TO:	T10 Membership
FROM:	Paul Suhler, Seagate Technology
DATE:	27 November 2001
SUBJECT:	Seagate Automation/Drive Interface Specification (T10/02-002r0)

To close two of my action items (# 4 and 5) from the November meeting, here is the current Seagate ADI specification, known as the “Library Interface Encapsulated SCSI Protocol.”

I look forward to hearing your comments.

Seagate

Viper ® 200

LTO Tape Drive

Library Interface Specification: Encapsulated SCSI Protocol

SRSS Firmware Engineering

19 November 2001

Revision 18

Changes

Rev	Date	Editor	Changes
1	01/29/1998	Adrienne Turenne	• First Draft as *Library Interface Proposal*
2	08/28/1998	Paul Suhler	• Defined SCSI encapsulation protocol.
3	11/06/1998	Paul Suhler	• Added flow control. • Added detailed packet statuses. • Renamed layers and transactions to avoid confusion. • Added character mode to drive display. • Added retry upon collision and put resynch time in link layer. • Reorganized packets into hierarchical structures
4	01/05/2000	Paul Suhler	• First release revision
5	03/14/2000	Paul Suhler	• Added Command Forwarding mode • Deleted LCD control operations • Reorganized Drive Status EVPD Page • Added Target Initiated Bus Control and Disable Wide Bus Mode fields to Interface Control Mode fields to Interface Control mode page.
6	06/27/2000	Paul Suhler	• Added Command Forwarding details • Added Final Data packet.
7	06/27/2000	Paul Suhler	• Minor edits.
8	10/10/2000	Paul Suhler	• Minor updates.
9	12/20/2000	Paul Suhler	• Reorganized Changes table to reflect SourceSafe versions. • Modified document number and headers and footers. • Added CRN to forwarded command payload. • Added definitions and numerical values for timeouts. • Removed references to Interface Disable, which is not implemented. • Removed references to Drive Usage log page, which is documented in Viper SCSI Interface Manual.
10	02/05/2001	Paul Suhler	• Changed descriptions of Drive Load Failed Jammed and Drive Eject Failed Jammed statuses to remove statements that cartridge cannot be ejected.
11	02/13/2001	Paul Suhler	• Added description of Threaded field in Inquiry VPD page DFh.
12	02/26/2001	Günter Knestele	• Added Drive Status for FW, Cleaning and Data tape
13	05/10/2001	Paul Suhler	• Added Table numbers for Drive Status sequences.
14	07/27/2001	Paul Suhler	• Deleted Written and Bus Reset from Drive Status VPD page (DFh).
15	07/27/2001	Paul Suhler	• Revised description of Transport Type, Next Selection ID, and Jumpered Selection ID fields for Fibre Channel.
16	09/20/2001	Paul Suhler	• Revised description of Transport Type to include reporting of link up and down for Fibre Channel.
17	11/12/2001	Paul Suhler	• Added application layer implementation guidelines for libraries. • Added explanation of restricted interaction of reservation and contingent allegiance between library and other initiators.
18	11/19/2001	Paul Suhler	• Revised organizational names.

Table of Contents

Figures

Tables

1. Scope

The Tape Library Interface for Seagate Removable Storage Solutions tape drives is an RS-422 serial interface connecting one drive and the tape library in which it is installed. It provides an out-of-band communication path (i.e., not the parallel SCSI or Fibre Channel transport medium) by which the library can determine the status of and exercise control over the drive. There are two forms of communication over the serial interface: SCSI commands from the library for execution by the drive, and alerts sent to the library to inform it of check conditions not associated with a SCSI command.

This interface specification describes how this information is transmitted between the library and drive. It also describes vendor-unique information provided by the drive. It covers the following areas:

- Application layer – the commands implemented by the drive
- Transaction layer – execution of commands in terms of link requests
- Link layer – the protocol for transmitting information across the interface
- Physical layer – the connectors, cables, levels, and data rates

