete | → | DT_TransStatus |
| 2. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

| DTDMAACT1: DT_TransStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. | |
|---|---|---|
| 1. Status checked, and no error detected. | → | DT_DeviceIdle |
| 2. Status checked, and error detected. | → | DT_DMAACTFIS |

**DTDMAACT0: DT_DMAACTFIS state:** This state is entered the Application requests the transmission of a Legacy DMA Activate FIS.

When in this state, the Transport layer shall construct a Legacy DMA Activate FIS, notify the Link layer that the FIS is to be transmitted, and pass the FIS to the Link layer. The Transport layer shall not perform flow control on a register FIS.

**Transition DTDMAACT0:1:** When the entire FIS has been passed to the Link layer, the Transport layer shall indicate to the Link layer that the FIS transmission is complete and make a transition to the DTDMAACT1: DT_TransStatus state.

**Transition DTDMAACT0:2:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

**DTDMAACT1: DT_TransStatus state:** This state entered is when the entire FIS has been passed to the Link layer.

When in this state, the Transport layer shall wait for the Link and Phy ending status for the FIS and take appropriate error handling action if required.

**Transition DTDMAACT1:1:** When the FIS status has been handled and no error detected, the Transport layer shall transition to the DTI0: DT_DeviceIdle state.

**Transition DTDMAACT1**:**2**: When the FIS status has been handled and an error detected, the Transport layer shall report status to the Link layer and retry this transfer by transitioning to the DT_DMAACTFIS state.

### 8.7.6 Device Transport transmit First-party DMA Setup – Device to Host FIS state diagram

This protocol transmits a First-party DMA Setup – Device to Host FIS to the host adapter. This FIS is a request by the device for the host adapter to program the DMA controller for a First-party DMA transfer and is followed by one or more Data FIS's that transfer the data to or from the host adapter depending on the direction of the transfer. The First-party DMA Setup – Device to Host request includes the transfer direction indicator, the host buffer identifier, the host buffer offset, the byte count, and the interrupt flag. The Transport layer shall not perform flow control on the First-party DMA Setup – Device to Host FIS.

| DTDMASTUP0: DT_DMASTUPDHFIS | Construct the First-party DMA Setup – Device to Host FIS from the content provided by the Application layer and notify Link to transfer. | |
|---|---|---|
| 1. FIS transfer complete | → | DT_TransStatus |
| 2. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

| DTDMASTUP1: DT_TransStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. | |
|---|---|---|
| 1. Status checked, and no error detected. | → | DT_DeviceIdle |
| 2. Status checked, and error detected. | → | DT_DMASTUPFIS |

**DTDMASTUP0: DT_DMASTUPDHFIS state:** This state is entered when the Application requests the transmission of a First-party DMA Setup – Device to Host FIS.

When in this state, the Transport layer shall construct a First-party DMA Setup – Device to Host FIS, notify the Link layer that the FIS is to be transmitted, and pass the FIS to the Link layer. The Transport layer shall not perform flow control on a First-party DMA Setup – Device to Host FIS.

**Transition DTDMASTUP0:1:** When the entire FIS has been passed to the Link layer, the Transport layer shall indicate to the Link layer that the FIS transmission is complete and make a transition to the DTDMASTUP1: DT_TransStatus state.

**Transition DTIDMASTUP0:2:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

**DTPDMASTUP1: DT_TransStatus state:** This state is entered when the entire FIS has been passed to the Link layer.

When in this state, the Transport layer shall wait for the Link and Phy ending status for the FIS and take appropriate error handling action if required.

**Transition DTDMASTUP1:1:** When the FIS status has been handled and no error detected, the Transport layer shall transition to the DTI0: DT_DeviceIdle state.

**Transition DTDMASTUP1:2:** When the FIS status has been handled and an error detected, the Transport layer shall report status to the Link layer and retry this transfer by transitioning to the DT_DMASTUPFIS state.

### 8.7.7 Device Transport transmit Data – Device to Host FIS diagram

This protocol builds a Data – Device to Host FIS.

| DTDATAI0: DT_DATAIFIS | Construct Data – Device to Host FIS content from the content provided by the Application layer and notify Link to transfer. | |
|---|---|---|
| 1. Unconditional | → | DT_DATAITrans |

| DTDATAI1: DT_DATAITrans | Pass data Dwords from data FIFO to Link layer | |
|---|---|---|
| 1. Transfer not complete | → | DT_DATAITrans |
| 2. Transfer complete | → | DT_DATAIEnd |
| 3. Application layer requests termination of DMA in transfer. | → | DT_DATAIEnd |
| 4. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

| DTDATAI2: DT_DATAIEnd | Check Link and Phy transmission results and if an error occurred take appropriate action. | |
|---|---|---|
| 1. Status checked, and no error detected. | → | DT_DeviceIdle |
| 2. Status checked, and error detected. | → | DT_DeviceIdle |
| 3. Application layer requests termination of DMA in transfer, no error detected. | → | DT_DeviceIdle |
| 4. Application layer requests termination of DMA in transfer, error detected. | → | DT_DeviceIdle |
| 5. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

**DTDATAI0: DT_DATAFIS state:** This state is entered when the Application layer has requested the transmission of a Data – Device to Host FIS.

When in this state the Transport layer shall pass a portion of the DMA data to the Link layer.

**Transition DTDATAI0:1:** When ready and there is data in the FIFO to be passed to the Link layer, the Transport layer shall transition to the DTDATAI1: DT_DATAITrans state.

**DTDATAI1: DT_DATAITrans state:** This state is entered when data is available in the FIFO to be passed the Link layer.

When in this state, the Transport layer shall pass a Dword of data from the FIFO to the Link layer.

**Transition DTDATAI1:1:** When the transfer is not complete, the Transport layer shall transition to the DTDATAI1: DT_DATAITrans state.

**Transition DTDATAI1:2:** When the transfer is complete, the Transport layer shall transition to the DTDATAI2: DT_DATAIEnd state.

**Transition DTDATAI1:3:** When the Application layer requests that a DMA operation is to be aborted, the Transport Layer shall transition to the DTDATAI2:DT_DATAIEnd state.

**Transition DTDATAI1:4:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

**DTDATAI2: DT_DATAIEnd state:** This state is entered when the data transfer is complete or an abort has been requested by the application layer.

When in this state, the Transport layer shall wait for the Link and Phy ending status for the FIS and take appropriate error handling action if required.

**Transition DTDATAI2:1:** When the FIS status has been handled and no error detected, the Transport layer shall transition to the DTI0: DT_DeviceIdle state.

**Transition DTDATAI2:2:** When the FIS status has been handled and an error detected, status shall be reported to the Link layer. The Transport layer shall transition to the DTI0: DT_DeviceIdle state.

**Transition DTDATAI2:3:** When the application layer requests the termination of a DMA data in transaction, it will report the abort condition to the Link, wait for an EOF and, if no error is detected, make a transition to the DTI0:DT_DeviceIdle state.

**Transition DTDATAI2:4:** When the application layer requests the termination of a DMA data in transaction, it will report the abort condition to the Link, wait for an EOF, and if an error is detected, report the error to the Link layer, and make a transition to the DTI0:DT_DeviceIdle state.

**Transition DTDATAI2:5:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

### 8.7.8 Device Transport transmit BIST Activate FIS diagram

This protocol builds a BIST Activate FIS that tells the host to prepare to enter the appropriate Built-in Self Test mode. After successful transmission, the device Transport layer enters the idle state. The application layer, upon detecting successful transmission to the host shall then cause the device's Transport layer, Link layer and Physical layer to enter the appropriate mode for the transmission of the Built-in Test data defined by the FIS. The means by which the Transport, Link and Physical layers are placed into self-test mode are not defined by this specification.

| DTXBIST1: DT_XmitBIST | Construct the BIST Activate FIS from the content provided by the Application layer and notifies Link to transfer. | | |
|---|---|---|---|
| | 1. FIS transfer complete | → | DT_TransBISTStatus |
| | 2. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

| DTXBIST2: DT_TransBISTStatus | Check Link and Phy transmission results and if an error occurred take appropriate action. | | |
|---|---|---|---|
| | 1. Status checked and no errors detected | → | DT_DeviceIdle |
| | 2. Status check and at least one error detected | → | DT_XmitBIST |

**DTXBIST1: DT_XmitBIST state:** This state is entered to send a BIST FIS to the host.

**Transition DTXBIST1:1:** When the entire FIS has been passed to the Link layer, the Transport layer shall indicate to the Link layer that the FIS transmission is complete and make a transition to the DTXBIST2:DT_TransBISTSTatus state.

**Transition DTXBIST1:2:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

**DTXBIST2: DT_TransBISTStatus state:** This state is entered when the entire FIS has been passed to the Link layer.

**Transition DTXBIST2:1:** When the FIS transmission is completed and no errors have been detected, the Transport layer shall transition to the DTI0: DT_DeviceIdle state.

**Transition DTXBIST2:2:** When the FIS transmission is completed and at least one error is detected, the Transport layer shall transition to the DTXBIST1: DT_XmitBIST state.

### 8.7.9 Device Transport decompose Register – Host to Device state diagram

This protocol receives a Register – Host to Device FIS, places received register content into the device registers, and notifies the Application layer of the FIS receipt. The Transport layer shall not flow control the receipt of this FIS.

| DTCMD0: DT_RegHDFIS | Receive a Register – Host to Device FIS | |
|---|---|---|
| 1. FIS transfer complete, and no error detected. | → | DT_DeviceIdle |
| 2. FIS transfer complete, and error detected | → | DT_DeviceIdle |
| 3. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

**DTCMD0: DT_RegHDFIS state:** This state is entered when the receipt of a DT_RegHDFISFIS is recognized.

When in this state, the Transport layer shall receive the FIS and place the contents of the FIS into the device registers when it is determined that the FIS was received without error. The Transport layer shall not perform flow control on a command FIS.

**Transition DTCMD0:1:** When the entire FIS has been received from the Link layer without error, the Transport layer shall indicate to the Application layer that a command FIS was received and make a transition to the DTI0: DT_DeviceIdle state.

**Transition DTCMD0:2:** When the entire FIS has been received from the Link layer and an error has been detected, status shall be sent to the Link layer. The Transport layer shall make a transition to the DTI0:DT_DeviceIdle state.

**Transition DTCMD0:3:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

### 8.7.10 Device Transport decompose Data (Host to Device) FIS state diagram

This protocol receives a Data - Host to Device FIS.

| DTDATAO0: DT_DATAOFIS | Prepare to receive data | | |
|---|---|---|---|
| | Unconditional | → | DT_DATAOREC |

| DTDATAO1: DT_DATAOREC | Place received data Dword into data FIFO and signal Link to continue transfer. | | |
|---|---|---|---|
| | 1. Transfer not complete | → | DT_DATAOREC |
| | 2. Transfer complete | → | DT_DeviceIdle |
| | 3. Abort Transfer from Application layer | → | DT_DeviceAbort |
| | 4. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

| DTDATAO2: DT_DeviceAbort | Signal Link to abort transfer. | | |
|---|---|---|---|
| | 1. Transfer not complete | → | DT_DATAOREC |
| | 2. Transfer complete | → | DT_ DeviceIdle |

**DTDATAO0: DT_DATAOFIS state:** This state is entered when the Link layer has indicated that an FIS is being received and that the Transport layer has determined the FIS is of Data - Host to Device type.

When in this state, the Transport layer shall prepare to receive the data.

**Transition DTDATAO0:1:** When ready to receive the data, the Transport layer shall make a transition to the DTDATAO1: DT_DATAOREC state.

**DTDATAO1: DT_DATAOREC state:** This state is entered when the Transport layer is ready to receive the data.

When in this state, the Transport layer shall wait for the Link layer to indicate the transfer is complete.

**Transition DTDATAO1:1:** When Link layer has not indicated that the end of the FIS has been reached, the Transport layer shall transition to the DTDATAO1: DT_DATAOREC state.

**Transition DTDATAO1:2:** When the Link layer indicates that the end of the FIS has been reached, the Transport layer shall transition to the DTI0: DT_DeviceIdle state.

**Transition DTDATAO1:3:** When the Application layer indicates that the FIS is to be aborted, the Transport layer shall transition to the DTDATAO2: DT_DeviceAbort state.

**Transition DTDATAO1:4:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

**DTDATAO2: DT_DeviceAbort state:** This state is entered when the application layer indicates that the current transfer is to be aborted.

When in this state, the Transport layer shall signal the Link layer to Abort the incoming transmission and return to either DT_DATAOREC or DT_DATAOhold, depending upon the current state of the

FIFO. If the abort occurs coincident with an end of transfer indication from the Link, then the transition to DTI0: DT_DeviceIdle is also accommodated. After issuing an Abort, the Transport returns to normal data transfer, and awaits the end of transfer indication from the Link.

**Transition DTDATAO2:1:** Inform Link layer to issue an abort. When Transfer is not complete, the Transport layer shall transition to the DTDATAO1: DT_DATAOREC state.

**Transition DTDATAO2:2:** Inform Link layer to issue an abort. When the Link layer indicates that the end of the FIS has been reached, the Transport layer shall transition to the DTI0: DT_DeviceIdle state.

### 8.7.11 Device Transport decompose DMA Setup – Host to Device or Device to Host state diagram

This protocol receives a DMA Setup – Host to Device or Device to Host FIS, passes received DMA Setup content, and notification of FIS receipt to the Application layer. The Transport layer shall not flow control the receipt of this FIS.

| DTSTP0: DT_DMASTUPHDFIS | Receive a Register – Host to Device FIS | |
|---|---|---|
| 1. FIS transfer complete, and no error detected. | → | DT_DeviceIdle |
| 2. FIS transfer complete, and error detected | → | DT_DeviceIdle |
| 3. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

**DTSTP0: DT_DMASTUPHDFIS state:** This state is entered when the receipt of a DT_DMASTUP FIS is recognized.

**Transition DTSTP0:1:** When the entire FIS has been received from the Link layer without error, the Transport layer shall indicate to the Application layer that a command FIS was received and make a transition to the DTI0: DT_DeviceIdle state.

**Transition DTSTP0:2:** When the entire FIS has been received from the Link layer and an error has been detected, status shall be sent to the link layer. The Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

**Transition DTSTP0:3:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

### 8.7.12 Device Transport decompose a BIST Activate FIS state diagram

This protocol receives an FIS that instructs the device to enter one of several Built-in Self-test modes that cause the device to retransmit the data it receives. If the mode is supported the Device's application layer will place both the transmit and receive portions of the Transport, Link and/or Physical layers into appropriate state to perform the loopback operation.

| DTRBIST1: DT_RcvBIST | Determine validity of loopback mode requested. | |
|---|---|---|
| 1. Status checked, no error detected and Loopback mode valid | → | DT_BISTTrans1 |
| 2. Status checked, no error detected and Loopback mode is invalid or not supported. | → | DT_DeviceIdle |
| 3. Status checked and error detected | → | DT_DeviceIdle |
| 4. Notification of illegal transition error received from Link layer | → | DT_DeviceIdle |

| DTRBIST2: DT_BISTTrans1 | Notify application layer of desired BIST modes | |
|---|---|---|
| 1. Unconditional | → | DT_DeviceIdle |

**DTRBIST1: DT_RcvBIST state**: This state is entered when the Link layer has indicated that an FIS is being received and the Transport layer has determined that a BIST Activate FIS is being received.

When in this state, the Transport layer shall determine the validity of the loopback request.

**Transition DTRBIST1:1:** If no reception error is detected and the FIS contents indicate a form of loopback request that is supported by the host the Transport layer shall make a transition to the DTRBIST2: DT_BISTTrans1 state.

**Transition DTRBIST1:2:** If no reception error is detected and the FIS contents indicate a form of loopback request that is not supported by the host the Transport layer shall make a transition to the DTI0:DT_DeviceIdle state.

**Transition DTRBIST1:3:** If a reception error is indicated the Transport layer shall make a transition to the DTI0:DT_DeviceIdle state.

**Transition DTRBIST1:4:** When the Transport layer receives notification from the Link layer of an illegal state transition, the Transport layer shall make a transition to the DTI0: DT_DeviceIdle state.

**DTRBIST2: DT_BISTTrans1 state:** This state is entered when the Transport layer has determined that a valid BIST Activate FIS has been received.

Having received a valid FIS, the Transport layer informs the application layer that it should place the Transport, Link and Physical layers into the appropriate modes to loop the received data back to the transmitter. The method by which this is performed is not defined by this specification.

**Transition DTRBIST2:1:** When the Application layer has been notified the Transport layer shall transition to the DTI0:DT_DeviceIdle state.

# 9 Device command layer protocol

## 9.1 Power-on and COMRESET protocol diagram

If the host asserts Hard Reset regardless of the power management mode or the current device command layer state, the device shall execute the hardware reset protocol.

| DHR0: Hardware_reset_asserted | Wait. | | |
|---|---|---|---|
| | 1. COMRESET negated. | → | DHR1: Execute_diagnostics |

| DHR1: Execute_diagnostics | Initialize hardware and execute diagnostics. | | |
|---|---|---|---|
| | 1. Initialization and diagnostics completed successfully. | → | DHR2: Send_good_status |
| | 2. Initialization completed and diagnostics failed. | → | DHR3: Send_bad_status |

| DHR2: Send_good_status | Request transmission of Register FIS with good status. | | |
|---|---|---|---|
| | 1. Register FIS transmitted. | → | DI0: Device_idle |

| DHR3: Send_bad_status | Request transmission of Register FIS with bad status. | | |
|---|---|---|---|
| | 1. Register FIS transmitted. | → | DI0: Device_idle |

**DHR0: Hardware_reset_asserted:** This state is entered when the Transport layer indicates that the COMRESET signal is asserted.

When in this state, the device awaits the negation of the COMRESET signal.

**Transition DHR0:1:** When the Transport layer indicates that the COMRESET signal has been negated, the device shall transition to the DHR1: Execute_diagnostics state.

**DHR1: Execute_diagnostics:** This state is entered when the Transport layer indicates that the COMRESET signal has been negated.

When in this state, the device initializes the device hardware and executes its power-up diagnostics.

**Transition DHR1:1:** When the device hardware has been initialized and the power-up diagnostics successfully completed, the device shall transition to the DHR2: Send_good_status state.

**Transition DHR1:2:** When the device hardware has been initialized and the power-up diagnostics failed, the device shall transition to the DHR3: Send_bad_status state.

**DHR2: Send_good_status:** This state is entered when the device hardware has been initialized and the power-up diagnostics successfully completed.

When in this state, the device requests that the Transport layer transmit a Register FIS to the host. If the device does not implement the PACKET command feature set the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 00h |
| Cylinder High | 00h |
| Device/Head | 00h |
| Error | 01h |
| Status | 00h-70h (see note) |
| | |
| NOTE – Setting of bit 6 thru 4 in the Status register are device specific. | |

If the device implements the PACKET command feature set, the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 14h |
| Cylinder High | EBh |
| Device/Head | 00h |
| Error | 01h |
| Status | 00h |
| | |

**Transition DHR2:1:** When the Transport layer indicates that the Register FIS has been transmitted, the device shall transition to the DI0: Device_Idle state.

**DHR3: Send_bad_status:** This state is entered when the device hardware has been initialized and the power-up diagnostics failed.

When in this state, the device requests that the Transport layer transmit a Register FIS to the host. If the device does not implement the PACKET command feature set the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 00h |
| Cylinder High | 00h |
| Device/Head | 00h |
| Error | 00h, 02h-7Fh |
| Status | 00h-70h (see note) |
| | |
| NOTE – Setting of bit 6 thru 4 in the Status register are device specific. | |

If the device implements the PACKET command feature set, the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 14h |
| Cylinder High | EBh |
| Device/Head | 00h |
| Error | 00h, 02h-7Fh |
| Status | 00h |
| | |

**Transition DHR3:1:** When the Transport layer indicates that the Register FIS has been transmitted, the device shall transition to the DI0: Device_Idle state.

## 9.2 Device Idle protocol

The state diagram below describes the idle protocol for a device. States and transitions preceded by an * are utilized only when queuing is implemented.

| DI0: Device_idle | Wait. | |
|---|---|---|
| 1. FIS receipt | → | DI1: Check_FIS |
| 2. * Ready to complete released command. | → | DI4: Set_service |

| DI1: Check_FIS | Check_FIS type and C bit. | |
|---|---|---|
| 1. Register type, C bit cleared, and SRST set. | → | DSR0: Software_reset_ asserted |
| 2. Register type, C bit cleared, and SRST cleared | → | DI0: Device_idle |
| 3. Register type and C bit set. | → | DI2: Check_command |
| 4. First Party Setup FIS Received | → | DI0 : Device_idle |
| 5. Unexpected FIS type. | → | DI0: Device_idle |

| DI2: Check_command[1] | Check the command to determine required command protocol. | | |
|---|---|---|---|
| | 1. Non-data command protocol. | → | DND0: Non-data |
| | 2. PIO data-in command protocol. | → | DPIOI0: PIO_in |
| | 3. PIO data-out command protocol. | → | DPIOO0: PIO_out |
| | 4. READ DMA command protocol. (legacy) | → | DDMAI0: DMA_in |
| | 5. WRITE DMA command protocol. (legacy) | → | DDMAO0: DMA_out |
| | 6. PACKET command protocol. | → | DP0: PACKET |
| | 7. READ DMA QUEUED command protocol. (legacy) | → | DDMAQI0: DMA_queued_in |
| | 8. WRITE DMA QUEUED command protocol. (legacy) | → | DDMAQIO: DMA_queued_out |
| | 9. EXECUTE DEVICE DIAGNOSTIC command protocol. | → | DEDD0: Execute_device_diag |
| | 10. DEVICE RESET command protocol. | → | DDR0: Device_reset |
| | 11. Command not implemented. | → | DI3: No_command |
| | 12. * SERVICE command protocol | → | DI5: Service |
| | NOTE:<br>1. This state shows transitions for all commands. If a device does not implement any particular command, then that transition should not be processed. | | |

| DI3: No_command | Request transmission of Register FIS with ABRT error. | | |
|---|---|---|---|
| | 1. FIS transmission complete. | → | DI0: Device_idle |

| * DI4: Set_service | Request transmission of Set Device Bits FIS with SERV set. | | |
|---|---|---|---|
| | 1. FIS transmission complete. | → | DI0: Device_idle |

| * DI5: Service_test | Test command to see if Register FIS is needed to set DRQ and set Tag. | | |
|---|---|---|---|
| | 1. PACKET PIO data-in or PACKET PIO data-out. | → | DI7: Service_decode |
| | 2. Other | → | DI6: Service_send_tag |

| * DI6: Service_send_tag | Request transmission of Register FIS with BSY=0, DRQ=1, and SC=Tag | | |
|---|---|---|---|
| | 1. FIS transmission complete | → | DI7: Service_decode |

| * DI7: Service_decode | Check command type to be serviced. | |
|---|---|---|
| 1. PACKET PIO data-in. | → | DP4: PACKET_PIO_in |
| 2. PACKET PIO data-out. | → | DP6: PACKET_PIO_out |
| 3. PACKET DMA data-in. | → | DP9: PACKET_DMA_in |
| 4. PACKET DMA data-out. | → | DP11: PACKET_DMA_out |
| 5. READ DMA QUEUED. | → | DDMAQI1: Send_data |
| 6. WRITE DMA QUEUED. | → | DDMAQO1: Send_DMA_activate |

**DI0: Device_Idle:** This state is entered when the device has completed the execution of a command protocol, a COMRESET protocol, a software reset protocol, or a queued command has been released.