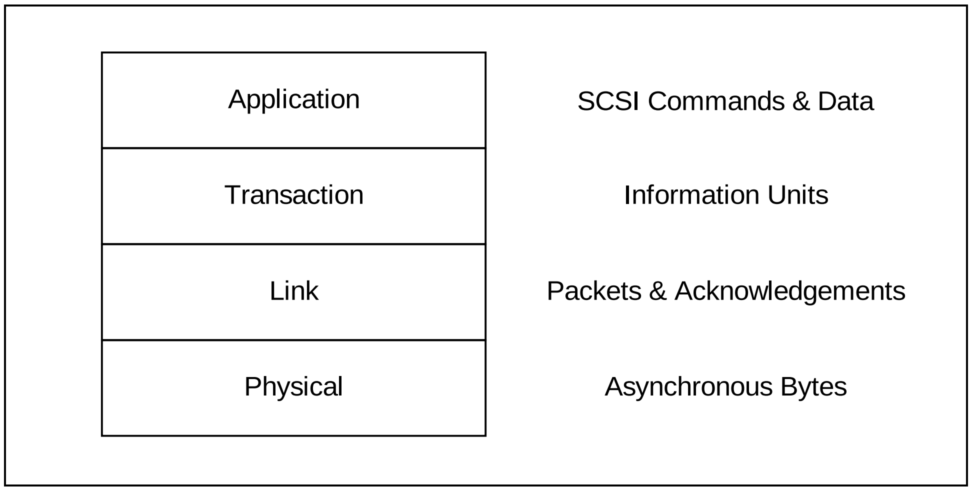


Figure 1. Protocol Layers and Information Transmitted

At the top of the protocol stack is the application layer – the set of commands that must satisfy the functional requirements of the interface; this includes SCSI commands and alerts.

- The only details presented in this document are those of the vendor-specific operations and those of the alerts (section 5.3).
- The transaction layer explains these high-level operations in terms of link-level transactions, each of which delivers a particular type of information (section 5.7).
- The link layer provides encapsulation for different types of information, like alerts and SCSI commands, data, and responses. It provides for detection of transmission errors (section 6.5).
- At the bottom of the stack is the physical layer, an RS-422 serial interface (section 7.3.7).

Because the serial interface connects exactly one initiator with one target, there is no need for destination addressing in the encapsulation protocol. Because data transfers are seldom lengthy, the maximum data payload size is 2k bytes. With these simplifications, this protocol resembles the ANSI Fibre Channel Protocol for SCSI (FCP).

Supported SCSI commands are specified in [LTO-SCSI]; their format and that of requested data is as specified in the SCSI standards. Any command implemented by the drive may be issued through the Library Interface, although large data transfers will require long times, owing to the slow speed of the interface, and should be avoided. Any command supported by the Library Interface is also supported by the SCSI interface.

To provide information not available via the standard inquiry, mode, and log pages, vendor-specific pages and buffers have been implemented.

It is the responsibility of the host and library to coordinate issuing of commands over the SCSI/FC and serial interfaces. Normally, the drive should not be reset over the serial interface while there are active (not hung) commands, which were issued over the SCSI/FC interface.

2. References

[FC-AL-2]	*Fibre Channel Arbitrated Loop*, NCITS 332-1999, standard, ANSI.
[LTO-SCSI]	*Viper 200 SCSI Interface Manual*, Publication 10006952-001, Seagate Removable Storage Solutions.
[SCSI-2]	*Small Computer System Interface – 2*, X3.131, standard, ANSI.
[SAM-2]	*SCSI Architecture Model – 2*, X3T10/1157D, draft standard, ANSI.
[SPC-2]	*SCSI Primary Commands – 2*, X3T10/1236D, draft standard, ANSI.
[SSC]	*SCSI-3 Stream Device Commands*, NCITS 335-2000, standard, ANSI.
[TapeAlert]	*TapeAlert Specification*, issue 2.1, 16 May 1998, Hewlett Packard.

3. Definitions and Abbreviations

Acknowledgement	A one-character response sent from the receiver of a packet to the sender to indicate the success or failure of the transmission.
Action	An operation performed by a state machine when it makes a transition between two states.
AL-PA	Arbitrated Loop Physical Address (Fibre Channel).
BOT	Beginning of tape.
LSB	Least Significant Bit.
MAM	Medium Auxiliary Memory – known as Cartridge Memory in LTO and as Memory in Cartridge in AIT.
LUN	Logical Unit Number.
MSB	Most Significant Bit.
Node	An entity connected by the library serial interface, i.e., a library or a drive.
NPort ID	A 24-bit identifier for a port in a Fibre Channel node. The NPort ID is used by other nodes to address the port/node via the FC interconnect.
Packet	A sequence of data bytes containing one type of information, such as an alert message, a SCSI command, or data, sent from one node to another, plus an acknowledgement byte from the receiver to the sender.
Receiver	The node that receives a packet and transmits an acknowledgement.
Requester	The node that begins a transaction by sending the first packet.
Responder	The node participating in a transaction that it did not initiate, i.e., the recipient of the first packet.
Sender	The node that transmits a packet.
Signal	An event that causes a transition in a state machine.
Transaction	A set of packet transmissions that comprise a complete operation, such as performing a SCSI command and transferring its data and status.

4. Conventions

4.1 Numerical Field Types

All numerical fields in structures are unsigned integers. When a number requires a multiple-byte representation, big-endian format is used, i.e., the byte closest to the beginning of the structure is the most-significant byte.

4.2 Base

Numerals have suffixes that represent their base:

- nnn**b** for binary, where n ∈ {0, 1}
- nnn**h** is hexadecimal, where n ∈ {0, .., 9, A, .., F}
- nnn is decimal, where n ∈ {0, .., 9}, i.e., no suffix

4.3 Field Numbering

In numbering bits within a byte, 7 is the most-significant bit and 0 is the least-significant. In numbering bytes within a structure, the first byte is numbered zero; this is the first byte to be transmitted across the serial interface. See Table 1.

Table 1. Numerical Field Example

Bit / Byte	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	Reserved (0)			Foo (110b)			Bar	Baz (0)

4.4 Fixed Numerical Fields

When a field always takes the same value, the value is shown in parentheses after the field name. In Table 1, **Reserved** is always 000b, **Foo** is always 110b, **Bar** can be zero or one, and **Baz** is always zero.

Unless otherwise stated, reserved fields are always required to be zero.

4.5 Unspecified Field Values

When some of the possible values of a field are enumerated, the remaining unenumerated values are reserved.

5. Application Layer

The application layer is the set of SCSI commands, SCSI task management commands, and drive alerts that are supported by the interface. The SCSI commands are extended with sets of vendor-specific inquiry, mode, and log pages. It also includes the Command Forwarding functionality.

5.1 Supported SCSI Commands

The Library Interface supports all commands implemented by the tape drive. However, the SSC READ, WRITE, and WRITE FILEMARKS commands should not be issued. It is the responsibility of the tape library not to issue commands that will require transferring large quantities of data across the serial interface. Furthermore, no reservation commands should be issued over the library interface, as this would cause a reservation conflict in an initiator connected via the primary interface (Parallel SCSI or Fibre Channel).

To prevent undesirable interactions between commands issued by the library and commands issued by initiators on the primary interface, certain commands are executed in a special manner when received from the library. In particular, commands issued by the library will not set a contingent allegiance condition visible to initiators on the primary interface. Also, reservations made by initiators on the primary interface will not affect commands issued by the library.

5.2 Task Management Commands

Task management commands can always be executed by the drive, whether the drive is in SCSI-2 or SCSI-3 mode. They are implemented as non-data transactions (Figure 3), where a command packet is sent to the drive, and the drive returns a response packet. The implemented commands are shown in Table 2, and the meanings of the responses in Table 3.

Table 2. Task Management Commands

Command	Supported
Abort Task	Yes
Abort Task Set	Yes
Clear ACA	Yes
Clear Task Set	Yes
Target Reset	Yes
Logical Unit Reset	Yes

Table 3. Task Management Command Responses

Response Name	Meaning
Function Complete	Function completed normally
Function in Progress	Function execution underway
Function Rejected	Function not supported

5.3 Alerts

An alert is an unsolicited message from the drive to the library to announce a check condition that is not associated with any command. They are implemented as single-packet transactions. The packet is sent from the drive to the library and the library acknowledges receipt to inform the drive of whether the packet was received correctly. Further actions by the library are taken using SCSI or task management commands, as shown in Table 4.

Alerts are currently not implemented by the drive.