When in this state, the device is awaiting a command. If queuing is supported, the device may be waiting to acquire data or establish buffer space to complete a queued command

**Transition DI0:1:** When the device receives an FIS from the Transport layer, the device shall transition to the DI1: Check_FIS state.

*** Transition DI0:2:** When the device is ready to complete the data transfer for a queued command, the device shall transition to the DI4: Set_service state.

**DI1: Check_FIS state:** This state is entered when the device receives an FIS from the Transport layer.

When in this state, the device shall check the FIS type.

**Transition DI1:1:** If the FIS type is a Register FIS, the C bit in the FIS is cleared, and the SRST bit in the FIS is set, the device shall transition to the DSR0: Software_reset_asserted state.

**Transition DI1:2:** If the FIS type is a Register FIS, the C bit in the FIS is cleared, and the SRST bit in the FIS is cleared, the device shall transition to the DI0: Device_idle state.

**Transition DI1:3:** If the FIS type is a Register FIS and the C bit in the FIS is set, the device shall transition to the DI2: Check_command state.

**Transition DI1:4:** If the FIS type is a First Party DMA Setup FIS, the device shall inform the Transport layer of the reception of the First Party DMA Setup FIS, and transition to the DI0: Device_idle state.

**Transition DI1:5:** For any other FIS, the device shall transition to the DI0: Device_idle state.

**DI2: Check_command state:** This state is entered when the device recognizes that the received Register FIS contains a new command. NOTE: This state shows transitions for all commands. If a device does not implement any particular command, then transition DI2:11 to state DI3:No_command shall be made.

When in this state, the device shall check the command protocol required by the received command.

**Transition DI2:1:** When the received command is a non-data transfer command, the device shall transition to the DND0: Non-data state.

**Transition DI2:2:** When the received command is a PIO data-in command, the device shall transition to the DPIOI0: PIO_in state.

**Transition DI2:3:** When the received command is a PIO data-out command, the device shall transition to the DPIOO0: PIO_out state.

**Transition DI2:4:** When the received command is a READ DMA command, the device shall transition to the DDMAI0: DMA_in state.

**Transition DI2:5:** When the received command is a WRITE DMA command, the device shall transition to the DDMAO0: DMA_out state.

**Transition DI2:6:** When the received command is a PACKET command, the device shall transition to the DP0: PACKET state.

**Transition DI2:7:** When the received command is a READ DMA QUEUED command, the device shall transition to the DDMAQI0: DMA_queued_in state.

**Transition DI2:8:** When the received command is a WRITE DMA QUEUED command, the device shall transition to the DDMAQO0: DMA_queued_out state.

**Transition DI2:9:** When the received command is an EXECUTE DEVICE DIAGNOSTICS command, the device shall transition to the DEDD0: Execute_device_diag state.

**Transition DI2:10:** When the received command is an RESET DEVICE command, the device shall transition to the DDR0: Device_reset state.

**Transition DI2:11:** When the received command is not implemented by the device, the device shall transition to the DI3: No_command state.

**Transition DI2:12:** When the received command is a SERVICE command, the device shall transition to the DI5: Service state.

**DI3: No_command state:** This state is entered when the device recognizes that the received command is not implemented by the device.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register content as described in the command description in ATA/ATAP-5 (T13/d1321R2) clause 8 and the I bit set to one.

**Transition DI2:1:** When the Transport layer has transmitted the Register FIS, the device shall transition to the DI0: Device_idle state.

*** DI4: Set_service state:** This state is entered when the ready to complete the data transfer for a queued command.

When in this state, the device shall request that the Transport layer transmit a Set Device Bits FIS with the SERV bit set in the Status register and with all other bits in the Error and Status fields the same as the current contents of the respective registers, and the I bit set to one.

**Transition DI4:1:** When the Transport layer has transmitted the Set Device Bits FIS, the device shall transition to the DI0: Device_idle state.

*** DI5: Service_test state:** This state is entered when the SERVICE command has been received.

When in this state, the device shall determine the type of command that the device has requested service to complete. The PACKET_PIO command will provide its own register update to set DRQ and send the command tag, but other queued commands require a Register FIS.

**Transition DI5:1:** When the command to be serviced is a PIO data-in or PIO data-out command, the device shall transition to the DI7: Service_decode state.

**Transition DI5:2:** When the command to be serviced is neither a PIO data-in nor a PIO data-out command, the device shall transition to the DI6: Service_send_tag state.

*** DI6: Service_send_tag state:** This state is entered when the SERVICE command has been received and sending a Register Device to Host FIS is necessary for the command being serviced.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register contents, including the desired command tag, as described in the command description of the ATA/ATAPI standard for the command being serviced.

**Transition DI6:1:** When the Transport layer has transmitted the Register FIS, the device shall transition to theDI7: Service_decode state.

*** DI7: Service_decode state:** This state is entered when a register FIS has been transmitted, if necessary to send the register contents, including the desired command tag, in response to a SERVICE command.

When in this state, the device shall again determine the type of command that the device has requested service to complete, and branch to that command's data transfer and completion.

**Transition DI7:1:** When the command to be serviced is a PIO data-in command, the device shall transition to the DP4: PACKET_PIO_in state.

**Transition DI7:2:** When the command to be serviced is a PIO data-out command, the device shall transition to the DP6: PACKET_PIO_out state.

**Transition DI7:3:** When the command to be serviced is a DMA data-in command, the device shall transition to the DP9: PACKET_DMA_in state.

**Transition DI7:4:** When the command to be serviced is a DMA data-out command, the device shall transition to the DP11: PACKET_DMA_out state.

**Transition DI7:5:** When the command to be serviced is a READ DMA QUEUED command, the device shall transition to the DDMAQI1: Send_data state.

**Transition DI7:6:** When the command to be serviced is a WRITE DMA QUEUED command, the device shall transition to the DDMAQO1: Send_DMA_activate state.

## 9.3 Software reset protocol

When the host sends a Register FIS with a one in the SRST bit position of the Device Control register byte, regardless of the power management mode, the device shall execute the software reset protocol.

| DSR0: Software_reset_asserted | Begin initialization and diagnostic execution. | |
|---|---|---|
| 1. Software reset negated. | → | DSR1: Execute_diagnostics |
| 2. Software reset asserted. | → | DSR0: Software_reset_ asserted |

| DSR1: Execute_diagnostics | Complete Initialization and diagnostics execution. | |
|---|---|---|
| 1. Initialization and diagnostics completed successfully. | → | DSR2: Send_good_status |
| 2. Initialization completed and diagnostics failed. | → | DSR3: Send_bad_status |

| DSR2: Send_good_status | Request transmission of Register FIS with good status. | |
|---|---|---|
| 1. Register FIS transmitted. | → | DI0: Device_idle |

| DSR3: Send_bad_status | Request transmission of Register FIS with bad status. | |
|---|---|---|
| 1. Register FIS transmitted. | → | DI0: Device_idle |

**DSR0: Software_reset_asserted:** This state is entered when a Register FIS is received with the C bit in the FIS cleared to zero and the SRST bit set to one in the Device Control register.

When in this state, the device begins its initialization and diagnostics execution and awaits the clearing of the SRST bit.

**Transition DSR0:1:** When a Register FIS is received with the C bit in the FIS cleared to zero and the SRST bit cleared to zero in the Device Control register, the device shall transition to the DSR1: Execute_diagnostics state.

**Transition DSR0:2:** If a Register FIS is received with the C bit in the FIS set to one, or the SRST bit set to one in the Device Control register, the device shall transition to the DSR0: Software_reset_asserted state.

**DSR1: Execute_diagnostics:** This state is entered when a Register FIS is received with the C bit in the FIS cleared to zero and the SRST bit cleared to zero in the Device Control register.

When in this state, the device completes initialization and execution of its diagnostics.

**Transition DSR1:1:** When the device has been initialized and the diagnostics successfully completed, the device shall transition to the DSR2: Send_good_status state.

**Transition DSR1:2:** When the device has been initialized and the diagnostics failed, the device shall transition to the DSR3: Send_bad_status state.

**DSR2: Send_good_status:** This state is entered when the device has been initialized and the diagnostics successfully completed.

When in this state, the device requests that the Transport layer transmit a Register FIS to the host. If the device does not implement the PACKET command feature set the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 00h |
| Cylinder High | 00h |
| Device/Head | 00h |
| Error | 01h |
| Status | 00h-70h (see note) |
| NOTE – Setting of bit 6 thru 4 in the Status register are device specific. | |

If the device implements the PACKET command feature set, the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 14h |
| Cylinder High | EBh |
| Device/Head | 00h |
| Error | 01h |
| Status | 00h |

**Transition DSR2:1:** When the Transport layer indicates that the Register FIS has been transmitted, the device shall transition to the DI0: Device_Idle state.

**DSR3: Send_bad_status:** This state is entered when the device has been initialized and the diagnostics failed.

When in this state, the device requests that the Transport layer transmit a Register FIS to the host. If the device does not implement the PACKET command feature set the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 00h |
| Cylinder High | 00h |
| Device/Head | 00h |
| Error | 00h, 02h-7Fh |
| Status | 00h-70h (see note) |
| NOTE – Setting of bit 6 thru 4 in the Status register are device specific. | |

If the device implements the PACKET command feature set, the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 14h |
| Cylinder High | EBh |
| Device/Head | 00h |
| Error | 00h, 02h-7Fh |
| Status | 00h |

**Transition DSR3:1:** When the Transport layer indicates that the Register FIS has been transmitted, the device shall transition to the DI0: Device_Idle state.

## 9.4 EXECUTE DEVICE DIAGNOSTIC command protocol

This class includes:

- EXECUTE DEVICE DIAGNOSTIC

If the host sends COMRESET before the device has completed executing the EXECUTE DEVICE DIAGNOSTIC protocol, then the device shall immediately start executing the COMRESET protocol form the beginning. If the host asserts SRST in the Device Control register before the device has completed executing the EXECUTE DEVICE DIAGNOSTIC protocol, then the device shall immediately start executing its software reset protocol from the beginning.

| DEDD0: Execute_device_diag | Execute diagnostics. | |
|---|---|---|
| 1. Diagnostics completed successfully. | → | DEDD1: Send_good_status |
| 2. Diagnostics failed. | → | DEDD2: Send_bad_status |

| DEDD1: Send_good_status | Request transmission of Register FIS with good status. | |
|---|---|---|
| 1. Register FIS transmitted. | → | DI0: Device_idle |

| DEDD2: Send_bad_status | Request transmission of Register FIS with bad status. | |
|---|---|---|
| 1. Register FIS transmitted. | → | DI0: Device_idle |

**DEDD0: Execute_device_diag:** This state is entered when an EXECUTE DEVICE DIAGNOSTIC command is received.

When in this state, the device executes its diagnostics.

**Transition DEDD0:1:** When the device successfully completed the diagnostics, the device shall transition to the DEDD1: Send_good_status state.

**Transition DEDD1:2:** When the device has failed the diagnostics, the device shall transition to the DEDD2: Send_bad_status state.

**DEDD1: Send_good_status:** This state is entered when the device has successfully completed the diagnostics.

When in this state, the device shall request that the Transport layer transmit a Register FIS to the host, with the I bit set to one. If the device does not implement the PACKET command feature set the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 00h |
| Cylinder High | 00h |
| Device/Head | 00h |
| Error | 01h |
| Status | 00h-70h (see note) |
| NOTE – Setting of bit 6 thru 4 in the Status register are device specific. | |

If the device implements the PACKET command feature set, the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 14h |
| Cylinder High | EBh |
| Device/Head | 00h |
| Error | 01h |
| Status | 00h |

**Transition DEDD1:1:** When the Transport layer indicates that the Register FIS has been transmitted, the device shall transition to the DI0: Device_Idle state.

**DEDD2: Send_bad_status:** This state is entered when the device has been initialized and the diagnostics failed.

When in this state, the device shall request that the Transport layer transmit a Register FIS to the host, with the I bit set to one. If the device does not implement the PACKET command feature set the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 00h |
| Cylinder High | 00h |
| Device/Head | 00h |
| Error | 00h, 02h-7Fh |
| Status | 00h-70h (see note) |
| NOTE – Setting of bit 6 thru 4 in the Status register are device specific. | |

If the device implements the PACKET command feature set, the register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 14h |
| Cylinder High | EBh |
| Device/Head | 00h |
| Error | 00h, 02h-7Fh |
| Status | 00h |

**Transition DEDD2:1:** When the Transport layer indicates that the Register FIS has been transmitted, the device shall transition to the DI0: Device_Idle state.

## 9.5 DEVICE RESET command protocol

This class includes:

- DEVICE RESET

If the host sends COMRESET before the device has completed executing the DEVICE RESET protocol, then the device shall immediately start executing the COMRESET protocol from the beginning. If the host asserts SRST in the Device Control register before the device has completed executing the DEVICE RESET protocol, then the device shall immediately start executing its software reset protocol from the beginning.

| DDR0: Device_reset | Stop execution and background activity. | |
|---|---|---|
| 1. Activity ceased. | → | DDR1: Send_good_status |

| DDR1: Send_good_status | Request transmission of Register FIS with good status. | |
|---|---|---|
| 1. Register FIS transmitted. | → | DI0: Device_idle |

**DDR0: Device_reset:** This state is entered when an DEVICE RESET command is received.

When in this state, the device stops any execution or activity in progress.

**Transition DDR0:1:** When the device has ceased any execution or activity and has completed its internal diagnostics, the device shall transition to the DDR1: Send_good_status state.

**DDR1: Send_good_status:** This state is entered when the device has been initialized and the diagnostics successfully completed.

When in this state, the device requests that the Transport layer transmit a Register FIS to the host.

The register content shall be:

| | |
|---|---|
| Sector Count | 01h |
| Sector Number | 01h |
| Cylinder Low | 14h |
| Cylinder High | EBh |
| Device/Head | 00h |
| Error | 01h |
| Status | 00h |

**Transition DDR1:1:** When the Transport layer indicates that the Register FIS has been transmitted, the device shall transition to the DI0: Device_Idle state.

## 9.6 Non-data command protocol

This class includes:

- CFA ERASE SECTORS
- CFA REQUEST EXTENDED ERROR CODE
- CHECK POWER MODE
- FLUSH CACHE
- FLUSH CACHE EXT
- GET MEDIA STATUS
- IDLE
- IDLE IMMEDIATE
- INITIALIZE DEVICE PARAMETERS
- MEDIA EJECT
- MEDIA LOCK
- MEDIA UNLOCK

- NOP
- READ NATIVE MAX ADDRESS
- READ NATIVE MAX ADDRESS EXT
- READ VERIFY SECTOR(S)
- SECURITY ERASE PREPARE
- SECURITY FREEZE LOCK
- SEEK
- SET FEATURES
- SET MAX ADDRESS
- SET MAX ADDRESS EXT
- SET MULTIPLE MODE
- SLEEP
- SMART DISABLE OPERATION
- SMART ENABLE/DISABLE AUTOSAVE
- SMART ENABLE OPERATION
- SMART EXECUTE OFFLINE IMMEDIATE
- SMART RETURN STATUS
- STANDBY
- STANDBY IMMEDIATE

Execution of these commands involves no data transfer. See the NOP command description and the SLEEP command description for additional protocol requirements.

| DND0: Non-data | Execute Non-data command. | | |
|---|---|---|---|
| | 1. Command execution complete. | → | DND1: Send_status |

| DND1: Send_status | Request transmission of a Register FIS. | | |
|---|---|---|---|
| | 1. FIS transmission complete. | → | DI0: Device_idle |

**DND0: Non-data State:** This state is entered when a received command is a non-data command.

When in this state, the device shall execute the requested command if supported.

**Transition DND0: 1:** When command execution completes, the device shall transition to the DND1: Send_status state.

**DND1: Send_status State:** This state is entered when the execution of the non-data command has been completed.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register content as described in the command description in ATA/ATAP-5 (T13/d1321R2) clause 8 and the I bit set to one.

**Transition DND1:1:** When the FIS has been transmitted, then the device shall transition to the DI0: Device_idle state.

## 9.7 PIO data-in command protocol

This class includes:

- CFA TRANSLATE SECTOR
- IDENTIFY DEVICE
- IDENTIFY PACKET DEVICE
- READ BUFFER
- READ MULTIPLE
- READ MULTIPLE EXT
- READ SECTOR(S)
- READ SECTOR(S) EXT
- SMART READ DATA
- SMART READ LOG SECTOR

Execution of this class of command includes the PIO transfer of one or more blocks of data from the device to the host.

| DPIOI0: PIO_in | Prepare a DRQ data block for transfer to the host. | | |
|---|---|---|---|
| | 1. DRQ data block ready to transfer. | → | DPIOI1: Send_PIO_setup |
| | 2. Command aborted due to error. | → | DPIOI3: Error_status |

| DPIOI1: Send_PIO_setup | Request transmission of a PIO Setup FIS to host. | | |
|---|---|---|---|
| | 1. PIO Setup FIS transmitted. | → | DPIOI2: Transmit_data |

| DPIOI2: Transmit_data | Request transmission of a Data FIS to host. | | |
|---|---|---|---|
| | 1. Data FIS transmitted, no more data transfer required for this command. | → | DI0: Device_idle |
| | 2. Data FIS transmitted, more data transfer required for this command, or 8KB transmitted. | → | DPIOI0: PIO_in |

| DPIOI3: Error_status | Request transmission of a Register - Device to Host FIS. | | |
|---|---|---|---|
| | 1. FIS transmission complete. | → | DI0: Device_idle |

**DPIOI0: PIO_in State:** This state is entered when the device receives a PIO data-in command or the transmission of one or more additional DRQ data blocks is required to complete the command.

When in this state, device shall prepare a DRQ data block for transfer to the host.

**Transition DPIOI0:1:** When the device has a DRQ data block ready to transfer, the device shall transition to the DPIOI1: Send_PIO_setup state.

**Transition DPIOI0:2:** When the device has encountered an error that causes the command to abort before completing the transfer of the requested data, the device shall transition to the DPIOI3: Error_status state.

**DPIOI1: Send_PIO_setup:** This state is entered when the device is ready to transmit a DRQ data block from the host.

When in this state, the device shall request that the Transport layer transmit a PIO Setup FIS. The initial status shall have BSY cleared to zero and DRQ set to one and with register content as described in the command description in ATA/ATAP-5 (T13/d1321R2) clause 8. The I bit shall be set. If this is the last DRQ data block requested by the command, the ending status shall have BSY cleared to zero and DRQ cleared to zero. If this is not the last data block requested by the command, the ending status shall have BSY set to one and DRQ cleared to zero.

**Transition DPIOI1:1:** When the PIO Setup FIS has been transferred, the device shall transition to the DPIOI2: Transmit_data state.

**DPIOI2: Transmit_data:** This state is entered when the device has transmitted a PIO Setup FIS to the host.

When in this state, the device shall request that the Transport layer transmit a Data FIS containing the DRQ data block.

**Transition DPIOI2:1:** When the Data FIS has been transferred and all data requested by this command have been transferred, the device shall transition to the DI0: Device_idle.

**Transition DPIOI2:2:** When the Data FIS has been transferred but all data requested by this command has not been transferred, or the 8KB transfer limit has been reached, then the device shall transition to the DPIOI0: PIO_in state.

**DPIOI3: Error_status:** This state is entered when the device has encountered an error that causes the command to abort before completing the transfer of the requested data.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register content as described in the command description in ATA/ATAP-5 (T13/d1321R2) clause 8 and the I bit set to one.

**Transition DPIOI3:1:** When the FIS has been transmitted, the device shall transition to the DI0: Device_idle state.

## 9.8 PIO data-out command protocol

This class includes:

- CFA WRITE MULTIPLE WITHOUT ERASE
- CFA WRITE SECTORS WITHOUT ERASE
- DOWNLOAD MICROCODE
- SECURITY DISABLE PASSWORD
- SECURITY ERASE UNIT
- SECURITY SET PASSWORD
- SECURITY UNLOCK
- SMART WRITE LOG SECTOR
- WRITE BUFFER
- WRITE MULTIPLE
- WRITE MULTIPLE EXT
- WRITE SECTOR(S)
- WRITE SECTOR(S) EXT

Execution of this class of command includes the PIO transfer of one or more blocks of data from the host to the device.

| DPIOO0: PIO_out | Prepare to receive DRQ data block transfer from the host. | |
|---|---|---|
| 1. Ready to receive DRQ data block transfer. | → | DPIOO1: Send_PIO_setup |
| 2. All DRQ data blocks received or command aborted due to error. | → | DPIOO3: Send_status |

| DPIOO1: Send_PIO_setup | Request transmission of a PIO Setup FIS to host. | |
|---|---|---|
| 1. PIO Setup FIS transmitted. | → | DPIOO2: Receive_data |

| DPIOO2: Receive_data | Receive Data FIS from the Transport layer. | |
|---|---|---|
| 1. Data FIS received. | → | DPIOO0: PIO_out |

| DPIOO3: Send_status | Request transmission of a Register - Device to Host FIS. | |
|---|---|---|
| 1. FIS transmission complete. | → | DI0: Device_idle |

**DPIOI0: PIO_out State:** This state is entered when the device receives a PIO data-out command or the receipt of one or more DRQ data blocks is required to complete this command.

When in this state, device shall prepare to receive a DRQ data block transfer from the host.

**Transition DPIOO0:1:** When the device is ready to receive a DRQ data block, the device shall transition to the DPIOO1: Send_PIO_setup state.

**Transition DPIOO0:2:** When the device has received all DRQ data blocks requested by this command or the device has encountered an error that causes the command to abort before completing the transfer of the requested data, then the device shall transition to the DPIOO3: Send_status state.

**DPIOO1: Send_PIO_setup:** This state is entered when the device is ready to receive a DRQ data block from the host.

When in this state, the device shall request that the Transport layer transmit a PIO Setup FIS. The initial status shall have BSY cleared to zero and DRQ set to one. If this is the first DRQ data block for this command, the I bit shall be cleared to zero. If this is not the first DRQ data block for this command, the I bit shall be set to one. The ending status shall have BSY set to one and DRQ cleared to zero. The byte count for the DRQ data block shall be indicated.

**Transition DPIOO1:1:** When the PIO Setup FIS has been transferred, the device shall transition to the DPIOO2: Receive_data state.

**DPIOO2:Receive_data:** This state is entered when the device has transmitted a PIO Setup FIS to the host.

When in this state, the device shall receive the requested Data FIS from the Transport layer.

**Transition DPIOO2:1:** When the Data FIS has been received, the device shall transition to the DPIOO0: PIO_out state.

**DPIOO3: Send_status:** This state is entered when the device has received all DRQ data blocks requested by this command or the device has encountered an error that causes the command to abort before completing the transfer of the requested data.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register content as described in the command description in ATA/ATAP-5 (T13/d1321R2) clause 8 and the I bit set to one.

**Transition DPIOO3:1:** When the FIS has been transmitted, the device shall transition to the DI0: Device_idle state.

## 9.9 DMA data in command protocol

This class includes:

- READ DMA
- READ DMA EXT

Execution of this class of command includes the transfer of one or more blocks of data from the device to the host using DMA transfer.

| DDMAI0: DMA_in | Prepare data for the transfer of a Data FIS. | |
|---|---|---|
| 1. Data for Data FIS ready to transfer. | → | DDMAI1: Send_data |
| 2. Command completed or aborted due to error. | → | DDMAI2: Send_status |

| DDMAI1: Send_data | Request transmission of a Data FIS to host. | |
|---|---|---|
| 1. Data FIS transmitted. No more data transfer required for this command, or 8KB transmitted. | → | DDMAI0: DMA_in |

| DDMAI2: Send_status | Request transmission of a Register FIS to host. | |
|---|---|---|
| 1. Register FIS transmitted. | → | DI0: Device_idle |

**DDMAI0: DMA_in State:** This state is entered when the device receives a DMA data-in command or the transmission of one or more data FIS is required to complete the command.

When in this state, device shall prepare the data for transfer of a data FIS to the host.

**Transition DDMAI0:1:** When the device has the data ready to transfer a data FIS, the device shall transition to the DDMAI1: Send_data state.

**Transition DDMAI0:2:** When the device has transferred all of the data requested by this command or has encountered an error that causes the command to abort before completing the transfer of the requested data, then the device shall transition to the DDMAI2: Send_status state.

**DDMAI1: Send_data:** This state is entered when the device has the data ready to transfer a data FIS to the host.

When in this state, the device shall request that the Transport layer transmit a data FIS containing the data. The device command layer shall request a data FIS size of no more than 8KB.

**Transition DDMAI1:1:** When the data FIS has been transferred, the device shall transition to the DMAOI0: DMA_in state.

**DDMAI2: Send_status:** This state is entered when the device has transferred all of the data requested by the command or has encountered an error that causes the command to abort before completing the transfer of the requested data.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register content as described in the command description in ATA/ATAP-5 (T13/d1321R2) clause 8 and the I bit set to one.

**Transition DDMAI2:1:** When the FIS has been transmitted, the device shall transition to the DI0: Device_idle state.

## 9.10 DMA data out command protocol

This class includes:

- WRITE DMA
- WRITE DMA EXT

Execution of this class of command includes the transfer of one or more blocks of data from the host to the device using DMA transfer. A single interrupt is issued at the completion of the successful transfer of all data required by the command.

| DDMAO0: DMA_out | Prepare to receive a Data FIS from the host. | |
|---|---|---|
| 1. Ready to receive Data FIS. | → | DDMAO1: Send_DMA_activate |
| 2. All data requested for this command received or command aborted due to error. | → | DDMAO3: Send_status |

| DDMAO1: Send_DMA_activate | Request transmission of a DMA Activate FIS to host. | |
|---|---|---|
| 1. DMA Activate FIS transmitted. | → | DDMAO2: Receive_data |

| DDMAO2: Receive_data | Receive Data FIS from the Transport layer. | |
|---|---|---|
| 1. Data FIS received. | → | DDMAO0: DMA_out |

| DDMAO3: Send_status | Request transmission of a Register FIS to host. | |
|---|---|---|
| 1. Register FIS transmitted. | → | DI0: Device_idle |

**DDMAO0: DMA_out State:** This state is entered when the device receives a DMA data-out command or the receipt of one or more Data FIS is required to complete this command.

When in this state, device shall prepare to receive a Data FIS from the host.

**Transition DDMAO0:1:** When the device is ready to receive a Data FIS, the device shall transition to the DDMAO1: Send_DMA_activate state.

**Transition DDMAO0:2:** When the device has received all the data requested by this command or the device has encountered an error that causes the command to abort before completing the transfer of the requested data, then the device shall transition to the DDMAO3: Send_status state.

**DDMAO1: Send_DMA_activate:** This state is entered when the device is ready to receive a Data FIS from the host.

When in this state, the device shall request that the Transport layer transmit a DMA Activate FIS.

**Transition DDMAO1:1:** When the DMA Activate FIS has been transferred, the device shall transition to the DDMAO2: Receive_data state.

**DDMAO2: Receive_data:** This state is entered when the device transmitted a DMA Activate FIS to the host.

When in this state, the device shall receive the requested Data FIS from the Transport layer.

**Transition DDMAO2:1:** When the Data FIS has been received, the device shall transition to the DDMAO0: DMA_out state.

**DDMAO3: Send_status:** This state is entered when the device has received all the data requested by this command or the device has encountered an error that causes the command to abort before completing the transfer of the requested data.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register content as described in the command description in ATA/ATAP-5 (T13/d1321R2) clause 8 and the I bit set to one.

**Transition DDMAO3:1:** When the FIS has been transmitted, the device shall transition to the DI0: Device_idle state.

## 9.11 PACKET protocol

This class includes:

- PACKET

States marked with an * are only utilized when queuing is implemented

| DP0: PACKET | Request transmission of a PIO Setup FIS. | | |
|---|---|---|---|
| | 1. FIS transmission complete. | → | DP1: Receive_command |

| DP1: Receive_command | Receive Data FIS containing command packet. | | |
|---|---|---|---|
| | 1. FIS receiption complete. | → | DP2: Check_command |

| DP2: Check_command | Determine the protocol required for the received command. | | |
|---|---|---|---|
| | 1. Non-data command. | → | DP3: PACKET_non-data |
| | 2. PIO data-in command. | → | DP4: PACKET_PIO_in |
| | 3. PIO data-out command. | → | DP6: PACKET_PIO_out |
| | 4. DMA data-in command | → | DP9: PACKET_DMA_in |
| | 5. DMA data-out command | → | DP11: PACKET_DMA_out |

| DP3: PACKET_non-data | Execute Non-data command. | | |
|---|---|---|---|
| | 1. Command execution complete. | → | DP14: Send_status |

| DP4: PACKET_PIO_in | Prepare a DRQ data block for transfer to the host. | | |
|---|---|---|---|
| | 1. DRQ data block ready to transfer. | → | DP4a: PIO_in_setup |
| | 2. Transfer complete or command aborted due to error. | → | DP14: Send_status |
| | 3. * DRQ block is not ready for immediate transfer | → | DP15: Release |

| DP4a: PIO_in_setup | Request transmission of a PIO Setup FIS to host. | | |
|---|---|---|---|
| | 1. PIO Setup FIS transmitted. | → | DP5: Send_PIO_data |

| DP5: Send_PIO_data | Request transmission of a Data FIS to host. | | |
|---|---|---|---|
| | 1. Data FIS transmitted. | → | DP4: PACKET_PIO_in |

| DP6: PACKET_PIO_out | Prepare to receive DRQ data block from the host. | | |
|---|---|---|---|
| | 1. Ready to receive DRQ data block transfer. | → | DP7: PIO_out_setup |
| | 2. All DRQ data blocks received or command aborted due to error. | → | DP14: Send_status |
| | 3. * Not ready to accept DRQ block immediately. | → | DP15: Release |

| DP7: PIO_out_setup | Request transmission of a PIO Setup FIS to host. | |
|---|---|---|
| 1. PIO Setup FIS transmitted. | → | DP8: Receive_PIO_data |

| DP8: Receive_PIO_data | Receive Data FIS from the Transport layer. | |
|---|---|---|
| 1. Data FIS received. | → | DP6: PACKET_PIO_out |

| DP9: PACKET_DMA_in | Prepare data for the transfer of a Data FIS. | |
|---|---|---|
| 1. Data for Data FIS ready to transfer. | → | DP10: Send_DMA_data |
| 2. Command completed or aborted due to error. | → | DP14: Send_status |
| 3. * Data is not ready for immediate transfer. | → | DP15: Release |

| DP10: Send_DMA_data | Request transmission of a Data FIS to host. | |
|---|---|---|
| 1. Data FIS transmitted. No more data transfer required for this command, or 8KB transmitted. | → | DP9: PACKET_DMA_IN |

| DP11: PACKET_DMA_out | Prepare to receive a Data FIS from the host. | |
|---|---|---|
| 1. Ready to receive Data FIS. | → | DP12: Send_DMA_activate |
| 2. All data requested for this command received or command aborted due to error. | → | DP14: Send_status |
| 3. * Not ready for immediate transfer. | → | DP15: Release |

| DP12: Send_DMA_activate | Request transmission of a DMA Activate FIS to host. | |
|---|---|---|
| 1. DMA Activate FIS transmitted. | → | DP13: Receive_DMA_data |

| DP13: Receive_DMA_data | Receive Data FIS from the Transport layer. | |
|---|---|---|
| 1. Data FIS received. | → | DP11: PACKET_DMA_out |

| DP14: Send_status | Request transmission of a Register FIS. | |
|---|---|---|
| 1. FIS transmission complete. | → | DI0: Device_idle |

| * DP15: Release | Request transmission of a Register FIS. | |
|---|---|---|
| 1. FIS transmission complete. | → | DI0: Device_idle |

**DP0: PACKET:** This state is entered when the device receives a PACKET command.

When in this state, the device shall request that the Transport layer transmit a PIO Setup FIS to acquire the command packet associated with this command. The initial status shall have BSY cleared to zero and DRQ set to one. The I bit shall be set to zero. The ending status shall have BSY set to one and DRQ cleared to zero. The byte count for the DRQ data block shall be indicated.

**Transition DP0:1:** When the PIO Setup FIS has been transferred, the device shall transition to the DP1: Receive_command state.

**DP1: Receive_command:** This state is entered when the device transmitted a PIO Setup FIS to the host to get the command packet.

When in this state, the device shall receive the requested Data FIS from the Transport layer.

**Transition DP1:1:** When the Data FIS has been received, the device shall transition to the DP2: Check_command state.

**DP2: Check_command:** This state is entered when the Data FIS containing the command packet has been received.

When in this state, the device shall determine the protocol for the command contained in the command packet.

**Transition DP2:1:** When the command is a non-data transfer command, the device shall transition to the DP3: PACKET_non-data state.

**Transition DP2:2:** When the command is a PIO data-in transfer command, the device shall transition to the DP4: PACKET_PIO_in state.

**Transition DP2:3:** When the command is a PIO data-out transfer command, the device shall transition to the DP6: PACKET_PIO_out state.

**Transition DP2:4:** When the command is a DMA data-in transfer command, the device shall transition to the DP9: PACKET_DMA_in state.

**Transition DP2:5:** When the command is a DMA data-out transfer command, the device shall transition to the DP11: PACKET_DMA_out state.

**DP3: PACKET_non-data State:** This state is entered when a received command is a non-data command.

When in this state, the device shall execute the requested command.

**Transition DP3:1:** When command execution completes, the device shall transition to the DP14: Send_status state.

**DP4: PACKET_PIO_in State:** This state is entered when the device receives a PIO data-in command or the transmission of one or more DRQ data blocks is required to complete the command.

When in this state, device shall prepare a DRQ data block for transfer to the host.

**Transition DP4:1:** When the device has a DRQ data block ready to transfer, the device shall transition to the DP4a: PIO_in_setup.

**Transition DP4:2:** When all of the data requested by this command has been transferred or the device has encountered an error that causes the command to abort before completing the transfer of the requested data, then the device shall transition to the DP14: Send_status state.

*** Transition DP4:3:** When the device supports overlap and queuing and does not have a DRQ data block ready to transfer immediately, the device shall transition to the DP15: Release state.

**DP4a: PIO_in_setup:** This state is entered when the device is ready to transfer a DRQ block to the host.

When in this state, the device shall request that the Transport layer transmit a PIO Setup FIS. The initial status shall have BSY cleared to zero and DRQ set to one. The I bit shall be set to one. The ending status shall have BSY set to one and DRQ cleared to zero. The byte count for the DRQ data block shall be indicated.

**Transition DP4a:1:** When the PIO Setup FIS has been transferred, the device shall transition to the DP5:SendPIO_data state.

**DP5:Send_PIO_data:** This state is entered when the device is ready to transfer a DRQ data block to the host.

When in this state, the device shall request that the Transport layer transmit a Data FIS containing the DRQ data block.

**Transition DP5:1:** When the Data FIS has been transferred, the device shall transition to the DP4: PACKET_PIO_in state.

**DP6: PACKET_PIO_out State:** This state is entered when the device receives a PIO data-out command or the receipt of one or more DRQ data blocks is required to complete the command.

When in this state, device shall prepare to receive a DRQ data block transfer from the host.

**Transition DP6:1:** When the device is ready to receive a DRQ data block transfer, the device shall transition to the DP7: PIO_out_setup state.

**Transition DP6:2:** When the device has received all DRQ data blocks requested by this command or the device has encountered an error that causes the command to abort before completing the transfer of the requested data, then the device shall transition to the DP14: Send_status state.

*** Transition DP6:3:** When the device supports overlap and queuing and can not accept a DRQ data block immediately, the device shall transition to the DP15: Release state.

**DP7: PIO_out_setup:** This state is entered when the device is ready to receive a DRQ data block from the host.

When in this state, the device shall request that the Transport layer transmit a PIO Setup FIS. The initial status shall have BSY cleared to zero and DRQ set to one. The I bit shall be set to one. The ending status shall have BSY set to one and DRQ cleared to zero. The byte count for the DRQ data block shall be indicated.

**Transition DP7:1:** When the PIO Setup FIS has been transferred, the device shall transition to the DP8: Receive_PIO_data state.

**DP8: Receive_PIO_data:** This state is entered when the device transmitted a PIO Setup FIS to the host.

When in this state, the device shall receive the requested Data FIS from the Transport layer.

**Transition DP8:1:** When the Data FIS has been received, the device shall transition to the DP6: PACKET_PIO_out state.

**DP9: PACKET_DMA_in State:** This state is entered when the device receives a DMA data-in command or the transmission of one or more Data FIS is required to complete the command.

When in this state, device shall prepare the data for transfer of a Data FIS to the host.

**Transition DP9:1:** When the device has the data ready to transfer a Data FIS, the device shall transition to the DP10: Send_DMA_data state.

**Transition DP9:2:** When the device has transferred all of the data requested by this command or has encountered an error that causes the command to abort before completing the transfer of the requested data, then the device shall transition to the DP14: Send_status state.

*** Transition DP9:3:** When the device supports overlap and queuing and does not have data ready to transfer immediately, the device shall transition to the DP15: Release state.

**DP10: Send_DMA_data:** This state is entered when the device has the data ready to transfer a Data FIS to the host.

When in this state, the device shall request that the Transport layer transmit a Data FIS containing the data.

**Transition DP10:1:** When the Data FIS has been transferred, the device shall transition to the DP9: PACKET_DMA_in state. The device command layer shall request a data FIS size of no more than 8KB.

**DP11: PACKET_DMA_out State:** This state is entered when the device receives a DMA data-out command or the receipt of one or more Data FIS is required to complete the command.

When in this state, device shall prepare to receive a Data FIS from the host.

**Transition DP11:1:** When the device is ready to receive a Data FIS, the device shall transition to the DP12: Send_DMA_activate state.

**Transition DP11:2:** When the device has received all the data requested by this command or the device has encountered an error that causes the command to abort before completing the transfer of the requested data, then the device shall transition to the DP14: Send_status state.

*** Transition DP11:3:** When the device supports overlap and queuing and can not accept a Data FIS immediately, the device shall transition to the DP15: Release state.

**DP12: Send_DMA_activate:** This state is entered when the device is ready to receive a Data FIS from the host.

When in this state, the device shall request that the Transport layer transmit a DMA Activate FIS.

**Transition DP12:1:** When the DMA Activate FIS has been transferred, the device shall transition to the DP13: Receive_DMA_data state.

**DP13: Receive_DMA_data:** This state is entered when the device transmitted a DMA Activate FIS to the host.

When in this state, the device shall receive the requested Data FIS from the Transport layer.

**Transition DP13:1:** When the Data FIS has been received, the device shall transition to the DP11: PACKET_DMA_out state.

**DP14: Send_status:** This state is entered when the device has received all the data requested by this command or the device has encountered an error that causes the command to abort before completing the transfer of the requested data.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register content as described in the command description in ATA/ATAP-5 (T13/d1321R2) clause 8 and the I bit set to one.

**Transition DP14:1:** When the FIS has been transmitted, then the device shall transition to the DI0: Device_idle state.

*** DP15: Release:** This state is entered when the device is not able to do a data transfer immediately.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register content as described in the command description in ATA/ATAPI-5 (T13/d1321R2) clause 8, with the REL bit set to one, and, if the bus release interrupt has been enabled by a previous Set Features Command, with the I bit set to one.

**Transition DP15:1:** When the FIS has been transmitted, then the device shall transition to the DI0: Device_idle state.

## 9.12 READ DMA QUEUED command protocol

This class includes:

- READ DMA QUEUED
- READ DMA QUEUED EXT

Execution of this class of command includes the transfer of one or more blocks of data from the device to the host using DMA transfer. All data for the command may be transferred without a bus release between the command receipt and the data transfer. This command may bus release before transferring data. The host shall initialize the DMA controller prior to transferring data. When data transfer is begun, all data for the request shall be transferred without a bus release.

| DDMAQI0: DMA_queued_in | Determine whether to transfer or release. | | |
|---|---|---|---|
| | 1. Data for Data FIS ready to transfer. | → | DDMAQI1: Send_data |
| | 2. Command aborted due to error. | → | DDMAQI3: Send_status |
| | 3. Bus release | → | DDMAQI4: Release |

| DDMAQI1: Send_data | Request transmission of a Data FIS to host. | | |
|---|---|---|---|
| | 1. Data FIS transmitted. No more data transfer required for this command, or 8KB transmitted. | → | DDMAQI2: Prepare_data |

| DDMAQI2: Prepare_data | Prepare data for the next Data FIS. | | |
|---|---|---|---|
| | 1. Data ready. | → | DDMAQI1: Send _data |
| | 2. Command complete or aborted due to error. | → | DDMAQI3: Send_status |

| DDMAQI3: Send_status | Request transmission of a Register FIS to host. | | |
|---|---|---|---|
| | 1. Register FIS transmitted. | → | DI0: Device_idle |

| DDMAQI4: Release | Request transmission of a Register FIS to host. | |
|---|---|---|
| 1. Register FIS transmitted. | → | DI0: Device_idle |

**DDMAQI0: DMA_queued_in State:** This state is entered when the device receives a READ DMA QUEUED command.

When in this state, device shall determine if the requested data is ready to transfer to the host.

**Transition DDMAQI0:1:** When the device has the requested data ready to transfer a Data FIS immediately, the device shall transition to the DDMAQI1: Send_data state.

**Transition DDMAQI0:2:** When the device has encountered an error that causes the command to abort before completing the transfer of the requested data, the device shall transition to the DDMAQI3: Send_status state.

**Transition DDMAQI0:3:** When the device does not have the requested data ready to transfer a Data FIS immediately, the device shall transition to the DDMAQI4: Release state.

**DDMAQI1: Send_data:** This state is entered when the device has the data ready to transfer a Data FIS to the host.

When in this state, the device shall request that the Transport layer transmit a Data FIS containing the data.
**Transition DDMAQI1:1:** When the Data FIS has been transferred, the device shall transition to the DDMAQI2: Prepare_data state. The device command layer shall request a data FIS size of no more than 8KB.

**DDMAQI2: Prepare_data:** This state is entered when the device has completed the transfer a Data FIS to the host.

When in this state, the device shall prepare the data for the next Data FIS.

**Transition DDMAQI2:1:** When data is ready for the Data FIS, the device shall transition to the DDMAQI1: Send_data state.

**Transition DDMAQI2:2:** When all data requested for the command has been transmitted or an error has been encountered that causes the command to abort before completing the transfer of the requested data, the device shall transition to the DDMAQI3: Send_status state.