Table 4. Alert Meanings

Alert Name	Recommended Action by Library
Unit Attention	Issue Request Sense
Detect Deferred Errors	Issue Request Sense
Tape Alert	Issue Log Select for Tape Alert page, 2Eh

5.4 Command Forwarding

When Command Forwarding is enabled, the drive acts as a bridge between the primary interface (parallel SCSI or Fibre Channel) and the library interface. SCSI commands received over a primary interface port and addressed to LUN 1 are retransmitted to the tape library via the library interface. Data and command status received from the library are retransmitted over the primary interface to the initiator. If the primary interface is parallel SCSI, then the drive will disconnect from the bus after receipt of the CDB. The drive does no checking of the command's validity.

Command Forwarding mode is controlled by the **CmdFwd** field of the Interface Control Mode Page. If Command Forwarding is disabled, then for any command to LUN 1, the drive will return CHECK CONDITION status with a sense key of ILLEGAL REQUEST and additional sense data of LOGICAL UNIT NOT SUPPORTED (05/25/00).

When the drive receives a command for LUN 1, it will parse the CDB to determine the Transaction Class (read, write, or non-data) and the data length. It will then immediately send a Forwarded Command packet to the library. At the same time, it sets the **Lun1Cmd** flag that can be retrieved in the Drive Status Page. When the acknowledgement byte for the Forwarded Command packet is received, the **Lun1Cmd** flag is cleared, unless there is another LUN 1 command waiting.

5.5 Vendor-Specific Inquiry Vital Product Data Pages

One vendor-specific inquiry page, the Drive Status Page, was designed to support the Library Interface and is described here, in addition to being described in [LTO-SCSI].

5.5.1 Drive Status Page

The Drive Status Page, shown in Table 5, provides a snapshot of the current state of the drive and cartridge. It provides an extract of selected mode and log pages. The page code is DFh.

Table 5. Drive Status Page

Byte \ Bit	7	6	5	4	3	2	1	0
0	Peripheral Qualifier (000b)			Peripheral Device Type (01h)				
1	Page Code (DFh)							
2	Reserved (0)							
3	Page Length (3Ch)							
4	Drive State (see Table 6)							

5	CmdFwd		Alerts	Reserved	NoRemov	Unit Rsvd	Reserved	Clean
6	Reserved (0)			Threaded	Lun1Cmd	Autoload Mode (see Table 10)		
7	Reserved (0)							
8	Cartridge Type (see Table 11)							
9	Cartridge Format (0)							
10	(MSB)			Cartridge Capacity				
11								(LSB)
12	Port A Transport Type (see Table 14)							
13	(MSB)			Port A Selection ID				
14								
15								(LSB)
16	Port B Transport Type (see Table 14)							
17	(MSB)			Port B Selection ID				
18								
19								(LSB)
20	(MSB)			Operating Hours Since Manufacture				
21								
22								
23								(LSB)
24	(MSB)			Initiator ID				
...								
31								(LSB)
32	(MSB)			Cartridge Serial Number				
...								
63								(LSB)

Drive State has the values shown in Table 6. The Medium Auxiliary Memory can be accessed only in the states indicated.

Table 6. Drive State Values

Value	Description	MAM Accessible
0	DRIVE EMPTY NOT READY – No cartridge in drive, but no commands may be issued or cartridge inserted.	N
1	DRIVE EMPTY READY – No cartridge in drive. Commands will be accepted and a cartridge may be inserted.	N
2	DRIVE MEDIA LOADABLE – Cartridge is in carrier and loading may be initiated by issuing a SCSI Load or a library LOAD CARTRIDGE command.	N
3	DRIVE LOADING – Drive is loading and threading the cartridge.	N
4	DRIVE LOADED HOLD – Drive is loaded to Hold point.	Y
5	DRIVE LOADED READY – Drive can accept non-status commands.	Y
6	DRIVE WRITING – Drive cannot accept non-status commands.	Y
7	DRIVE READING – Drive cannot accept non-status commands.	Y
8	DRIVE BUSY – Drive cannot accept non-status commands.	Y
9	DRIVE UNLOADING – Tape is being unthreaded and ejected.	Y
10	DRIVE MEDIA REMOVABLE – Cartridge has been ejected and is ready for extraction by the library.	N
11	DRIVE LOAD FAILED EJECTED – Loading failed and the cartridge was returned to the loadable/removable position.	N
12	DRIVE LOAD FAILED JAMMED – Loading failed.	N
13	DRIVE LOAD FAILED HOLD – Loading failed and the cartridge is in the drive at the Hold position.	Y

14	DRIVE EJECT FAILED JAMMED – Ejection failed.	N
15	DRIVE EJECT FAILED HOLD – Ejection failed and the cartridge is in the drive at the Hold position.	Y
16	DRIVE STATE UNKNOWN.	N
17	DRIVE CLEANING FAILED	Y
18 – 255	Reserved	N/A

The following figure shows the various states of the drive for various positions of the cartridge. Actions or commands, that cause changes in the state, are shown, along with the state reported during the transition.