**DDMAQI3: Send_status:** This state is entered when the device has transferred all of the data requested by the command or has encountered an error that causes the command to abort before completing the transfer of the requested data.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register content as described in the command description in ATA/ATAP-5 (T13/d1321R2) clause 8 and the I bit set to one.

**Transition DDMAQI3:1:** When the FIS has been transmitted, the device shall transition to the DI0: Device_idle state.

**DDMAQI4: Release:** This state is entered when the device does not have the requested data available for immediate transfer.

When in this state, the device shall request that the Transport layer transmit a Register FIS with the REL bit set to one, with register content as described in the command description in ATA/ATAPI-5 (T13/d1321R2) clause 8, and, if the bus release interrupt has been enabled by a previous Set Features Command, with the I bit set to one.

**Transition DDMAQI4:1:** When the FIS has been transmitted, then the device shall transition to the DI0: Device_idle state.

## 9.13 WRITE DMA QUEUED command protocol

This class includes:

- WRITE DMA QUEUED
- WRITE DMA QUEUED EXT

Execution of this class of command includes the transfer of one or more blocks of data from the device to the host using DMA transfer. All data for the command may be transferred without a bus release between the command receipt and the data transfer. This command may bus release before transferring data. The host shall initialize the DMA controller prior to transferring data. When data transfer is begun, all data for the request shall be transferred without a bus release.

| DDMAQO0: DMA-queued_out | Determine whether to transfer or release. | |
|---|---|---|
| 1. Ready to accept Data FIS | → | DDMAQO1: Send_DMA_activate |
| 2. Command due to error. | → | DDMAQO4: Send_status |
| 3. Bus release | → | DDMAQO5: Release |

| DDMAQO1: Send_DMA_activate | Request transmission of a DMA Activate FIS to host. | |
|---|---|---|
| 1. DMA Activate FIS transmitted. | → | DDMAQO2: Receive_data |

| DDMAQO2: Receive_data | Receive Data FIS from the Transport layer. | |
|---|---|---|
| 1. Data FIS received. | → | DDMAQO3: Prepare_data_buffer |

| DDMAQO3: Prepare_data_buffer | Prepare to receive the next Data FIS. | |
|---|---|---|
| 1. Ready to receive. | → | DDMAQO1: Send DMA_activate |
| 2. Command complete or aborted due to error. | → | DDMAQO4: Send_status |

| DDMAQO4: Send_status | Request transmission of a Register FIS to host. | |
|---|---|---|
| 1. Register FIS transmitted. | → | DI0: Device_idle |

| DDMAQO5: Release | Request transmission of a Register FIS to host. | |
|---|---|---|
| 1. Register FIS transmitted. | → | DI0: Device_idle |

**DDMAQO0: DMA_queued_out State:** This state is entered when the device receives a WRITE DMA QUEUED command.

When in this state, device shall determine if it is ready to accept the requested data from the host.

**Transition DDMAQO0: 1:** When the device is ready to receive a Data FIS immediately, the device shall transition to the DDMAQO1: Send_DMA_activate state.

**Transition DDMAQO0:2:** When the device has encountered an error that causes the command to abort before completing the transfer of the requested data, the device shall transition to the DDMAQO4: Send_status state.

**Transition DDMAQO0:3:** When the device does not have the requested data ready to transfer a Data FIS immediately, the device shall transition to the DDMAQO5: Release state.

**DDMAQO1:Send_DMA_activate:** This state is entered when the device is ready to receive a Data FIS from the host.

When in this state, the device shall request that the Transport layer transmit a DMA Activate FIS.

**Transition DDMAQO1:1:** When the DMA Activate FIS has been transferred, the device shall transition to the DDMAQO2: Receive_data state.

**DDMAQO2:Receive_data:** This state is entered when the device transmitted a DMA Activate FIS to the host.

When in this state, the device shall receive the requested Data FIS from the Transport layer.

**Transition DDMAQO2:1:** When the Data FIS has been received, the device shall transition to the DDMAQO3: Prepare_data_buffer state.

**DDMAQO3: Prepare_data_buffer:** This state is entered when the device has completed receiving a Data FIS from the host.

When in this state, the device shall prepare for receipt of the next Data FIS.

**Transition DDMAQO3:1:** When ready to receive the Data FIS, the device shall transition to the DDMAQO1: Send_DMA_activate state.

**Transition DDMAQO3:2:** When all data requested for the command has been transmitted or an error has been encountered that causes the command to abort before completing the transfer of the requested data, the device shall transition to the DDMAQO4: Send_status state.

**DDMAQO4: Send_status:** This state is entered when the device has transferred all of the data requested by the command or has encountered an error that causes the command to abort before completing the transfer of the requested data.

When in this state, the device shall request that the Transport layer transmit a Register FIS with register content as described in the command description in ATA/ATAP-5 (T13/d1321R2) clause 8 and the I bit set to one.

**Transition DDMAQO4:1:** When the FIS has been transmitted, then the device shall transition to the DI0: Device_idle state.

**DDMAQO5: Release:** This state is entered when the device cannot receive the requested data immediately.

When in this state, the device shall request that the Transport layer transmit a Register FIS with REL set to one, with register content as described in the command description in ATA/ATAPI-5 (T13/d1321R2) clause 8, and, if the bus release interrupt has been enabled by a previous Set Features Command, with the I bit set to one.

**Transition DDMAQO5:1:** When the FIS has been transmitted, then the device shall transition to the DI0: Device_idle state.

# 10 Host adapter register interface

Serial ATA host adapters include an additional block of registers mapped separately and independently from the ATA Command Block Registers for reporting additional status and error information and to allow control of capabilities unique to Serial ATA. These additional registers, referred to as the Serial ATA Status and Control Registers (SCR's) are organized as 16 contiguous 32-bit registers. The base address and mapping scheme for these registers is defined by the specific host adapter implementation – for example, PCI controller implementations may map the SCR's using the PCI mapping capabilities. Table 27. – SCR definition illustrates the overall organization of the Serial ATA register interface including both the ATA Command Block Registers and the Status and Control registers. Legacy software does not make use of the Serial ATA Status and Control registers. The Serial ATA Status and Control register are associated with the serial interface and are independent of any master/slave emulation the host adapter may implement.

**Table 27. – SCR definition**

ATA Command Block and Control Block registers

| | A2 | A1 | A0 | Read | Write |
|---|---|---|---|---|---|
| CS 0 | 0 | 0 | 0 | Data | Data |
| | 0 | 0 | 1 | Error | Features |
| | 0 | 1 | 0 | Sector Count [15:8] [7:0] | Sector Count [15:8] [7:0] |
| | 0 | 1 | 1 | Sector Number [31:24] [7:0] | Sector Number [31:24] [7:0] |
| | 1 | 0 | 0 | Cylinder Low [39:32] [15:8] | Cylinder Low [39:32] [15:8] |
| | 1 | 0 | 1 | Cylinder High [47:40] [23:16] | Cylinder High [47:40] [23:16] |
| | 1 | 1 | 0 | Device/Head | Device/Head |
| | 1 | 1 | 1 | Status | Command |
| CS 1 | 1 | 1 | 0 | Alternate Status | Device Control |

Serial ATA Status and Control registers

| | | |
|---|---|---|
| SATA register | 0 | SATA Status/Control |
| SATA register | 1 | SATA Status/Control |
| … | … | SATA Status/Control |
| SATA register | 14 | SATA Status/Control |
| SATA register | 15 | SATA Status/Control |

## 10.1 SStatus, SError and SControl registers

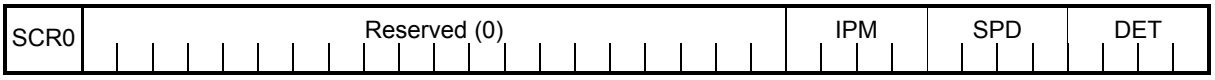
Serial ATA provides an additional block of registers to control the interface and to retrieve interface state information. There are 16 contiguous registers allocated of which the first three are defined and the remaining 13 are reserved for future definition. Table 28. – SCR Definition defines the Serial ATA Status and Control registers.

**Table 28. – SCR Definition**

| | |
|---|---|
| SCR[0] | SStatus register |
| SCR[1] | SError register |
| SCR[2] | SControl register |
| SCR[3] | Reserved |
| … | … |
| SCR[15] | Reserved |

### 10.1.1 SStatus register

The Serial ATA interface Status register - SStatus - is a 32-bit read-only register that conveys the current state of the interface and host adapter. The register conveys the interface state at the time it is read and is updated continuously and asynchronously by the host adapter. Writes to the register have no effect.




DET The DET value indicates the interface device detection and Phy state.
- 0000 No device detected and Phy commnunication not established
- 0001 Device presence detected but Phy communication not established
- 0011 Device presence detected and Phy communication established
- 0100 Phy in offline mode as a result of the interface being disabled or running in a BIST loopback mode

All other values reserved

SPD The SPD value indicates the negotiated interface communication speed established
- 0000 No negotiated speed (device not present or communication not established)
- 0001 Generation 1 communication rate negotiated

All other values reserved

IPM The IPM value indicates the current interface power management state
- 0000 Device not present or communication not established
- 0001 Interface in active state
- 0010 Interface in PARTIAL power management state
- 0110 Interface in SLUMBER power management state

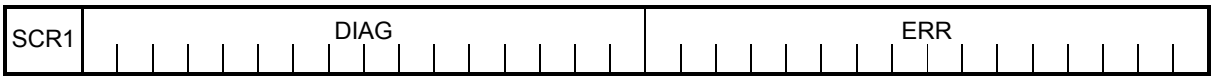
All other values reserved

Reserved All reserved fields shall be cleared to zero.

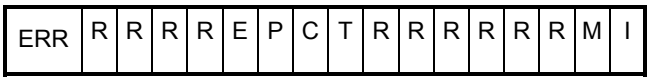
NOTE – The interface must be in the active state for the interface device detection value (DET field) to be accurate. When the interface is in the partial or slumber state no communication between the host and target is established resulting in a DET value corresponding to no device present or no communication established. As a result the insertion or removal of a device may not be accurately detected under all conditions such as when the interface is quiescent as a result of being in sleep or slumber state. This field alone may therefore be insufficient to satisfy all the requirements for device attach or detatch detection during all possible interface states.

### 10.1.2 SError register

The Serial ATA interface Error register - SError - is a 32-bit register that conveys supplemental Interface error information to complement the error information available in the Shadow Register Block Error register. The register represents all the detected errors accumulated since the last time the SError register was cleared (whether recovered by the interface of not). Set bits in the error register are explicitly cleared by a write operation to the SError register, or a reset operation. The value written to clear set error bits shall have 1's encoded in the bit positions corresponding to the bits that are to be cleared. Host software should clear the Interface SError register at appropriate checkpoints in order to best isolate error conditions and the commands they impact.




ERR The ERR field contains error information for use by host software in determining the appropriate response to the error condition. The field is bit significant as defined in the following figure.




C Non-recovered persistent communication or data integrity error: A communication error that was not recovered occurred that is expected to be persistent. Since the error condition is expected to be persistent the operation need not be retried by host software. 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.

E Internal error: The host bus adapter experienced an internal error that caused the operation to fail and may have put the host bus adapter into an error state. Host software should reset the interface before re-trying the operation. If the condition persists, the host bus adapter may suffer from a design issue rendering it incompatible with the attached device.

I Recovered data integrity error: A data integrity error occurred that was recovered by the interface through a retry operation or other recovery action. This can arise from a noise burst in the transmission, a voltage supply variation, or from other causes. No action is required by host software since the operation ultimately succeeded, however, host software may elect to track such recovered errors in order to gauge overall communications integrity and potentially step down the negotiated communication speed.

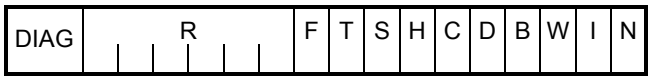
M Recovered communications error: Communications between the device and host was temporarily lost but was re-established. This can arise from a device temporarily being removed, from a temporary loss of Phy synchronization, or from other causes and may be derived from the PhyNRdy signal between the Phy and Link layers. No action is required by the host software since the operation ultimately succeeded, however, host software may elect to track such recovered errors in order to gauge overall communications integrity and potentially step down the negotiated communication speed.

P Protocol error: A violation of the Serial ATA protocol was detected. This can arise from invalid or poorly formed FIS's being received, from invalid state transitions, or from other causes. Host software should reset the interface and retry the corresponding operation. If such an error persists, the attached device may have a design issue rendering it incompatible with the host bus adapter.

R Reserved bit for future use: Shall be cleared to zero.

T Non-recovered transient data integrity error: A data integrity error occurred that was not recovered by the interface. Since the error condition is not expected to be persistent the operation should be retried by host software.

DIAG The DIAG field contains diagnostic error information for use by diagnostic software in validating correct operation or isolating failure modes. The field is bit significant as defined in the following figure.




B 10b to 8b Decode error: When set to a one, this bit indicates that one or more 10b to 8b decoding errors occurred since the bit was last cleared.

C CRC Error: When set to one, this bit indicates that one or more CRC errors occurred with the Link Layer since the bit was last cleared.

D Disparity Error: When set to one, this bit indicates that incorrect disparity was detected one or more times since the last time the bit was cleared.

F Unrecognized FIS type: When set to one, this bit indicates that since the bit was last cleared one or more FIS's were received by the Transport layer with good CRC, but had a type field that was not recognized.

I Phy Internal Error: When set to one, this bit indicates that the Phy detected some internal error since the last time this bit was cleared.

N PhyRdy change: When set to one, this bit indicates that the PhyRdy signal changed state since the last time this bit was cleared.

H Handshake error: When set to one, this bit 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 10b/8b decoding error, or other error condition leading to a negative handshake on a transmitted frame.

R Reserved bit for future use: Shall be cleared to zero.

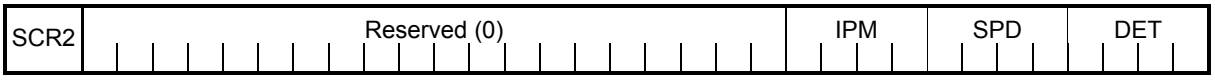
S Link Sequence Error: When set to one, this bit indicates that one or more Link state machine error conditions was encountered since the last time this bit was cleared. The Link Layer state machine defines the conditions under which the link layer detects an erroneous transition.

T Transport state transition error: When set to one, this bit 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.

W Comm Wake: When set to one this bit indicates that a Comm Wake signal was detected by the Phy since the last time this bit was cleared.

### 10.1.3 SControl register

The Serial ATA interface Control register - SControl - is a 32-bit read-write register that provides the interface by which software controls Serial ATA interface capabilities. Writes to the SControl register result in an action being taken by the host adapter or interface. Reads from the register return the last value written to it.




DET The DET field controls the host adapter device detection and interface initialization.
- 0000b No device detection or initialization action requested
- 0001b 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. Upon a write to the SControl register that sets the LSB of the DET field to one, the host interface shall transition to the HP1: HR_Reset state and shall remain in that state until the LSB of the DET field is cleared to zero by a subsequent write to the SControl register.
- 0100b Disable the Serial ATA interface and put Phy in offline mode.
- All other values reserved

SPD The SPD field represents the highest allowed communication speed the interface is allowed to negotiate when interface communication speed is established
- 0000b No speed negotiation restrictions
- 0001b Limit speed negotiation to a rate not greater than Generation 1 communication rate
- All other values reserved

IPM The IPM field represents the enabled interface power management states that can be invoked via the Serial ATA interface power management capabilities
- 0000b No interface power management state restrictions
- 0001b Transitions to the PARTIAL power management state disabled
- 0010b Transitions to the SLUMBER power management state disabled
- 0011b Transitions to both the PARTIAL and SLUMBER power management states disabled
- All other values reserved

Reserved All reserved fields shall be cleared to zero.

# 11 Error handling

## 11.1 Architecture

The layered architecture of Serial ATA extends to error handling as well. As indicated in Figure 67 – Error handling architecture, each layer in the Serial ATA stack has as inputs error indication from the next lower layer (except for the Phy layer which has no lower layer associated with it), data from the next lower layer in the stack, and data from the next higher layer in the stack. Each layer has its local error detection capability to identify errors specific to that layer based on the received data from the lower and higher layers. Each layer performs local recovery and control actions and may forward error information to the next higher layer in the stack.

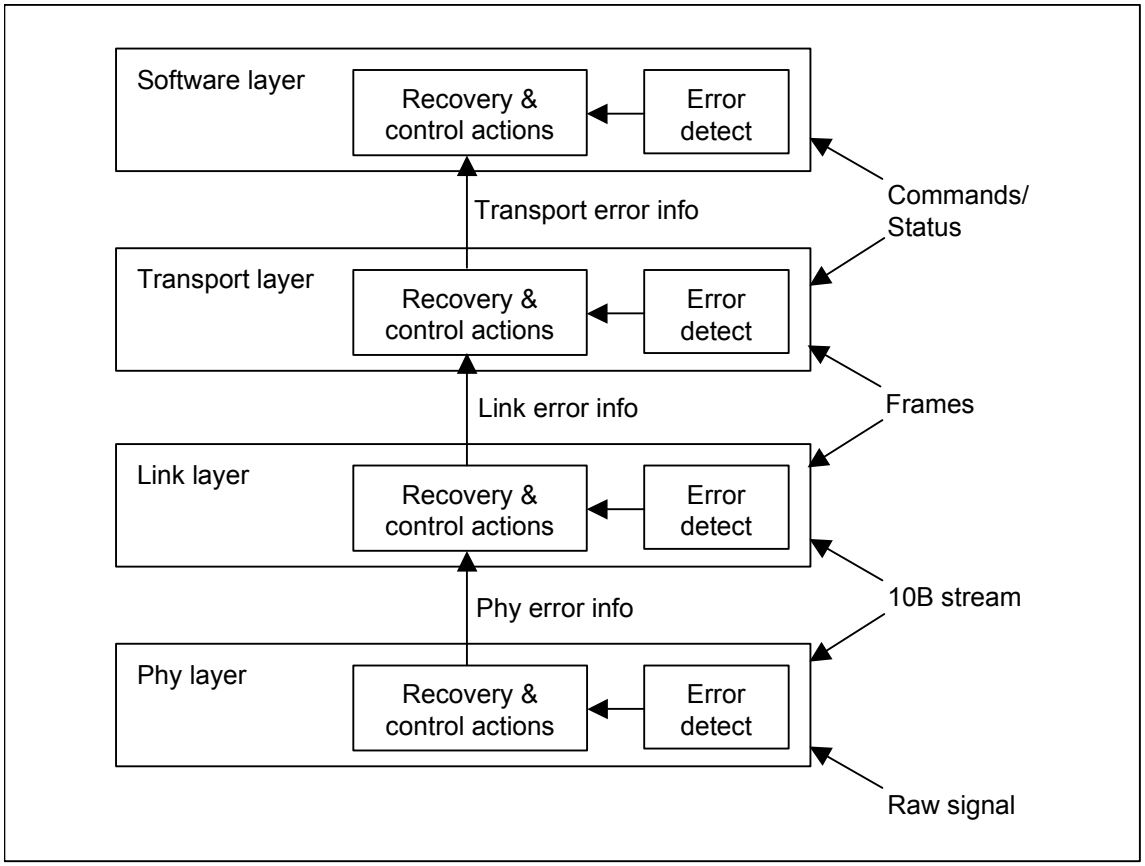



**Figure 67 – Error handling architecture**

Error responses are generally classified into four categories

Freeze
Abort
Retry
Track/ignore

The error handling responses described in this section are not comprehensive and are included to cover specific known error scenarios as well as to illustrate typical error control and recovery actions. This section is therefore descriptive and supplemental to the error reporting interface defined in section 10 and implementations may vary in their internal error recovery and control actions.

For the most severe error conditions in which state has been critically perturbed in a way that it is not recoverable, the appropriate error response is to freeze and rely on a reset or similar operation to restore all necessary state to return to normal operation.

For error conditions that are expected to be persistent, the appropriate error response is to abort and fail the attempted operation. Such failures usually imply notification up the stack in order to inform host software of the condition.

For error conditions that are transient and not expected to persist, the appropriate response is to retry the failed operation. Only failed operations that have not perturbed system state can be retried. Such retries may either be handled directly by the recovery and error control actions in the relevant layer or may be handled by host software in response to error information being conveyed to it.

Non-critical recoverable conditions can either be tracked or ignored. Such conditions include those that were recovered through a retry or other recovery operation at a lower layer in the stack. Tracking such errors can often be beneficial in identifying a marginally operating component or other imminent failure.

## 11.2 Phy error handling overview

### 11.2.1 Error detection

There are three primary categories of error that the Phy layer detects internally:

- No device present
- OOB signaling sequence failure
- Phy internal error (loss of synchronization of communications link)

A no device present condition results from a physical disconnection in the media between the host controller and device, whether intermittent or persistent. The host controller shall detect device presence as part of the interface reset sequence defined in section 6.7 - Functional specifications.

An OOB signaling failure condition arises when the sequence of OOB signaling events cannot be completed, prior to declaration of the "Phy Ready" state. OOB signaling sequences are required when emerging from a power management "partial" or "slumber" state, from a "loopback/BIST" test state, or from initial power-up. OOB signaling sequences are used to achieve a specifically ordered exchange of COMRESET, COMINIT, COMWAKE, and ALIGN patterns to bring the communications link up between host controller and device. The specific sequences are detailed in section 6 - Physical layer.

A Phy Internal Error can arise from a number of conditions, whether it is caused by the characteristics of the input signal, or an internal error unique to the implementation, it will always result in the loss of the synchronization of the communications link.

Fixed local receive PLL frequency architectures (oversampling, tracking/non-tracking) are sensitive to input data rate frequency variations from the nominal expected rate, thus they usually have elasticity buffers. Elasticity buffers are used to accommodate the difference between input data rate frequency, and the local receive PLL frequency -- overrun/underrun conditions will occur if the Tx difference in frequency is too high/low with respect to the Rx local PLL frequency, commonly declared as a Phy Internal Error.

VCO-based (PLL) clock recovery architectures are also sensitive to input data rate frequency variations, high frequency jitter, and may have trouble achieving lock -- which may be declared as a Phy Internal Error.

A number of state machine, impedance compensation, and serializer/deserializer (SerDes) circuits make up a typical Serial ATA interface Phy, and the various types of error conditions can be grouped together to make up the declared "Phy Internal Error". These errors are usually specific to each implementation.

### 11.2.2 Error control actions

#### 11.2.2.1 No device present

Due to the nature of the physical interface, it is possible for the Phy to determine that a device is attached to the cable at various times. As a direct result, the Phy is responsible for detecting presence of an attached device and this presence shall be reported in the Interface Status register such that host software can respond appropriately.

During the interface initialization sequence, an internal state bit “Device Detect” shall be cleared in the host controller when the COMRESET signal is issued. The “Device Detect” state bit shall be set in the host controller when the host controller detects a COMINIT signal from the attached device. The “Device Detect” state bit corresponds to the device presence detect information in the Interface Status register defined in section 10.1.

Note that device presence and communications established are separately reported in the Interface Status register in order to encompass situations in which an attached device is detected by the Phy, but the Phy is unable to establish communications with it.

#### 11.2.2.2 OOB signaling sequence failure

The Phy does not have any timeout conditions for the interface reset signaling sequence as defined in section 6.7. If a device is present, the Phy shall detect device presence within 10ms of a power-on reset (i.e. COMINIT must be returned within 10ms of an issued COMRESET). If a device is not present, the Phy is not required to time-out and may remain in the reset state indefinitely until host software intervenes. Upon successful completion of the interface initialization sequence, the Phy shall be ready, active, and synchronized, and the Interface Status register bits shall reflect this as defined in section 10.1.

#### 11.2.2.3 Phy internal error

As described in section 11.2.1, there are several potential sources of errors categorized as “Phy Internal Errors.” In order to accommodate a range of implementations without making the software error handling approach implementation dependent, all the different potential sources of internal Phy errors are combined for the purpose of reporting the condition in the Interface Error register as defined in section 10.1.2.. This requirement does not preclude each vendor from implementing their own level of error diagnostic bits, but those shall reside in vendor specific register locations.

Phy internal errors shall result in the Phy becoming not ready (the PhyRdy signal being deasserted) and the corresponding Interface Status register and Interface Error registers bits shall be updated as defined in section 10.1.2.

The "Phy Ready Change" bit, as defined in the Interface Error Register, shall be updated as per its definition in section 10.1.2

### 11.2.3 Error reporting

Phy errors are generally reported to the Link layer in addition to being reflected in the Interface Status register and Interface Error registers as defined in section 10.1.2.

## 11.3 Link error handling overview

### 11.3.1 Error detection

There are two primary categories of errors that the Link layer detects internally:

Invalid state transitions
Data integrity errors

Invalid state transition errors can arise from a number of sources and the Link layer responses to many such error conditions are defined in section 7 - Link layer. Data integrity errors generally arise from noise in the physical interconnect.

### 11.3.2 Error control actions

Errors detected by the Link during a transmission are generally handled by accumulating the errors until the end of the transmission and reflecting the reception error condition in the final R_ERR/R_OK handshake. Specific scenarios are listed in the following sections.

#### 11.3.2.1 Invalid state transitions

Invalid state transitions are generally handled through the return to a known state, for example where the Link state diagrams in section 7.6 - Link layer state diagrams define the responses for invalid state transition attempts. Returning to a known state is generally achieved through two recovery paths depending on the state of the system.

If the invalid state transitions are attempted during the transmission of a frame (after the receipt of an SOF), the Link shall signal negative acknowledgement (R_ERR) to the transmitting agent.
If the invalid state transition is not during a frame transmission, the Link shall go directly to the idle state, and await the next operation.

The following paragraphs outline the requirements during specific state transition scenarios, and their respective Link error control actions:

Following reception of one or more consecutive X_RDY at the receiver interface, if the next control character received is not SOF, the Link shall notify the Transport Layer of the condition and transition to the idle state.

Following transmission of X_RDY, if there is no returned R_RDY received, no Link recovery action shall be attempted. The higher-level layers will eventually time out, and reset the interface.

On receipt of an unexpected SOF, when the receiving interface had not yet signaled readiness to receive data with the R_RDY signal, that receiving interface shall remain in the idle state issuing

SYNC primitives until the transmitting interface terminates the transmission and also returns to the idle state.

If the transmitter closes a frame with EOF and WTRM, and receives neither a R_OK nor an R_ERR within a predetermined timeout, no Link recovery action shall be attempted. The higher-level layers will eventually time out, and reset the interface.

If the transmitter signals EOF, and a control character other than SYNC, R_OK, or R_ERR is received, the Link layer shall persistently continue to await reception of a proper terminating primitive.

Data integrity errors:
Data integrity errors are generally handled by signaling the Transport Layer in order to potentially trigger a transmission retry operation, or to convey failed status information to the host software. In order to return to a known state, data integrity errors are usually signaled via the frame acknowledgement handshake, at the end of a frame transmission, before returning to the idle state.

The following paragraphs outline the requirements during specific data integrity error scenarios, and the respective Link error control actions:

On detection of a CRC error at the end of receiving a frame (at EOF), the Link Layer shall notify the Transport Layer that the received frame contains a CRC error. Furthermore, the Link Layer shall issue the negative acknowledgement, R_ERR, as the frame status handshake, and shall return to the idle state.

On detection of a disparity error or other 8b/10b coding violation during the receipt of a frame, the Link Layer will retain this error information, and at the close of the received frame the Link Layer shall provide the negative acknowledgement, R_ERR, as the frame handshake, and shall notify the Transport Layer of the error.

The control actions are essentially the same for coding violations as for CRC errors.

### 11.3.3 Error reporting

Link error conditions are reported to the Transport layer via a private interface between the Link and Transport Layers. Additionally, Link errors are reported in the Interface Error register as defined section 10.1.2.

## 11.4 Transport error handling overview

The Transport is the highest level layer in the Serial ATA interface. The Transport layer communicates errors to the software and/or performs local error recovery, and initiates control actions, such as retrying a class of FIS transmissions

The Transport layer informs the Link layer of detected errors so that the Link layer can reflect Transport errors in the R_ERR/R_OK handshake at the end of each frame. Devices shall reflect any R_ERR frame handshakes in the command ending status reflected in the transmitted register FIS that conveys the operation ending status. The Transport layer also reflects any encountered error information in the Interface Error register.

The Transport may retry any FIS transmission, provided the system state has not changed as a result of the corresponding failure, and may retry any number of times. For scenarios where repeated retry operations persistently fail, host software will eventually time out the corresponding command and perform recovery operations.

### 11.4.1 Error detection

In addition to the error information passed to it by the Link layer, the Transport layer internally detects the following categories of errors:

Internal errors
Frame errors
Protocol errors & state errors

There are several kinds of internal errors to the Transport layer, including overflow/underflow of the various speed matching FIFOs. Internal errors are generally handled by failing the corresponding transaction, and returning to a state equivalent to a failed transaction (e. g. the state that would result from a bad CRC).

The Transport layer detects several kinds of frame errors including reception of frames with incorrect CRC, reception of frames with invalid TYPE field, and reception of ill-formed frames (such as a register frames that are not the correct length). Frame errors are generally handled by failing the corresponding transaction and returning to a state equivalent to a failed transaction (such as the state that would result from a bad CRC).

Protocol and state transition errors generally stem from ill-behaved devices not following the proper Serial ATA protocol, and include errors such as, the PIO count value not matching the number of data character subsequently transferred, and errors in the sequence of events.

Protocol and state transition errors are generally handled by failing the corresponding transaction and returning to a state equivalent to a failed transaction (such as the state that would result from a frame being received with a bad CRC).

### 11.4.2 Error control actions

#### 11.4.2.1 Internal errors

Internal errors are generally handled by failing the corresponding transaction and either re-trying the transaction or notifying host software of the failure condition in order to ultimately generate a host software retry response. The following are specific internal error scenarios and their corresponding Transport error control actions.

If the receive FIFO overflows, the Transport layer shall signal frame reception negative acknowledgement, by signaling the Link layer to return R_ERR during the frame acknowledgement handshake. Subsequent actions are equivalent to a frame reception with erroneous CRC.

If the transmit FIFO underruns, the Transport layer shall close the transmitting frame with an EOF and CRC value that is forced to be incorrect in order to ensure the receiver of the corrupted frame also executes appropriate error control actions. This scenario results only from a transmitter design problem and should never occur for properly implemented devices.

#### 11.4.2.2 Frame errors

Frame errors generally are handled in one of two ways depending on whether the error is expected to be transient or persistent and whether system state has been perturbed. For error conditions expected to be transient (such as a CRC error), and for which the system state has not been perturbed, the Transport layer may retry the corresponding transaction any number of times until ultimately a host timeout and software reset, or other error recovery attempt is made. For error conditions that are not a result of a transient error condition (such as an invalid TYPE field in a received FIS), the error response is generally to fail the transaction and report the failure.

The following are specific frame error scenarios and their corresponding Transport error control actions.

If the Transport receives an FIS with an invalid CRC signaled from the Link layer, the Transport layer shall signal the Link layer to negatively acknowledge frame reception by asserting R_ERR during the frame acknowledgement handshake.

The transmitter of a negatively acknowledged frame may retry the FIS transmission provided the system state has not been perturbed. Frame types that can be retransmitted are:

Register – Host to Device
Register – Device to Host
DMA Activate – Device to Host
DMA Setup – Device to Host
PIO Setup – Device to Host
Set Device Bits – Device to Host

Because data transmission FIS's result in a change in the host bus adapter's internal state, either through the DMA controller changing its state or through a change in the remaining PIO repetition count, data transmission FIS's should never be retried.

The Transport layer is not required to retry those failed FIS transmissions that do not change system state, but the Transport layer may attempt retry any number of times. For conditions that are not addressed through retries, such as persistent errors, host software will eventually time out the transaction and reset the interface.

If the Transport Layer detects reception of an FIS with unrecognized TYPE value, the Transport Layer shall signal the Link Layer to negatively acknowledge the frame reception by asserting R_ERR during the frame acknowledgement handshake.

If the Transport Layer detects reception of a malformed frame, such as a frame with incorrect length, the Transport Layer shall signal the Link Layer to negatively acknowledge the frame reception by asserting R_ERR during the frame acknowledgement handshake.

#### 11.4.2.3 Protocol and state transition errors

Protocol and state errors stem from ill-behaved devices not following the proper protocol. Such errors are generally handled by failing the corresponding transactions and returning to a known state. Since such errors are not caused by an environmental transient, no attempt to retry such failed operations should be made. The following are specific frame error scenarios and their corresponding Transport error control actions.

If the PIO transfer count expires, and two symbols later is not the EOF control character (the CRC falls between the last data character and the EOF), the transfer count stipulated in the PIO Setup FIS did not match the size of the subsequent data payload. For this data-payload/transfer-count mismatch, the Transport Layer shall signal the Link Layer to negatively acknowledge frame reception by asserting R_ERR during the frame acknowledgement handshake.

### 11.4.3 Error reporting

The Transport Layer reports errors to host software via the Serial ATA Status and Control registers. Devices communicate Transport error information to host software via transmitting a register FIS to update the ATA Shadow Register Block Status and Error register values.

All Transport error conditions that are not handled/recovered by the Transport layer shall set the error bit in the Shadow Register Block Status register, and update the value in the Error register through transmission of an appropriate Register FIS.

Host Transport error conditions shall result in the status and error values in the SStatus and SError register being updated with values corresponding to the error condition and shall result in the Link layer being notified to negatively acknowledge the offending FIS during the final reception handshake.

## 11.5 Software error handling overview

The software layer error handling is in part defined by legacy software behavior. Superset error reporting capabilities are supported by the Transport layer through the SCR's, and software can take advantage of those error reporting capabilities to improve error handling for Serial ATA.

### 11.5.1 Error detection

There are three overall error detection mechanisms by which software identifies and responds to Serial ATA errors

- Bad status in the Command Block Status register
- Bad status in the Serial ATA SError register
- Command failed to complete (timeout)

Conditions that return bad status in the Command Block Status register, but for which no Serial ATA interface error information is available, correspond to the error conditions specified in the ATA standard. Such error conditions and responses are defined in the ATA standard and there is no unique handling of those in Serial ATA. Errors in this category include command errors, such as attempts to read from an LBA past the end of the disk, as well as device-specific failures such as data not readable from the given sector number. These failures are not related to the Serial ATA interface, and thus no Serial ATA specific interface status information is available for these error conditions. Only the status information returned by the device is available for identifying the source of the problem, plus any available SMART data that might apply.

Transport layer error conditions, whether recovered or not, are reflected in the Interface Status and Interface Error registers as defined in section 10.1.2. The host Transport layer is responsible for reflecting error information in the Interface Status and Interface Error registers, while the device Transport layer is responsible for reflecting unrecovered errors in the Shadow Register Block Status and Error registers through transmission of appropriate Register FIS's.

Commands that fail to complete are detected by host driver software through a timeout mechanism. Generally such timeouts result in no status or error information for the command being conveyed to host software and software may not be able to determine the source or cause of such errors.

### 11.5.2 Error control actions

Conditions that return bad status in the Shadow Register Block Status register but for which no interface error information in the SError register is available shall be handled as defined in the ATA standard.

Conditions that return interface error information in the SError register are handled through four basic responses:

Freeze
Abort/Fail
Retry (possible after reset)
Track/ignore

Error conditions that result in catastrophic system perturbation that is not recoverable should result in the system halting. Serial ATA does not define any explicit halting conditions, however, software may be able to infer such conditions.

Error conditions that are not expected to be transient and which are not expected to succeed with subsequent attempts should result in the affected command being aborted and failed. Failure of such commands should be reported to higher software layers for handling. Scenarios in which this response is appropriate include attempts to communicate with a device that is not attached, and failure of the interface to successfully negotiate communications with an attached device.

Error conditions that are expected to be transient should result in the affected command being retried. Such commands may either be retried directly or may be retried after an interface and/or device reset, depending on the particular error value reported in the SError register. Scenarios in which this response is appropriate include noise events resulting in CRC errors, 8b/10b code violations, or disparity errors.

Conditions that are recoverable and for which no explicit error handling is required may be tracked or ignored. Tracking such errors allows subsequent fault isolation for marginal components and accommodates possible recovery operations. Scenarios in which this response is appropriate include tracking the number of Phy synchronization losses in order to identify a potential cable fault or to accommodate an explicit reduction in the negotiated communications rate.

# APPENDICES

# APPENDIX A. SAMPLE CODE FOR CRC AND SCRAMBLING

## A.1 CRC calculation

### A.1.1 Overview

The following section provides an informative implementation of the CRC polynomial. The example is intended as an aid in verifying a HDL implementation of the algorithm.

### A.1.2 Maximum frame size

The 32-bit CRC used by Serial ATA can be shown to provide detection of two 10-bit errors up to a maximum frame size of 16384 bytes. This will provide for future expansion of Serial ATA FIS's to a maximum of 64 bytes of fixed overhead while still permitting a maximum user data payload of 8192 bytes.

### A.1.3 Example code for CRC algorithm

The following code, written in C, illustrates an implementation of the CRC algorithm. A Register Host to Device FIS containing a PIO write command is used as example input. The CRC calculated is the check Dword appended to a transmitted serial stream immediately preceding the EOF primitive.. The code displays both the resulting command FIS and the intermediate CRC polynomial values. The GNU tool chain will compile the code using the following command:

"gcc –o crc.exe crc.c"

This code is supplied for illustrative purposes only.