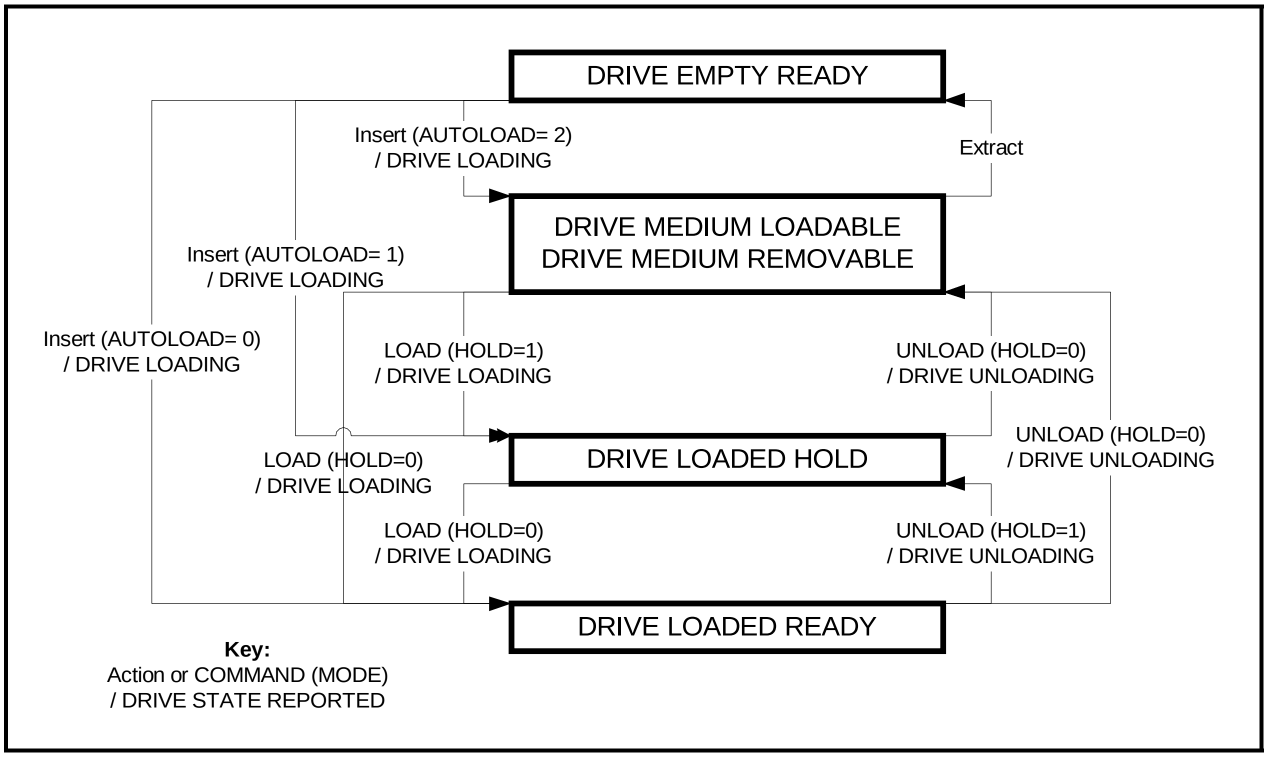

DRIVE EMPTY READY
Insert (AUTOLOAD= 2) / DRIVE LOADING
Extract
DRIVE MEDIUM LOADABLE DRIVE MEDIUM REMOVABLE
Insert (AUTOLOAD= 1) / DRIVE LOADING
Insert (AUTOLOAD= 0) / DRIVE LOADING
LOAD (HOLD=1) / DRIVE LOADING
UNLOAD (HOLD=0) / DRIVE UNLOADING
UNLOAD (HOLD=0) / DRIVE UNLOADING
DRIVE LOADED HOLD
LOAD (HOLD=0) / DRIVE LOADING
LOAD (HOLD=0) / DRIVE LOADING
UNLOAD (HOLD=1) / DRIVE UNLOADING
DRIVE LOADED READY
Key:
Action or COMMAND (MODE)
/ DRIVE STATE REPORTED

Figure 2. Load/Unload Cycle

Table 7. Drive Status Values for Cleaning Tape Insertion

Drive Loaded Hold	0x4
Drive Busy	0x8
Drive Loaded Hold	0x4

Table 8. Drive Status Values for Firmware Tape Insertion, AutoUnload = 0 or 1

Drive Loaded and Hold	0x4
Drive Loading	0x3
Drive Ready	0x5
Drive Reading	0x7
Drive Loaded and Hold	0x4
Drive Unloading Operator pressed Eject button	0x9

Table 9. Drive Status Values for Firmware Tape Insertion, AutoUnload = 2

Drive Loaded and Hold	0x4
Drive Loading	0x3
Drive Ready	0x5
Drive Reading	0x7
Drive Media Removable	0xA

Clean equal to one means that cleaning is needed.

Unit Rsvd equal to one means that an initiator has reserved the device, and the **Initiator Selection Address** field will contain the three-byte Selection Address of the initiator holding the reservation. If **Unit Rsvd** is zero, the **Initiator Selection Address** field is invalid.

NoRemov equal to one means that removal of the cartridge has been disabled via the Prevent/Allow Medium Removal command. **NoRemov** equal to zero means that removal is enabled.

Alerts equal to one indicates that unsolicited transmission of alert packets to the Library is enabled.

CmdFwd equal to one indicates that Command Forwarding is enabled. A value of zero indicates that it is disabled, one that it is enabled. Other values are reserved. This is controlled by the vendor-specific Interface Control Mode Page.

Autoload Mode specifies how the cartridge is handled when it is inserted. It contains a value specified by the AUTOLOAD MODE field of the Control mode page, as specified in SPC-2. For reference, the values are:

Table 10. Autoload Mode Values

Value	Meaning
000h	Loaded until medium is ready
001h	Loaded until medium auxiliary memory is accessible.
010h	Not loaded until a Load command is issued.
Others	Reserved

Lun1Cmd indicates that the drive has received a command over the primary interface which is addressed to LUN 1. This field will contain a value of one until the Forwarded Command packet has been transmitted to the library and the packet acknowledgement byte received. If multiple commands have been received – such as untagged commands from multiple initiators – the field will remain one until all Forwarded Command packets have been transmitted. If **CmdFwd** is zero, then **Lun1Cmd** will always be zero, as commands to LUN 1 are rejected by the drive.

Threaded is one when the tape is threaded, although the tape may still be winding and not yet ready for a medium access command.

Cartridge Type is shown in Table 11:

Table 11. Cartridge Types

Value	Generic Meaning	LTO Drives
0	Empty – no cartridge is present	Empty – no cartridge is present

1	Cleaning cartridge	Cleaning cartridge
2	Unknown data cartridge	Unknown data cartridge
3	Firmware cartridge	Firmware cartridge
4	Data cartridge	Ultrium Type A data cartridge
5	Data cartridge	Ultrium Type B data cartridge
6	Data cartridge	Ultrium Type C data cartridge
7	Data cartridge	Ultrium Type D data cartridge

Cartridge Format identifies the tape format of this cartridge. As there is only one Ultrium format defined at this time, this value is zero. During insertion, the type of a data cartridge is unknown between the time the cartridge is seated and the drive completes winding to BOT; the length calculation takes place during winding.

Cartridge Capacity is the uncompressed capacity in gigabytes.

Port A/B Transport Type and **Port A/B Selection ID** are the current drive addresses on each port, as specified in Table 14. For Parallel SCSI, the Port B fields are zeroes.

Operating Hours Since Manufacture is the number of hours of head-tape contact time.