### A.1.4 Example code for CRC algorithm

```
/*****************************************************************************/
/*                                                                           */
/* crc.c                                                                     */
/*                                                                           */
/* This sample code reads standard in for a sequence of 32 bit values       */
/* formatted in hexadecimal with a leading "0x" (e.g. 0xDEADBEEF).  The     */
/* code calculates the Serial ATA CRC for the input data stream.  The       */
/* generator polynomial used is:                                            */
/*         32   26   23   22   16   12   11   10   8   7   5   4   2        */
/* G(x) = x  + x  + x  + x  + x  + x  + x  + x  + x + x + x + x + x + x + 1 */
/*                                                                           */
/* This sample code uses a parallel implementation of the CRC calculation   */
/* circuit that is suitable for implementation in hardware.  A block        */
/* diagram of the circuit being emulated is shown below.                    */
/*                                                                           */
/*                      +---+          +---+          +---+                 */
/*   Data_In ---------->|   |          |   |          | R |                 */
/*                      | + |--------->| * |--------->| e |----+            */
/*               +----->|   |          |   |          | g |    |            */
/*               |      +---+          +---+          +---+    |            */
/*               |                                             |            */
/*               |                                             |            */
/*               +---------------------------------------------+            */
/*                                                                           */
/* The CRC value is initialized to 0x52325032 as defined in the Serial ATA  */
/* specification.                                                            */
/*                                                                           */
/*****************************************************************************/

#include <stdlib.h>
#include <stdio.h>
```

```
main(argc,argv)
int argc;
char *argv[];
{
   int               i,
                     data_count;
   unsigned int      crc,
                     data_in;
   unsigned char     crc_bit[32],
                     new_bit[32];


   crc = 0x52325032;
   data_count = 0;

   while (scanf(" 0x%8x", &data_in) == 1)  {
      data_count++;
      /* Add the data_in value to the current value of the CRC held in the  */
      /* "register".  The addition is performed modulo two (XOR).           */
      crc ^= data_in;
      /* Expand the value of the CRC held in the register to 32 individual  */
      /* bits for easy manipulation.                                        */
      for (i = 0; i < 32; ++i)  {
         crc_bit[i] = (crc >> i) & 0x01;
      }

      /* The following 32 assignments perform the function of the box       */
      /* labeled "*" in the block diagram above.  The new_bit array is a    */
      /* temporary holding place for the new CRC value being calculated.    */
      /* Note that there are lots of shared terms in the assignments below. */
      new_bit[31] = crc_bit[31] ^ crc_bit[30] ^ crc_bit[29] ^ crc_bit[28] ^ crc_bit[27] ^ crc_bit[25] ^ crc_bit[24] ^
                    crc_bit[23] ^ crc_bit[15] ^ crc_bit[11] ^ crc_bit[9]  ^ crc_bit[8]  ^ crc_bit[5];
      new_bit[30] = crc_bit[30] ^ crc_bit[29] ^ crc_bit[28] ^ crc_bit[27] ^ crc_bit[26] ^ crc_bit[24] ^ crc_bit[23] ^
                    crc_bit[22] ^ crc_bit[14] ^ crc_bit[10] ^ crc_bit[8]  ^ crc_bit[7]  ^ crc_bit[4];
      new_bit[29] = crc_bit[31] ^ crc_bit[29] ^ crc_bit[28] ^ crc_bit[27] ^ crc_bit[26] ^ crc_bit[25] ^ crc_bit[23] ^
                    crc_bit[22] ^ crc_bit[21] ^ crc_bit[13] ^ crc_bit[9]  ^ crc_bit[7]  ^ crc_bit[6]  ^ crc_bit[3];
      new_bit[28] = crc_bit[30] ^ crc_bit[28] ^ crc_bit[27] ^ crc_bit[26] ^ crc_bit[25] ^ crc_bit[24] ^ crc_bit[22] ^
                    crc_bit[21] ^ crc_bit[20] ^ crc_bit[12] ^ crc_bit[8]  ^ crc_bit[6]  ^ crc_bit[5]  ^ crc_bit[2];
      new_bit[27] = crc_bit[29] ^ crc_bit[27] ^ crc_bit[26] ^ crc_bit[25] ^ crc_bit[24] ^ crc_bit[23] ^ crc_bit[21] ^
                    crc_bit[20] ^ crc_bit[19] ^ crc_bit[11] ^ crc_bit[7]  ^ crc_bit[5]  ^ crc_bit[4]  ^ crc_bit[1];
      new_bit[26] = crc_bit[31] ^ crc_bit[28] ^ crc_bit[26] ^ crc_bit[25] ^ crc_bit[24] ^ crc_bit[23] ^ crc_bit[22] ^
                    crc_bit[20] ^ crc_bit[19] ^ crc_bit[18] ^ crc_bit[10] ^ crc_bit[6]  ^ crc_bit[4]  ^ crc_bit[3]  ^
                    crc_bit[0];
      new_bit[25] = crc_bit[31] ^ crc_bit[29] ^ crc_bit[28] ^ crc_bit[22] ^ crc_bit[21] ^ crc_bit[19] ^ crc_bit[18] ^
                    crc_bit[17] ^ crc_bit[15] ^ crc_bit[11] ^ crc_bit[8]  ^ crc_bit[3]  ^ crc_bit[2];
      new_bit[24] = crc_bit[30] ^ crc_bit[28] ^ crc_bit[27] ^ crc_bit[21] ^ crc_bit[20] ^ crc_bit[18] ^ crc_bit[17] ^
                    crc_bit[16] ^ crc_bit[14] ^ crc_bit[10] ^ crc_bit[7]  ^ crc_bit[2]  ^ crc_bit[1];
      new_bit[23] = crc_bit[31] ^ crc_bit[29] ^ crc_bit[27] ^ crc_bit[26] ^ crc_bit[20] ^ crc_bit[19] ^ crc_bit[17] ^
                    crc_bit[16] ^ crc_bit[15] ^ crc_bit[13] ^ crc_bit[9]  ^ crc_bit[6]  ^ crc_bit[1]  ^ crc_bit[0];
      new_bit[22] = crc_bit[31] ^ crc_bit[29] ^ crc_bit[27] ^ crc_bit[26] ^ crc_bit[24] ^ crc_bit[23] ^ crc_bit[19] ^
                    crc_bit[18] ^ crc_bit[16] ^ crc_bit[14] ^ crc_bit[12] ^ crc_bit[11] ^ crc_bit[9]  ^ crc_bit[0];
      new_bit[21] = crc_bit[31] ^ crc_bit[29] ^ crc_bit[27] ^ crc_bit[26] ^ crc_bit[24] ^ crc_bit[22] ^ crc_bit[18] ^
                    crc_bit[17] ^ crc_bit[13] ^ crc_bit[10] ^ crc_bit[9]  ^ crc_bit[5];
      new_bit[20] = crc_bit[30] ^ crc_bit[28] ^ crc_bit[26] ^ crc_bit[25] ^ crc_bit[23] ^ crc_bit[21] ^ crc_bit[17] ^
                    crc_bit[16] ^ crc_bit[12] ^ crc_bit[9]  ^ crc_bit[8]  ^ crc_bit[4];
      new_bit[19] = crc_bit[29] ^ crc_bit[27] ^ crc_bit[25] ^ crc_bit[24] ^ crc_bit[22] ^ crc_bit[20] ^ crc_bit[16] ^
                    crc_bit[15] ^ crc_bit[11] ^ crc_bit[8]  ^ crc_bit[7]  ^ crc_bit[3];
      new_bit[18] = crc_bit[31] ^ crc_bit[28] ^ crc_bit[26] ^ crc_bit[24] ^ crc_bit[23] ^ crc_bit[21] ^ crc_bit[19] ^
                    crc_bit[15] ^ crc_bit[14] ^ crc_bit[10] ^ crc_bit[7]  ^ crc_bit[6]  ^ crc_bit[2];
      new_bit[17] = crc_bit[31] ^ crc_bit[30] ^ crc_bit[27] ^ crc_bit[25] ^ crc_bit[23] ^ crc_bit[22] ^ crc_bit[20] ^
                    crc_bit[18] ^ crc_bit[14] ^ crc_bit[13] ^ crc_bit[9]  ^ crc_bit[6]  ^ crc_bit[5]  ^ crc_bit[1];
      new_bit[16] = crc_bit[30] ^ crc_bit[29] ^ crc_bit[26] ^ crc_bit[24] ^ crc_bit[22] ^ crc_bit[21] ^ crc_bit[19] ^
                    crc_bit[17] ^ crc_bit[13] ^ crc_bit[12] ^ crc_bit[8]  ^ crc_bit[5]  ^ crc_bit[4]  ^ crc_bit[0];
      new_bit[15] = crc_bit[30] ^ crc_bit[27] ^ crc_bit[24] ^ crc_bit[21] ^ crc_bit[20] ^ crc_bit[18] ^ crc_bit[16] ^
                    crc_bit[15] ^ crc_bit[12] ^ crc_bit[9]  ^ crc_bit[8]  ^ crc_bit[7]  ^ crc_bit[5]  ^ crc_bit[4]  ^
                    crc_bit[3];
      new_bit[14] = crc_bit[29] ^ crc_bit[26] ^ crc_bit[23] ^ crc_bit[20] ^ crc_bit[19] ^ crc_bit[17] ^ crc_bit[15] ^
                    crc_bit[14] ^ crc_bit[11] ^ crc_bit[8]  ^ crc_bit[7]  ^ crc_bit[6]  ^ crc_bit[4]  ^ crc_bit[3]  ^
                    crc_bit[2];
      new_bit[13] = crc_bit[31] ^ crc_bit[28] ^ crc_bit[25] ^ crc_bit[22] ^ crc_bit[19] ^ crc_bit[18] ^ crc_bit[16] ^
                    crc_bit[14] ^ crc_bit[13] ^ crc_bit[10] ^ crc_bit[7]  ^ crc_bit[6]  ^ crc_bit[5]  ^ crc_bit[3]  ^
                    crc_bit[2]  ^ crc_bit[1];
      new_bit[12] = crc_bit[31] ^ crc_bit[30] ^ crc_bit[27] ^ crc_bit[24] ^ crc_bit[21] ^ crc_bit[18] ^ crc_bit[17] ^
                    crc_bit[15] ^ crc_bit[13] ^ crc_bit[12] ^ crc_bit[9]  ^ crc_bit[6]  ^ crc_bit[5]  ^ crc_bit[4]  ^
                    crc_bit[2]  ^ crc_bit[1]  ^ crc_bit[0];
      new_bit[11] = crc_bit[31] ^ crc_bit[28] ^ crc_bit[27] ^ crc_bit[26] ^ crc_bit[25] ^ crc_bit[24] ^ crc_bit[20] ^
                    crc_bit[17] ^ crc_bit[16] ^ crc_bit[15] ^ crc_bit[14] ^ crc_bit[12] ^ crc_bit[9]  ^ crc_bit[4]  ^
                    crc_bit[3]  ^ crc_bit[1]  ^ crc_bit[0];
      new_bit[10] = crc_bit[31] ^ crc_bit[29] ^ crc_bit[28] ^ crc_bit[26] ^ crc_bit[19] ^ crc_bit[16] ^ crc_bit[14] ^
                    crc_bit[13] ^ crc_bit[9]  ^ crc_bit[5]  ^ crc_bit[3]  ^ crc_bit[2]  ^ crc_bit[0];
      new_bit[9]  = crc_bit[29] ^ crc_bit[24] ^ crc_bit[23] ^ crc_bit[18] ^ crc_bit[13] ^ crc_bit[12] ^ crc_bit[11] ^
```

```
                  crc_bit[9]  ^ crc_bit[5]  ^ crc_bit[4]  ^ crc_bit[2]  ^ crc_bit[1];
      new_bit[8]  = crc_bit[31] ^ crc_bit[28] ^ crc_bit[23] ^ crc_bit[22] ^ crc_bit[17] ^ crc_bit[12] ^ crc_bit[11] ^
                    crc_bit[10] ^ crc_bit[8]  ^ crc_bit[4]  ^ crc_bit[3]  ^ crc_bit[1]  ^ crc_bit[0];
      new_bit[7]  = crc_bit[29] ^ crc_bit[28] ^ crc_bit[25] ^ crc_bit[24] ^ crc_bit[23] ^ crc_bit[22] ^ crc_bit[21] ^
                    crc_bit[16] ^ crc_bit[15] ^ crc_bit[10] ^ crc_bit[8]  ^ crc_bit[7]  ^ crc_bit[5]  ^ crc_bit[3]  ^
                    crc_bit[2]  ^ crc_bit[0];
      new_bit[6]  = crc_bit[30] ^ crc_bit[29] ^ crc_bit[25] ^ crc_bit[22] ^ crc_bit[21] ^ crc_bit[20] ^ crc_bit[14] ^
                    crc_bit[11] ^ crc_bit[8]  ^ crc_bit[7]  ^ crc_bit[6]  ^ crc_bit[5]  ^ crc_bit[4]  ^ crc_bit[2]  ^
                    crc_bit[1];
      new_bit[5]  = crc_bit[29] ^ crc_bit[28] ^ crc_bit[24] ^ crc_bit[21] ^ crc_bit[20] ^ crc_bit[19] ^ crc_bit[13] ^
                    crc_bit[10] ^ crc_bit[7]  ^ crc_bit[6]  ^ crc_bit[5]  ^ crc_bit[4]  ^ crc_bit[3]  ^ crc_bit[1]  ^
                    crc_bit[0];
      new_bit[4]  = crc_bit[31] ^ crc_bit[30] ^ crc_bit[29] ^ crc_bit[25] ^ crc_bit[24] ^ crc_bit[20] ^ crc_bit[19] ^
                    crc_bit[18] ^ crc_bit[15] ^ crc_bit[12] ^ crc_bit[11] ^ crc_bit[8]  ^ crc_bit[6]  ^ crc_bit[4]  ^
                    crc_bit[3]  ^ crc_bit[2]  ^ crc_bit[0];
      new_bit[3]  = crc_bit[31] ^ crc_bit[27] ^ crc_bit[25] ^ crc_bit[19] ^ crc_bit[18] ^ crc_bit[17] ^ crc_bit[15] ^
                    crc_bit[14] ^ crc_bit[10] ^ crc_bit[9]  ^ crc_bit[8]  ^ crc_bit[7]  ^ crc_bit[3]  ^ crc_bit[2]  ^
                    crc_bit[1];
      new_bit[2]  = crc_bit[31] ^ crc_bit[30] ^ crc_bit[26] ^ crc_bit[24] ^ crc_bit[18] ^ crc_bit[17] ^ crc_bit[16] ^
                    crc_bit[14] ^ crc_bit[13] ^ crc_bit[9]  ^ crc_bit[8]  ^ crc_bit[7]  ^ crc_bit[6]  ^ crc_bit[2]  ^
                    crc_bit[1]  ^ crc_bit[0];
      new_bit[1]  = crc_bit[28] ^ crc_bit[27] ^ crc_bit[24] ^ crc_bit[17] ^ crc_bit[16] ^ crc_bit[13] ^ crc_bit[12] ^
                    crc_bit[11] ^ crc_bit[9]  ^ crc_bit[7]  ^ crc_bit[6]  ^ crc_bit[1]  ^ crc_bit[0];
      new_bit[0]  = crc_bit[31] ^ crc_bit[30] ^ crc_bit[29] ^ crc_bit[28] ^ crc_bit[26] ^ crc_bit[25] ^ crc_bit[24] ^
                    crc_bit[16] ^ crc_bit[12] ^ crc_bit[10] ^ crc_bit[9]  ^ crc_bit[6]  ^ crc_bit[0];


      /* The new CRC value has been calculated as individual bits in the    */
      /* new_bit array.  Re-assembled it into a 32 bit value and "clock" it */
      /* into the "register".                                               */
      crc = 0;
      for (i = 31; i >= 0; --i)  {
         crc = crc << 1;
         crc |= new_bit[i];
      }
      printf("Running CRC value is 0x%08X\n", crc);
   }

   printf("\n\nThe total number of data words processed was %d\n", data_count);
   printf("The CRC is 0x%08X\n\n", crc);

   return 0;
}
```

## A.1.5 Example CRC implementation output

The following is the sample data used as input for the example stored in file *sample*:

```
0x00308027
0xE1234567
0x00000000
0x00000002
0x00000000
```

Executing the command *./crc* < *sample* yields the following output:

```
Running CRC value is 0x11E353FD
Running CRC value is 0x0F656DA7
Running CRC value is 0x3D14369C
Running CRC value is 0x92D0D681
Running CRC value is 0x319FFF6F


The total number of data words processed was 5
The CRC is 0x319FFF6F
```

# A.2 Scrambling calculation

## A.2.1 Overview

The following section provides an informative implementation of the scrambling polynomial. The example is intended as an aid in verifying a HDL implementation of the algorithm.

## A.2.2 Example code for scrambling algorithm

The following code, written in C, illustrates an implementation of the Scrambling algorithm. A Register Host to Device FIS containing a PIO write command is used as example input. The code displays both the resulting scrambled data and the raw output of the scrambler. The GNU tool chain will compile the code using the following command:

"gcc –o scramble.exe scramble.c"

This code is supplied for illustrative purposes only.

## A.2.3 Example scrambler implementation

```
/*************************************************************************/
/*                                                                       */
/* scramble.c                                                            */
/*                                                                       */
/* This sample code generates the entire sequence of 65535 Dwords produced  */
/* by the scrambler defined in the Serial ATA specification.  The        */
/* specification calls for an LFSR to generate a string of bits that will   */
/* be packaged into 32 bit Dwords to be XORed with the data Dwords.  The    */
/* generator polynomial specified is:                                    */
/*                          16   15   13   4                             */
/*                 G(x) = x  + x  + x  + x + 1                           */
/*                                                                       */
/* Parallelized versions of the scrambler are initialized to a value     */
/* derived from the initialization value of 0xFFFF defined in the        */
/* specification.  This implementation is initialized to 0xF0F6.  Other  */
/* parallel implementations will have different initial values.  The     */
/* important point is that the first Dword output of any implementation  */
/* must equal 0xC2D2768D.                                                */
/*                                                                       */
/* This code does not represent an elegant solution for a C implementation, */
/* but it does demonstrate a method of generating the sequence that can be  */
/* easily implemented in hardware.  A block diagram of the circuit emulated */
/* by this code is shown below.                                          */
/*                                                                       */
/*         +-----------------------------------+                         */
/*         |                                   |                         */
/*         |                                   |                         */
/*         |     +---+               +---+     |                         */
/*         |     | R |               | * |     |                         */
/*         +---->| e |---------+---->| M |---+---> Output(31 downto 16)  */
/*               | g |         |     | 1 |                               */
/*               +---+         |     +---+                               */
/*                             |                                         */
/*                             |     +---+                               */
/*                             |     | * |                               */
/*                             +---->| M |---------> Output(15 downto 0)   */
/*                                   | 2 |                               */
/*                                   +---+                               */
/*                                                                       */
/* The register shown in the block diagram is a 16 bit register.  The two   */
/* boxes, *M1 and *M2, each represent a multiply by a 16 by 16 binary    */
/* matrix.  A 16 by 16 matrix times a 16 bit vector yields a 16 bit vector. */
/* The two vectors are the two halves of the 32 bit scrambler value.  The   */
/* upper half of the scrambler value is stored back into the context     */
/* register to be used to generate the next value in the scrambler       */
/* sequence.                                                             */
/*                                                                       */
/*************************************************************************/
```

```
#include <stdlib.h>
#include <stdio.h>


main(argc,argv)
int argc;
char *argv[];


{
   int               i, j;
   unsigned short    context;        /* The 16 bit register that holds the context or state */
   unsigned long     scrambler;      /* The 32 bit output of the circuit    */
   unsigned char     now[16];        /* The individual bits of context      */
   unsigned char     next[32];       /* The computed bits of scrambler      */



   /* Parallelized versions of the scrambler are initialized to a value     */
   /* derived from the initialization value of 0xFFFF defined in the        */
   /* specification.  This implementation is initialized to 0xF0F6.  Other  */
   /* parallel implementations will have different initial values.  The     */
   /* important point is that the first Dword output of any implementation  */
   /* must equal 0xC2D2768D.                                                */
   context = 0xF0F6;


   for (i = 0; i < 65535; ++i)  {
      /* Split the register contents (the variable context) up into its     */
      /* individual bits for easy handling.                                 */
      for (j = 0; j < 16; ++j)  {
         now[j] = (context >> j) & 0x01;
      }

      /* The following 16 assignments implement the matrix multiplication   */
      /* performed by the box labeled *M1.                                  */
      /* Notice that there are lots of shared terms in these assignments.   */
      next[31] = now[12] ^ now[10] ^ now[7]  ^ now[3]  ^ now[1]  ^ now[0];
      next[30] = now[15] ^ now[14] ^ now[12] ^ now[11] ^ now[9]  ^ now[6]  ^ now[3]  ^ now[2]  ^ now[0];
      next[29] = now[15] ^ now[13] ^ now[12] ^ now[11] ^ now[10] ^ now[8]  ^ now[5]  ^ now[3]  ^ now[2]  ^ now[1];
      next[28] = now[14] ^ now[12] ^ now[11] ^ now[10] ^ now[9]  ^ now[7]  ^ now[4]  ^ now[2]  ^ now[1]  ^ now[0];
      next[27] = now[15] ^ now[14] ^ now[13] ^ now[12] ^ now[11] ^ now[10] ^ now[9]  ^ now[8]  ^ now[6]  ^ now[1]  ^ now[0];
      next[26] = now[15] ^ now[13] ^ now[11] ^ now[10] ^ now[9]  ^ now[8]  ^ now[7]  ^ now[5]  ^ now[3]  ^ now[0];
      next[25] = now[15] ^ now[10] ^ now[9]  ^ now[8]  ^ now[7]  ^ now[6]  ^ now[4]  ^ now[3]  ^ now[2];
      next[24] = now[14] ^ now[9]  ^ now[8]  ^ now[7]  ^ now[6]  ^ now[5]  ^ now[3]  ^ now[2]  ^ now[1];
      next[23] = now[13] ^ now[8]  ^ now[7]  ^ now[6]  ^ now[5]  ^ now[4]  ^ now[2]  ^ now[1]  ^ now[0];
      next[22] = now[15] ^ now[14] ^ now[7]  ^ now[6]  ^ now[5]  ^ now[4]  ^ now[1]  ^ now[0];
      next[21] = now[15] ^ now[13] ^ now[12] ^ now[6]  ^ now[5]  ^ now[4]  ^ now[0];
      next[20] = now[15] ^ now[11] ^ now[5]  ^ now[4];
      next[19] = now[14] ^ now[10] ^ now[4]  ^ now[3];
      next[18] = now[13] ^ now[9]  ^ now[3]  ^ now[2];
      next[17] = now[12] ^ now[8]  ^ now[2]  ^ now[1];
      next[16] = now[11] ^ now[7]  ^ now[1]  ^ now[0];

      /* The following 16 assignments implement the matrix multiplication   */
      /* performed by the box labeled *M2.                                  */
      next[15] = now[15] ^ now[14] ^ now[12] ^ now[10] ^ now[6]  ^ now[3]  ^ now[0];
      next[14] = now[15] ^ now[13] ^ now[12] ^ now[11] ^ now[9]  ^ now[5]  ^ now[3]  ^ now[2];
      next[13] = now[14] ^ now[12] ^ now[11] ^ now[10] ^ now[8]  ^ now[4]  ^ now[2]  ^ now[1];
      next[12] = now[13] ^ now[11] ^ now[10] ^ now[9]  ^ now[7]  ^ now[3]  ^ now[1]  ^ now[0];
      next[11] = now[15] ^ now[14] ^ now[10] ^ now[9]  ^ now[8]  ^ now[6]  ^ now[3]  ^ now[2]  ^ now[0];
      next[10] = now[15] ^ now[13] ^ now[12] ^ now[9]  ^ now[8]  ^ now[7]  ^ now[5]  ^ now[3]  ^ now[2]  ^ now[1];
      next[9]  = now[14] ^ now[12] ^ now[11] ^ now[8]  ^ now[7]  ^ now[6]  ^ now[4]  ^ now[2]  ^ now[1]  ^ now[0];
      next[8]  = now[15] ^ now[14] ^ now[13] ^ now[12] ^ now[11] ^ now[10] ^ now[7]  ^ now[6]  ^ now[5]  ^ now[1]  ^ now[0];
      next[7]  = now[15] ^ now[13] ^ now[11] ^ now[10] ^ now[9]  ^ now[6]  ^ now[5]  ^ now[4]  ^ now[3]  ^ now[0];
      next[6]  = now[15] ^ now[10] ^ now[9]  ^ now[8]  ^ now[5]  ^ now[4]  ^ now[2];
      next[5]  = now[14] ^ now[9]  ^ now[8]  ^ now[7]  ^ now[4]  ^ now[3]  ^ now[1];
      next[4]  = now[13] ^ now[8]  ^ now[7]  ^ now[6]  ^ now[3]  ^ now[2]  ^ now[0];
      next[3]  = now[15] ^ now[14] ^ now[7]  ^ now[6]  ^ now[5]  ^ now[3]  ^ now[2]  ^ now[1];
      next[2]  = now[14] ^ now[13] ^ now[6]  ^ now[5]  ^ now[4]  ^ now[2]  ^ now[1]  ^ now[0];
      next[1]  = now[15] ^ now[14] ^ now[13] ^ now[5]  ^ now[4]  ^ now[1]  ^ now[0];
      next[0]  = now[15] ^ now[13] ^ now[4]  ^ now[0];

      /* The 32 bits of the output have been generated in the "next" array. */
      /* Reassemble the bits into a 32 bit Dword.                           */
      scrambler = 0;
      for (j = 31; j >= 0; --j)  {
         scrambler = scrambler << 1;
         scrambler |= next[j];
      }
      /* The upper half of the scrambler output is stored backed into the   */
```

```
      /* register as the saved context for the next cycle.                */
      context = scrambler >> 16;
      printf("0x%08X\n", scrambler);

   }

   return 0;

}
```

### A.2.4 Example scrambler implementation output

The following lists the first 32 results generated by the scrambler and the C sample code listed above.

```
0xC2D2768D
0x1F26B368
0xA508436C
0x3452D354
0x8A559502
0xBB1ABE1B
0xFA56B73D
0x53F60B1B
0xF0809C41
0x747FC34A
0xBE865291
0x7A6FA7B6
0x3163E6D6
0xF036FE0C
0x1EF3EA29
0xEB342694
0x53853B17
0xE94ADC4D
0x5D200E88
0x6901EDD0
0xFA9E38DE
0x68DB4B07
0x450A437B
0x960DD708
0x3F35E698
0xFE7698A5
0xC80EF715
0x666090AF
0xFAF0D5CB
0x2B82009F
0x0E317491
0x76F46A1E
```

## A.3 Example frame

The following table shows the steps and values used in the transmission of a simple FIS.

For LBA = 1234567 and Sector Count = 2

**Table 29. – CRC and scrambler calculation example - PIO Write Command**

| FIS Data | Scrambler Value | Scrambled Data | Accumulated CRC Value | Comments |
|---|---|---|---|---|
| 3737B57C | N/A | 3737B57C | N/A | SOF, primitives not scrambled, scrambler reset |
| 00308027 | C2D2768D | C2E2F6AA | 11E353FD | RegH2D, Command = 30, Cbit set |
| E1234567 | 1F26B368 | FE05F60F | 0F656DA7 | LBA 1234567 |
| 00000000 | A508436C | A508436C | 3D14369C | Extended LBA = 0 |
| 00000002 | 3452D354 | 3452D356 | 92D0D681 | Control Reg = 0, 2 sectors |
| 00000000 | 8A559502 | 8A559502 | 319FFF6F | reserved = 0 |
| 319FFF6F | BB1ABE1B | 8A854174 | N/A | CRC |
| D5D5B57C | N/A | D5D5B57C | N/A | EOF, primitives not scrambled |

Note: All values in hexadecimal, shown 31:0

The transmitted dwords for this register FIS, prior to 8b/10b encoding, are :

SOF
c2e2f6aa
fe05f60f
a508436c
3452d356
8a559502
8A854174
EOF

# APPENDIX B. Type field value selection

The values for the TYPE field of the FIS's have been selected in order to provide additional robustness. In minimally buffered implementations that may not buffer a complete FIS, the state machines may begin acting on the received FIS TYPE value prior to the ending CRC having been checked. Because the TYPE value may be acted upon prior to the integrity of the complete FIS being checked against its ending CRC, the TYPE field values have been selected to maximize the Hamming distance between them.

Several considerations are made in selecting the TYPE field values. Since the value is 8b/10b encoded, the Hamming distance for the values after 10b encoding is maximized. Since the starting running disparity at the time the TYPE field is transmitted is not known, values have been selected that have the same encoding for both negative and positive starting running disparity.

## B.1 Type field values

Table 30 lists the selected TYPE field values (with their 10b encoding) that have a Hamming distance of 4 or greater in the 10b space after scrambling and encoding and which have the same encoding for both positive and negative running disparity.

**Table 30. – Type field values**

| Type field value (hex) | Scrambler syndrome[1] (hex) | 8b scrambled value (hex) | 10b encoding (binary) |
|---|---|---|---|
| 0x27 | 0x8D | 0xAA | 0101011010 |
| 0x34 | 0x8D | 0xB9 | 1001101010 |
| 0x39 | 0x8D | 0xB4 | 0010111010 |
| 0x41 | 0x8D | 0xCC | 0011010110 |
| 0x46 | 0x8D | 0xCB | 1101000110 |
| 0x58 | 0x8D | 0xD5 | 1010100110 |
| 0x5F | 0x8D | 0xD2 | 0100110110 |
| 0xA1 | 0x8D | 0x2C | 0011011001 |
| 0xA6 | 0x8D | 0x2B | 1101001001 |
| 0xB8 | 0x8D | 0x35 | 1010101001 |
| 0xBF | 0x8D | 0x32 | 0100111001 |
| 0xC7 | 0x8D | 0x4A | 0101010101 |
| 0xD4 | 0x8D | 0x59 | 1001100101 |
| 0xD9 | 0x8D | 0x54 | 0010110101 |

NOTE –
1. The scrambler syndrome generator is reset on the SOF primitive and the type field value immediately follows the SOF primitive. The scrambler syndrome at the time the type field value is transmitted is therefore deterministic and is equal to the seed used for the scrambling generator. See the scrambler generator section for more information on the details of the scrambler and its generated syndromes.

# APPENDIX C. Further information about cable and connector

## C.1 Device connector configurations

Serial ATA defines several device connector configurations, as described below.

### C.1.1 Configuration 35A1

Configuration 35A1 is the configuration for 3.5" form factor devices without any legacy connector support.

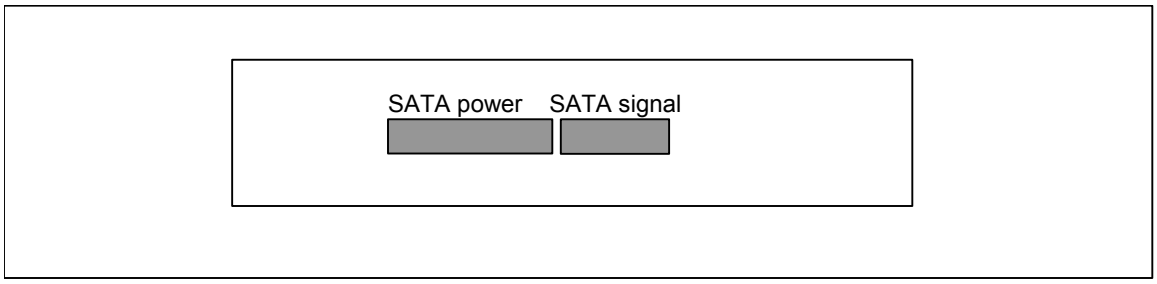



**Figure 68 - Device connector configuration 35A1**

### C.1.2 Configuration 35B1

Configuration 35B1 includes Serial ATA signal and power connectors and legacy jumper and power connectors.

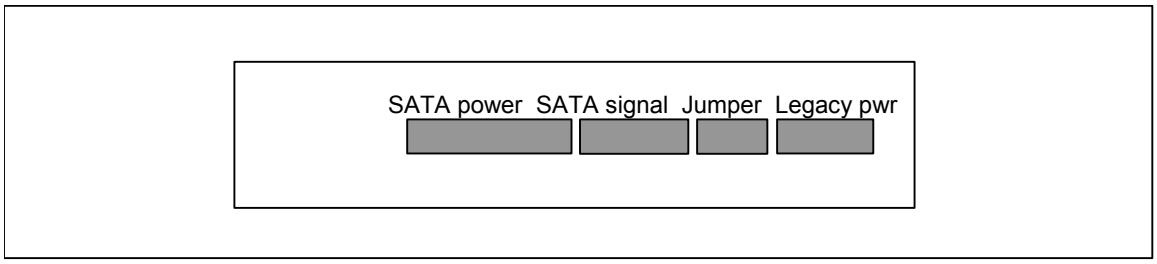



**Figure 69 - Device connector configuration 35B1**

### Configuration 35B2

Configuration 35B2 includes a Serial ATA signal connector and legacy jumper and power connectors.

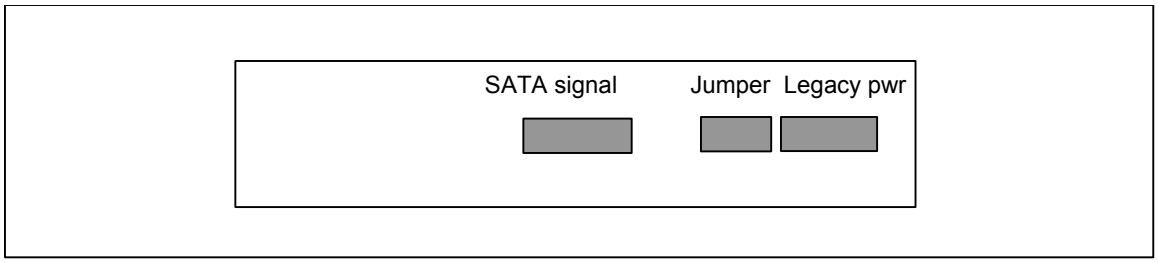



**Figure 70 - Device connector configuration 35B2**

### C.1.3 Configuration 35B3

Configuration 35B3 includes a Serial ATA signal connector and a legacy power connector.

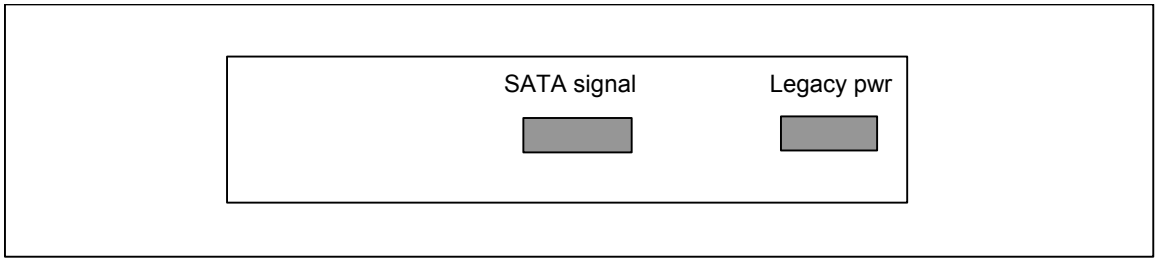



**Figure 71 - Device connector configuration 35B3**

### C.1.4 Configuration 35B4

Configuration 35B4 includes Serial ATA signal and power connectors and a legacy jumper connector.

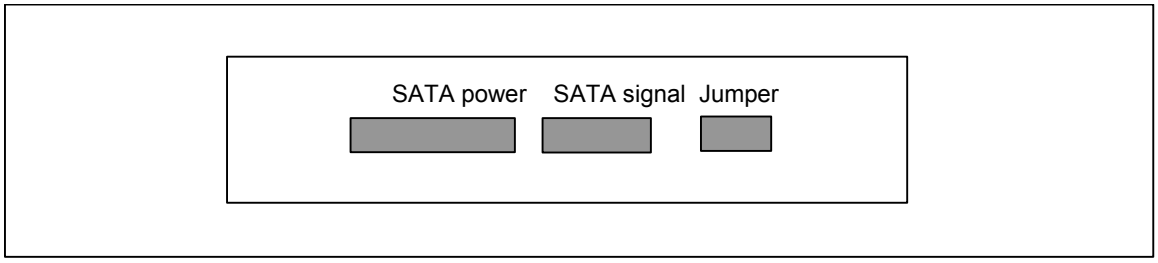



**Figure 72 - Device connector configuration 35B4**

### C.1.5 Configuration 35B5

Configuration 35B5 includes Serial ATA signal and power connectors and a legacy power connector.

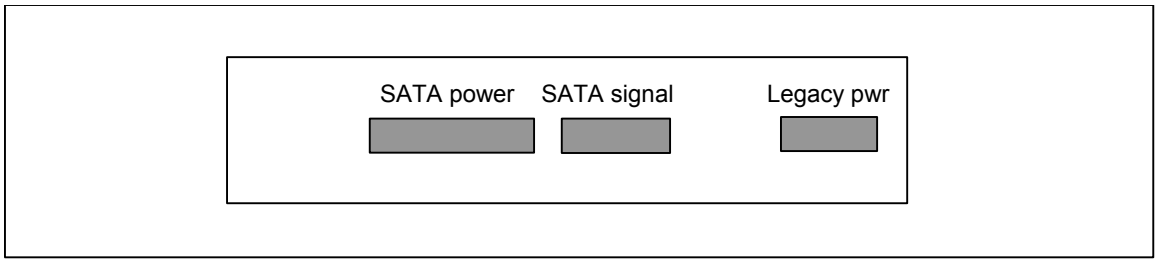



**Figure 73 - Device connector configuration 35B5**

### C.1.6 Configuration 35C1

Configuration 35C1 includes a legacy Parallel ATA signal connector, a Serial ATA signal connector and a legacy power connector.

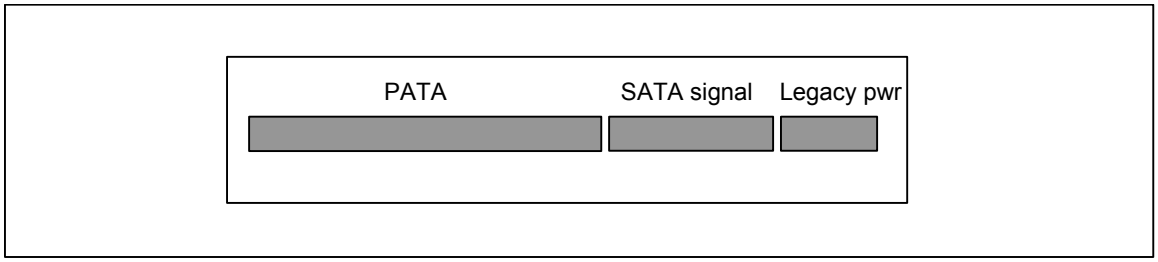



**Figure 74 - Device connector configuration 35C1**

The presence of the parallel ATA connector prevents the same connector location as for the other 3.5” device configurations.

### C.1.7 Configuration 25A1

Configuration 25A1 is the only configuration allowed for 2.5" form factor devices.

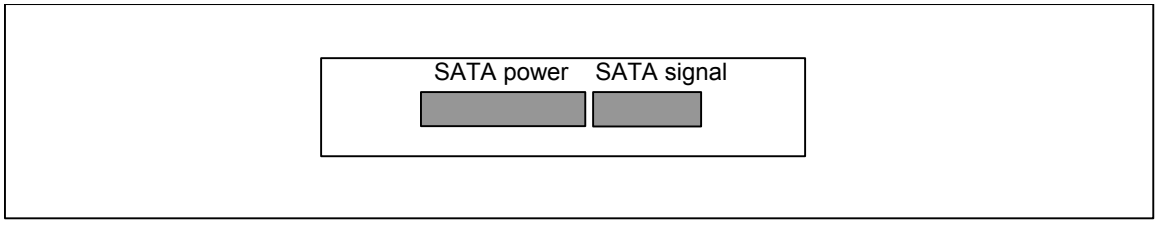
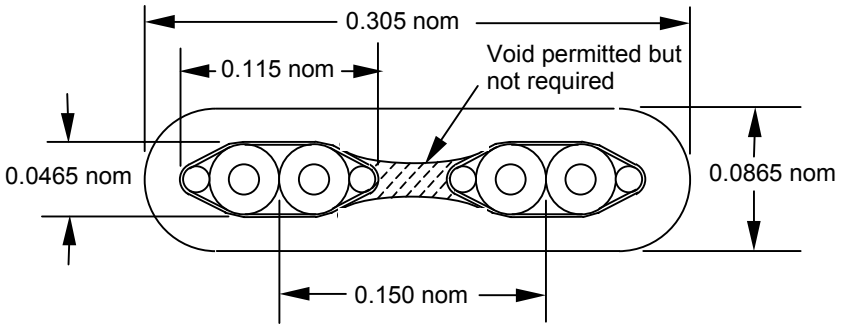



**Figure 75 - Device connector configuration 25A1**

## C.2 Cable construction example

Figure 76 shows a Serial ATA cable construction example.




Electrical
Impedance: 100 ± 5 Ω (differential)
Capacitance: 42 pf/M nominal
Prop delay: 4.25 ns/M nominal
Skew (within pair): TBD (TDT method, drains gounded) (differential 50%-50%, Tektronix 11801, SD-24/SD-26 sampling heads)
Attenuation (nominal): TBD db/M @ 4.5 GHz

Physical
(2) shielded pairs
26 AWG solid tinned copper
0.0435 nom diameter foam polyolefin, white
Parallel pair with (2) drains, 28 AWG solid tinned copper
0.001aluminized polyester, foil in, 0.035 inch min overlap
Blue typical longitudinal wrap, heat sealed

Jacket: 0.020 nom wall PVC, red

All dimensions are inches unless otherwise noted

**Figure 76 – Cable construction example**

## C.3 Contact material and plating

Table 31 shows the recommendations for the contact material and plating.

**Table 31 – Contact material and plating recommendations**

| Parameter | Recommendation | Comments |
|---|---|---|
| Material | Copper alloys, for example, brass for plug contacts and phosphor bronze for receptacle contact, the spring. | Material temper and thickness should be selected based on normal force and elastic deflection range consideration. |
| Mating Side Plating | For 50 durability cycles:<br>• 1.27 μm (50 μin) minimum Ni with either 0.38 μm (15 μin) minimum Au or 0.38 μm (15 μin) minimum 80/20 Pd/Ni with 0.051 μm (2 μin) minimum Au flash<br>For 500 durability cycles:<br>• 1.27 μm (50 μin) minimum Ni with either 0.76 μm (30 μin) minimum Au or 0.76 μm (30 μin) minimum 80/20 Pd/Ni with 0.051 μm (2 μin) minimum Au flash | Exposed underplate or base material is not allowed in the mating area. |
| Solder Side Plating | Either Sn/Pb plating or Pd/Ni with Au flash:<br>• 1.27 μm (50 μin) minimum Ni with 3.18 μm (125 μin) minimum Sn/Pb. Or<br>• 1.27 μm (50 μin) minimum Ni with 0.76 μm (30 μin) minimum 80/20 Pd/Ni with 0.051 μm (2 μin) minimum Au flash. | Exposed base material is allowed in small areas where the contact is excised from its carrier strip or bandolier. |

# Appendix D. Command processing overview

## D.1 Non-data commands

When the Command register is written by the BIOS or software driver, the host adapter sets BSY in the shadow Status register and transmits a Command frame to the device.

When command actions are complete, the device transmits a Register frame to set ending content of the shadow registers

### D.1.1 Legacy DMA read by host from device

- Prior to the command being issued to the device, the host driver software programs the host adapter's DMA controller with the memory address pointer(s) and the transfer direction, and arms the DMA controller (enables the "run" flag).
- The host driver software issues the command to the device by writing the Shadow Register Block Registers (command register last).
- In response to the command Shadow Register Block Register being written, the host adapter sets the BSY bit in the status shadow register and transmits a Register – Host to Device FIS to the device with the Shadow Register Block contents.
- When the device has processed the command and is ready, it transmits the read data to the host in the form of one or more Data FIS's. This transfer proceeds in response to flow control signals/readiness.
- The host adapter recognizes that the incoming frame is a Data FIS and the DMA controller is programmed, and directs the incoming data to the host adapter's DMA controller which forwards the incoming data to the appropriate host memory locations.
- Upon completion of the transfer, the device transmits a Register – Device to Host FIS to indicate ending status for the command, clearing the BSY bit in the Status register, and if the interrupt flag is set in the header an interrupt is asserted to the host.

In some error conditions there may be no Data FIS transmitted prior to the Register FIS being transmitted with the status information. If so, the host software driver shall abort the setup of the DMA controller.

### D.1.2 Legacy DMA write by host to device

- Prior to the command being issued to the device, the host driver software programs the host adapter's DMA controller with the memory address pointer(s) and the transfer direction, and arms the DMA controller (enables the "run" flag). As a result the DMA controller becomes armed but remains paused pending a signal from the device to proceed with the data transfer.
- The host driver software issues the command to the device by writing the Shadow Register Block Registers (command register last).
- In response to the command Shadow Register Block Register being written, the host adapter sets the BSY bit in the status shadow register and transmits a Register – Host to Device FIS to the device with the Shadow Register Block contents.
- When the device is ready to receive the data from the host, the device transmits a DMA Activate FIS to the host, which activates the armed DMA controller. The DMA controller transmits the write data to the device in the form of one or more Data FIS. If more than one data FIS is required to complete the overall data transfer request, a DMA Activate FIS shall be sent prior to each and every one of the subsequent data FIS's. The amount of data transmitted to the device is determined by the transfer count programmed into the host adapter's DMA controller by the host driver software during the command setup phase.
- Upon completion of the transfer, the device transmits a Register – Device to Host FIS to indicate ending status for the command, clearing the BSY bit in the Status register, and if the interrupt flag is set in the header an interrupt is asserted to the host.

In some error conditions the device may signal an ending status by transmitting a register frame to the host without having transmitted a DMA Activate FIS. In such cases the host driver software should abort and clean up the DMA controller.

### D.1.3 PIO data read from the device

- The host driver software issues a PIO read command to the device by writing the Shadow Register Block Registers (command register last).
- In response to the command register being written, the host adapter sets the BSY bit in the status shadow register and transmits a Register – Host to Device FIS to the device with the Shadow Register Block contents.
- When the device has processed the command and is ready to begin transferring data to the host, it first transmits a PIO Setup FIS to the host. Upon receiving the PIO Setup FIS, the host adapter holds the FIS contents in a temporary holding buffer.
- The device follows the PIO Setup FIS with a Data – Device to Host FIS. Upon receiving the Data FIS while holding the PIO Setup FIS, the host adapter transfers the register contents from the PIO Setup FIS into the shadow registers including the initial status value, resulting in DRQ getting set and BSY getting cleared in the Status register. Also, if the interrupt flag is set, an interrupt is generated to the host.
- The host controller receives the incoming data that is part of the Data FIS into a speed matching FIFO that is conceptually attached to the Shadow Register data Block Register.
- As a result of the issued interrupt and DRQ being set in the Status register, host software does a REP INSW on the data register and pulls data from the head of the speed matching FIFO while the serial link is adding data to the tail of the FIFO. The flow control scheme handles data throttling to avoid underflow/overflow of the receive speed matching FIFO that feeds the data Shadow Register Block Register.
- When the number of words read by host software from the Data shadow register reaches the value indicated in the PIO Setup FIS, the host transfers the ending status value from the earlier PIO Setup FIS into the status shadow register resulting in DRQ being cleared and the ending status reported.
- If there are more data blocks to be transferred, the ending status will indicate BSY, and the process will repeat from the device sending the PIO Setup FIS to the host.

### D.1.4 PIO data write to the device

- The host driver software issues a PIO write command to the device by writing the Shadow Register Block Registers (command register last).
- In response to the command register being written, the host adapter sets the BSY bit in the status shadow register and transmits a Register – Host to Device FIS to the device with the Shadow Register Block contents.
- When the device is ready to receive the PIO write data, it transmits a PIO Setup FIS to the host to indicate that the target is ready to receive PIO data and the number of words of data that are to be transferred.
- In response to a PIO Setup FIS with the D bit indicating a write to the device, the host transfers the beginning Status register contents from the PIO Setup FIS into the shadow Status register, resulting in DRQ getting set, BSY cleared. Also, if the interrupt flag is set, an interrupt is generated to the host.
- As a result of DRQ being set in the shadow Status register, the host driver software starts a REP OUTSW to the data register.
- The data written to the data register is placed in an outbound speed matching FIFO and is transmitted to the device as a Data – Host to Device FIS. The REP OUTSW pushes data onto the tail of the FIFO and the serial link pulls data from the head. The flow control scheme handles data throttling to avoid underflow of the transmit FIFO.
- When the number of words indicated in the PIO Setup FIS have been written to the transmit FIFO, the host controller transfers the final status value indicated in the PIO Setup frame into the shadow Status register resulting in DRQ being cleared, and closes the frame with a CRC

and EOF. If additional sectors of data are to be transferred, the ending status value transferred to the status shadow register would have the BSY bit set and the state is the same as immediately after the command was first issued to the device.

- If there are more data blocks to be transferred, the ending status will indicate BSY, and the process will repeat from the device sending the PIO Setup FIS to the host.
- When the number of sectors indicated in the Sector Count register have been transferred, the device shall send a Register – Device to Host FIS with the command complete interrupt and not BSY status.
- In the case of a write error, the device may, on any sector boundary include error status and a command complete interrupt in the PIO setup FIS, and there is no need to send the Register – Device to Host FIS.

## D.1.5 Queued DMA read from device

NOTE – Serial ATA devices may choose to not implement legacy ATA queuing in favor of a more efficient native Serial ATA queuing mechanism.

- Prior to the command being issued to the device, the host driver software programs the host-side DMA controller with the memory address pointer(s) and the transfer direction, and arms the DMA controller (enables the “run” flag).
- The host driver software issues the command to the device by writing the Shadow Register Block Registers (command register last).
- In response to the command Shadow Register Block Register being written, the host adapter sets the BSY bit in the status shadow register and transmits a Register – Host to Device FIS to the device with the Shadow Register Block contents.
- When the device has queued the command and wishes to release the bus, it transmits a register FIS to the host resulting in the BSY bit being cleared and the REL bit being set in the Status register.
- When the device is ready to complete the transfer for the queued command, it transmits a Set Device Bits FIS to the host resulting in the SERV bit being set in the Status register. If no other command is active (i.e. BSY set to one), then an interrupt is also generated.
- In response to the service request, the host software deactivates the DMA controller (if activated) and issues a SERVICE command to the device by writing the Shadow Register Block Registers, resulting in the BSY bit getting set and a register FIS being transmitted to the device.
- In response to the SERVICE request, the device transmits a register FIS to the host conveying the TAG value to the host and clearing the BSY bit and settings the DRQ bit.
- When the DRQ bit is set, the host software reads the TAG value from the Shadow Register Block and restores the DMA controller context appropriate for the command that is completing.
- The device transmits the read data to the host in the form of one or more Data FIS. This transfer proceeds in response to flow control signals/readiness. Any DMA data arriving before the DMA controller has its context restored will back up into the inbound speed-matching FIFO until the FIFO is filled and will thereafter be flow controlled to throttle the incoming data until the DMA controller has its context restored by host software.
- The host controller recognizes that the incoming packet is a Data FIS and the DMA controller is programmed, and directs the incoming data to the host controller’s DMA controller that forwards the incoming data to the appropriate host memory locations.
- Upon completion of the transfer, the target transmits a Register – Device to Host FIS to indicate ending status for the command, clearing the BSY bit in the Status register, and if the interrupt flag is set in the header an interrupt is asserted to the host.

## D.1.6 Queued DMA write to device

NOTE – Serial ATA devices may choose to not implement legacy ATA queuing in favor of a more efficient native Serial ATA queuing mechanism.

- Prior to the command being issued to the device, the host driver software programs the host-side DMA controller with the memory address pointer(s) and the transfer direction, and arms the DMA controller (enables the “run” flag).
- The host driver software issues the command to the device by writing the Shadow Register Block Registers (command register last).
- In response to the Shadow Register command Block Register being written, the host adapter sets the BSY bit in the status shadow register and transmits a Register – Host to Device FIS to the device with the Shadow Register Block contents.

- When the device has queued the command and wishes to release the bus, it transmits a register FIS to the host resulting in the BSY bit being cleared and the REL bit being set in the Status register.
- When the device is ready to complete the transfer for the queued command, it transmits a Set Device Bits FIS to the host resulting in the SERV bit being set in the Status register. If no other command is active (i.e. BSY set to one), then an interrupt is also generated.
- In response to the service request, the host software deactivates the DMA controller (if activated) and issues a SERVICE command to the device by writing the Shadow Register Block Registers, resulting in the BSYT bit getting set and a register FIS being transmitted to the device.
- In response to the SERVICE request, the device transmits a register FIS to the host conveying the TAG value to the host and clearing the BSY bit and setting the DRQ bit.
- When the DRQ bit is set, the host software reads the TAG value from the Shadow Register Block and restores the DMA controller context appropriate for the command that is completing.
- When the device is ready to receive the data from the host, the device transmits a DMA Activate FIS to the host, which activates the armed DMA controller. The DMA controller transmits the write data to the device in the form of one or more Data – Host to Device FIS's. If more than one data FIS is required to complete the overall data transfer request, a DMA Activate FIS shall be sent prior to each and every one of the subsequent data FIS's. The amount of data transmitted to the device is determined by the transfer count programmed into the host's DMA controller by the host driver software during the command setup phase. If the DMA Activate FIS arrives at the host prior to host software restoring the DMA context, the DMA Activate FIS results in the DMA controller starting the transfer as soon as the host software completes programming it (i.e. the controller is already activated, and the transfer starts as soon as the context is restored).
- Upon completion of the transfer, the target transmits a Register – Host to Device FIS to indicate ending status for the command, clearing the BSY bit in the Status register, and if the interrupt flag is set in the header an interrupt is asserted to the host.

### D.1.7 ATAPI Packet commands with PIO data in

- The host driver software issues a PACKET command to the device by writing the Shadow Register Block Registers (command register last).
- In response to the command register being written, the host adapter sets the BSY bit in the status shadow register and transmits a Register – Host to Device FIS to the device with the Shadow Register Block contents.
- When the device is ready to receive the ATAPI command packet, it transmits a PIO Setup – Device to Host FIS to the host to indicate that the target is ready to receive PIO data and the number of words of data that are to be transferred.
- The host transfers the beginning Status register contents from the PIO Setup FIS into the shadow Status register, resulting in BSY getting deasserted and DRQ getting asserted.
- As a result of BSY getting cleared and DRQ being set in the shadow Status register, the host driver software writes the command packet to the Shadow Register Block data register.
- The data written to the data register is placed in an outbound speed matching FIFO and is transmitted to the device as a Data – Host to Device FIS. The writes to the data register push data onto the tail of the FIFO and the serial link pulls data from the head. The flow control scheme handles data throttling to avoid underflow of the transmit FIFO.
- When the number of words indicated in the PIO Setup FIS have been written to the transmit FIFO, the host controller transfers the final status value indicated in the PIO setup frame into the shadow Status register resulting in DRQ being cleared and BSY being set, and closes the frame with a CRC and EOF. This completes the transmission of the command packet to the device.

- When the device has processed the command and is ready to begin transferring data to the host, it first transmits a PIO Setup – Device to Host FIS to the host. Upon receiving the PIO Setup – Device to Host FIS, the host adapter holds the FIS contents in a temporary holding buffer.
- The device follows the PIO Setup – Device to Host FIS with a Data – Device to Host FIS. Upon receiving the Data FIS while holding the PIO Setup FIS context, the host adapter transfers the register contents from the PIO Setup FIS into the shadow registers including the initial status value, resulting in DRQ getting set and BSY getting cleared in the Status register. Also, if the interrupt flag is set, an interrupt is generated to the host.
- The host controller receives the incoming data that is part of the Data FIS into a speed matching FIFO that is conceptually attached to the data Shadow Register Block Register.
- As a result of the issued interrupt and DRQ being set in the Status register, host software reads the byte count and does a REP INSW on the data register to pull data from the head of the speed matching FIFO while the serial link is adding data to the tail of the FIFO. The flow control scheme handles data throttling to avoid underflow/overflow of the receive speed matching FIFO that feeds the data Shadow Register Block Register.
- When the number of words received in the Data FIS reaches the value indicated in the PIO Setup FIS and the host FIFO is empty, the host transfers the ending status value from the earlier PIO Setup into the status shadow register resulting in DRQ being cleared and, BSY being set.
- The device transmits final ending status by sending a register FIS with BSY cleared to zero and the ending status for the command and the interrupt flag set.
- The host detects an incoming register frame that contains BSY cleared to zero and ending status for the command and the interrupt flag set and places the frame content into the shadow registers to complete the command.

### D.1.8 ATAPI Packet commands with PIO data out

- The host driver software issues a PACKET command to the device by writing the Shadow Register Block Registers (command register last).
- In response to the command register being written, the host adapter sets the BSY bit in the status shadow register and transmits a Register – Host to Device FIS to the device with the Shadow Register Block contents.
- When the device is ready to receive the ATAPI command packet, it transmits a PIO Setup – Device to Host FIS to the host to indicate that the target is ready to receive PIO data and the number of words of data that are to be transferred.
- The host transfers the beginning Status register contents from the PIO Setup – Device to Host FIS into the shadow Status register, resulting in BSY getting deasserted and DRQ getting asserted.
- As a result of BSY getting cleared and DRQ being set in the shadow Status register, the host driver software writes the command packet to the Shadow Register Block data register.
- The data written to the data register is placed in an outbound speed matching FIFO and is transmitted to the device as a Data – Host to Device FIS. The writes to the data register push data onto the tail of the FIFO and the serial link pulls data from the head. The flow control scheme handles data throttling to avoid underflow of the transmit FIFO.
- When the number of words indicated in the PIO Setup FIS have been written to the transmit FIFO, the host controller transfers the final status value indicated in the PIO Setup frame into the shadow Status register resulting in DRQ being cleared and BSY being set, and closes the frame with a CRC and EOF. This completes the transmission of the command packet to the device.
- When the device has processed the command and is ready to receive the PIO write data, it transmits a PIO Setup – Device to Host FIS to the host to indicate that the target is ready to receive PIO data and the number of words of data that are to be transferred.

- In response to a PIO Setup FIS with the D bit indicating a write to the device, the host transfers the beginning Status register contents from the PIO Setup FIS into the shadow Status register, resulting in DRQ getting set. Also, if the interrupt flag is set, an interrupt is generated to the host.
- As a result of DRQ being set in the shadow Status register, the host driver software reads the byte count and starts a REP OUTSW to the data register.
- The data written to the data register is placed in an outbound speed matching FIFO and is transmitted to the device as a Data – host to device FIS. The REP OUTSW pushes data onto the tail of the FIFO and the serial link pulls data from the head. The flow control scheme handles data throttling to avoid underflow of the transmit FIFO.
- When the number of words indicated in the PIO Setup FIS have been written to the transmit FIFO, the host controller transfers the final status value indicated in the PIO Setup FIS into the shadow Status register resulting in DRQ being cleared, BSY being set, and closes the frame with a CRC and EOF.
- The device transmits final ending status by sending a Register – Device to Host FIS with BSY cleared to zero and the ending status for the command and the interrupt flag set.
- The host detects an incoming Register – Device to Host FIS that contains BSY cleared to zero and ending status for the command and the interrupt flag set and places the frame content into the shadow registers to complete the command.

### D.1.9 ATAPI Packet commands with DMA data in

- Prior to the command being issued to the device, the host driver software programs the host-side DMA controller with the memory address pointer(s) and the transfer direction, and arms the DMA controller (enables the “run” flag).
- The host driver software issues a PACKET command to the device by writing the Shadow Register Block Registers (command register last).
- In response to the command register being written, the host adapter sets the BSY bit in the status shadow register and transmits a Register – Host to Device FIS to the device with the Shadow Register Block contents.
- When the device is ready to receive the ATAPI command packet, it transmits a PIO Setup – Device to Host FIS to the host to indicate that the target is ready to receive PIO data and the number of words of data that are to be transferred.
- The host transfers the beginning Status register contents from the PIO Setup FIS into the shadow Status register, resulting in BSY getting deasserted and DRQ getting asserted.
- As a result of BSY getting cleared and DRQ being set in the shadow Status register, the host driver software writes the command packet to the Shadow Register Block data register.
- The data written to the data register is placed in an outbound speed matching FIFO and is transmitted to the device as a Data – Host to Device FIS. The writes to the data register push data onto the tail of the FIFO and the serial link pulls data from the head. The flow control scheme handles data throttling to avoid underflow of the transmit FIFO.
- When the number of words indicated in the PIO Setup FIS have been written to the transmit FIFO, the host controller transfers the final status value indicated in the PIO setup frame into the shadow Status register resulting in DRQ being cleared and BSY being set, and closes the frame with a CRC and EOF. This completes the transmission of the command packet to the device.
- When the device has processed the command and is ready, it transmits the read data to the host in the form of a single Data FIS. This transfer proceeds in response to flow control signals/readiness.
- The host controller recognizes that the incoming packet is a Data FIS and the DMA controller is programmed, and directs the incoming data to the host controller’s DMA controller that forwards the incoming data to the appropriate host memory locations.

- Upon completion of the transfer, the target transmits a Register – Device to Host FIS to indicate ending status for the command, clearing the BSY bit in the Status register, and if the interrupt flag is set in the header an interrupt is asserted to the host.

### D.1.10 ATAPI Packet commands with DMA data out

- Prior to the command being issued to the device, the host driver software programs the host-side DMA controller with the memory address pointer(s) and the transfer direction, and arms the DMA controller (enables the “run” flag). As a result the DMA controller becomes armed but remains paused pending a signal from the device to proceed with the data transfer.
- The host driver software issues a PACKET command to the device by writing the Shadow Register Block Registers (command register last).
- In response to the command register being written, the host adapter sets the BSY bit in the status shadow register and transmits a Register – Host to Device FIS to the device with the Shadow Register Block contents.
- When the device is ready to receive the ATAPI command packet, it transmits a PIO Setup – Device to Host FIS to the host to indicate that the target is ready to receive PIO data and the number of words of data that are to be transferred.
- The host transfers the beginning Status register contents from the PIO Setup FIS into the shadow Status register, resulting in BSY getting deasserted and DRQ getting asserted.
- As a result of BSY getting cleared and DRQ being set in the shadow Status register, the host driver software writes the command packet to the Shadow Register Block data register.
- The data written to the data register is placed in an outbound speed matching FIFO and is transmitted to the device as a Data – Host to Device FIS. The writes to the data register push data onto the tail of the FIFO and the serial link pulls data from the head. The flow control scheme handles data throttling to avoid underflow of the transmit FIFO.
- When the number of words indicated in the PIO Setup FIS have been written to the transmit FIFO, the host controller transfers the final status value indicated in the PIO Setup frame into the shadow Status register resulting in DRQ being cleared and BSY being set, and closes the frame with a CRC and EOF. This completes the transmission of the command packet to the device.
- When the device is ready to receive the data from the host, the device transmits a DMA Activate FIS to the host, which activates the armed DMA controller. The DMA controller transmits the write data to the device in the form of one or more Data FIS. The transfer proceeds in response to flow control signals/readiness. The amount of data transmitted to the device is determined by the transfer count programmed into the host’s DMA engine by the host driver software during the command setup phase.
- Upon completion of the transfer, the device transmits a Register – Device to Host FIS to indicate ending status for the command, clearing the BSY bit in the Status register, and if the interrupt flag is set in the header an interrupt is asserted to the host.

### D.1.11 First-party DMA read of host memory by device

This section only outlines the basic First-party DMA read transaction and does not outline command sequences that utilize this capability.

- As a result of some condition and the target having adequate knowledge of host memory (presumably some prior state has been conveyed to the target), the target signals a DMA read request for a given address and given transfer count to the host by transmitting a First-Party DMA Setup FIS.
- Upon receiving the First-Party DMA Setup FIS, the host controller transfers the appropriate memory address, count value, and transfer direction into the host-side DMA controller and arms and activates the DMA controller (enables the “run” flag).

- In response to being set up, armed, and activated, the DMA controller retrieves data from host memory and transmits it to the device in the form of one or more Data – Host to Device FIS's. The transfer proceeds in response to flow control signals/readiness.

### D.1.12 First-party DMA write of host memory by device

This section only outlines the basic first-party DMA write transaction and does not outline command sequences which utilize this capability.

- As a result of some condition and the target having adequate knowledge of host memory (presumably some prior state has been conveyed to the target), the target signals a DMA write request for a given address and given transfer count to the host by transmitting a First-party DMA Setup FIS.
- Upon receiving the First-party DMA Setup FIS, the host controller transfers the appropriate memory address, count value, and transfer direction into the host-side DMA controller and arms the DMA controller (enables the “run” flag).
- The device then transmits the data to the host in the form of one or more Data – Device to Host FIS’s. This DMA transfer proceeds in response to flow control signals/readiness.
- The host controller recognizes that the incoming packet is a Data FIS and directs the incoming data to the host controller’s DMA controller which forwards the incoming data to the appropriate host memory locations.

### D.1.13 Odd word count considerations

This section outlines special considerations required to accommodate data transfers of an odd number of 16-bit word quantities. The considerations are separately outlined for each of the data transaction types. No accommodation in Serial ATA is made for the transfer of an odd number of 8-bit byte quantities.

### D.1.13.1 Legacy DMA read from target for odd word count

- Prior to the command being issued to the device, the host driver software programs the host-side DMA controller with the memory address pointer(s) and the transfer direction, and arms the DMA controller (enables the “run” flag). The count for the DMA transfer is an odd number of word (16-bit) quantities.
- When the device has processed the corresponding command and is ready to transmit the data to the host, it does so in the form of one or more data FIS. Because the transfer count is odd, the last 32-bit Dword transmitted to the host has the high order 16 bits padded with zeroes. The CRC value transmitted at the end of the FIS is computed over the entire FIS including any pad bytes in the final transmitted symbol.
- The host controller receives the incoming data and the DMA controller directs the received data from the receive FIFO to the appropriate host memory locations. The DMA controller has a transfer granularity of a 16-bit word (consistent with the 8038i specification).
- Upon receiving the final 32-bit Dword of receive data, the DMA controller transfers the first half (low order 16 bits) to the corresponding final memory location at which point the DMA engine’s transfer count is exhausted. The DMA controller drops the high-order 16 bits of the final received Dword since it represents data received beyond the end of the requested DMA transfer. The dropped 16 high order bits corresponds with the 16 bits of transmission pad inserted by the sender.

### D.1.13.2 Legacy DMA write by host to target for odd word count

- Prior to the command being issued to the device, the host driver software programs the host-side DMA controller with the memory address pointer(s) and the transfer direction, and arms the DMA controller (enables the “run” flag). The count for the DMA transfer is an odd number of word (16-bit) quantities.
- When the device has processed the corresponding command and is ready to receive the data from the host, it signals readiness with a DMA Activate FIS.
- Upon receiving the DMA Activate signal, the host transmits the data to the device in the form of one or more data FIS. Because the transfer count is odd, the DMA controller completes its data transfer from host memory to the transmit FIFO after filling only the low order 16-bits of the last Dword in the FIFO, leaving the upper 16 bits zeroed. This padded final Dword is transmitted as the final symbol in the data frame. The CRC value transmitted at the end of the FIS is computed over the entire FIS including any pad bytes in the final transmitted symbol.
- Having awareness of the command set and having decoded the current command, the device that receives the transmitted data has knowledge of the expected data transfer length. Upon receiving the data from the host, the device removes the 16-bit pad data in the upper 16 bits of the final 32-bit Dword of received data.

### D.1.13.3 PIO data read from the device

- In response to decoding and processing a PIO read command with a transfer count for an odd number of 16-bit words, the device transmits the corresponding data to the host in the form of a single Data FIS. The device pads the upper 16-bits of the final 32-bit Dword of the last transmitted FIS in order to close the FIS. The CRC value transmitted at the end of the FIS is computed over the entire FIS including any pad bytes in the final transmitted symbol.
- Host driver software responsible for retrieving the PIO data is aware of the number of words of data it expects to retrieve from the taskfile data register and performs a REP INSW operation for an odd number of repetitions.
- Upon exhaustion of the REP INSW operation by the host driver software, the receive FIFO that interfaces with the data register has one 16-bit word of received data remaining in it that corresponds to the pad that the device included at the end of the transmitted frame. This remaining word of data left in the data register FIFO is flushed upon the next write of the command taskfile register or upon the receipt of the next data FIS from the device.

### D.1.13.4 PIO data write to the device

- In response to decoding and processing a PIO write command with a transfer count for an odd number of 16-bit words, the device transmits a PIO Setup FIS to the host indicating it is ready to receive the PIO data and indicating the transfer count. The conveyed transfer count is for an odd number of 16-bit word quantities.
- Host driver software responsible for transmitting the PIO data is aware of the number of words of data it will write to the taskfile data register and performs a REP OUTSW operation for an odd number of repetitions.
- After the final write by the software driver to the taskfile data register, the transfer count indicated in the PIO Setup packet is exhausted, which signals the host controller to close the FIS. Since the transfer count was odd, the upper 16 bits of the final 32-bit Dword of data to transmit remains zeroed (pad) and the host controller closes the FIS after transmitting this final padded Dword. The CRC value transmitted at the end of the FIS is computed over the entire FIS including any pad bytes in the final transmitted symbol.
- Having awareness of the command set and having decoded the current command, the device that receives the transmitted data has knowledge of the expected data transfer length. Upon receiving the data from the host, the device removes the 16-bit pad data in the upper 16 bits of the final 32-bit Dword of received data.

### D.1.13.5 First-party DMA read of host memory by device

- When the device wishes to initiate a first-party DMA read of host memory for an odd word count, it constructs a First-party DMA Setup FIS that includes the base address and the transfer count (for an odd number of words).
- Upon receiving the DMA Setup FIS, the host controller transfers the starting address and (odd) transfer count to the DMA controller and activates the DMA controller to initiate a transfer to the device of the requested data.
- The host transmits the data to the device in the form of one or more data FIS. Because the transfer count is odd, the DMA controller completes its data transfer from host memory to the transmit FIFO after filling only the low order 16-bits of the last Dword in the FIFO, leaving the upper 16 bits zeroed. This padded final Dword is transmitted as the final symbol in the data frame. The CRC value transmitted at the end of the FIS is computed over the entire FIS including any pad bytes in the final transmitted symbol.
- Having awareness of the amount of data requested, the device that receives the transmitted data has knowledge of the expected data transfer length. Upon receiving the data from the host, the device removes the 16-bit pad data in the upper 16 bits of the final 32-bit Dword of received data.

### D.1.13.6 First-party DMA write of host memory by device

- When the device wishes to initiate a first-party DMA write to host memory for an odd word count, it constructs a DMA Setup FIS that includes the base address and transfer count (for an odd number of words).
- Upon receiving the DMA Setup FIS, the host controller transfers the starting address and (odd) transfer count to the DMA controller and activates the DMA controller.
- The device transmits the data to the host in the form of one or more data FIS. Because the transfer count is odd, the last 32-bit Dword transmitted to the host has the high order 16 bits padded with zeroes. The CRC value transmitted at the end of the FIS is computed over the entire FIS including any pad bytes in the final transmitted symbol.
- The host controller receives the incoming data and the DMA controller directs the received data from the receive FIFO to the appropriate host memory locations. The DMA controller has a transfer granularity of a 16-bit word (consistent with the 8038i specification).
- Upon receiving the final 32-bit Dword of receive data, the DMA controller transfers the first half (low order 16 bits) to the corresponding final memory location at which point the DMA controller’s transfer count is exhausted. The DMA controller drops the high-order 16 bits of the final received Dword since it represents data received beyond the end of the requested DMA transfer. The dropped 16 high order bits corresponds with the 16 bits of transmission pad inserted by the sender.