Initiator ID specifies the initiator holding a reservation on the drive, when the **Unit Rsvd** field is one. If the interface is Parallel SCSI, then the least-significant byte contains the initiator's SCSI ID; if the interface is Fibre Channel, then the field contains the initiator's 64-bit worldwide ID. If **Unit Rsvd** is zero, then this field is zero.

Cartridge Serial Number is as specified by attribute number 0201h of the SPC-2 Read Auxiliary Memory command. It is a 32-byte ASCII string, right padded with blanks.

5.6 Vendor-Specific Mode Pages

There is one vendor-specific mode page that is used to control the primary and library interfaces. It is described here, in addition to being described in [LTO-SCSI].

5.6.1 Interface Control Mode Page

The Interface Control Mode Page, shown in Table 12, is used to read and set various parameters that the drive will have after the next device reset or bus reset. The Page Code field is 22h.

Table 12. Interface Control Mode Page

Bit / **Byte**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	PS	Reserved	Page Code (22h)					
1	Page Length (0Ch)							
2	Baud Rate							
3	Reserved (0)				CmdFwd		2StopBits	Alerts
4	Port A Transport Type							
5	(MSB)							
6			Port A Selection ID					
7								(LSB)
8	Port B Transport Type							
9	(MSB)							
10			Port B Selection ID					

11		(LSB)
12	Next Selection ID	
13	Jumpered Selection ID	
14	Target Initiated Bus Control	
15	Reserved (0)	DisableWideBusMode

Baud Rate is a changeable field specifying the baud rate at which the Library interface will operate after the next reset. The values are consecutive integers, which map to baud rates as shown in Table 13. If the field is inadvertently set to 0 in the NVRAM, then the interface will operate at 9600 baud and this page will report a value of 2.

Table 13. Baud Rate Field Values

Baud Rate Field Value	Baud Rate
0	Reserved (9600)
1	4800
2	9600
3	19200
4	38400
5	57600
6	115200
7 – 255	Reserved

Alerts is a changeable field; when set to one, it enables unsolicited transmission of alert packets to the Library.

Note: Alerts are not currently implemented in the Viper drive.

2StopBits is a changeable field; when set to one, it causes the interface to send two stop bits for each byte; set to zero, it sends one stop bit.

CmdFwd is a changeable field; when set to one, enables command forwarding; a value of zero disables it. If the value is zero and it is not changeable, then command forwarding is not supported. Other values are reserved. See 5.4.

Port A/B Transport Type and **Port A/B Selection ID** are non-changeable fields; their relationship is shown in Table 14. For a Parallel SCSI-attached drive, the **Port A Selection ID** field contains the SCSI ID and the Port B fields are not meaningful and contain zeroes.

For a Fibre Channel drive, the selection ID fields contain the 24-bit Fibre Channel port address; the format shown is for a public loop device. If the link goes down then the Transport Type will be set to 0 and will return to 2 after the link is restored. In the data for a MODE SELECT command, these fields are not checked, as their values can change without the application client's knowledge.

Table 14. Transport Type Fields and Present Selection ID Fields

Transport Type	Meaning	MSB	Middle Byte	LSB
0	Invalid	0	0	0
1	SCSI Ultra 2	0	0	SCSI ID
2	FC 1 Gbaud	Domain	Area	AL-PA
3 – 255	Reserved	N/A	N/A	N/A

Next Selection ID is a changeable field specifying the selection ID which the drive will attempt to assume the next time it is reset. This functionality is provided to obviate the need for changing address jumpers. For Parallel SCSI, it is a SCSI ID; for Fibre Channel, it is the Assigned Loop Identifier to be requested in the LIHA phase of the next LIP.

In Parallel SCSI drives, this field is set to the current ID after reset. In Fibre Channel drives, this field is **not** changed if the drive assumes a different ID; the drive will continue to try to assume this ID in every LIP. The field only changes when either the external jumpers or the **Next Selection ID** field changes.

Jumpered Selection ID is a non-changeable field for reading the selection ID specified by external jumpers or by the SFF-8067 SEL_n signals, if supported. For Parallel SCSI, it is a SCSI ID; for Fibre Channel, it is the Assigned Loop Identifier specified in Table K.1 of [FC-AL-2].

If both the **Next Selection ID** field and the external jumpers are changed simultaneously, the selection ID specified by the external jumpers will be used.

Target Initiated Bus Control is a changeable field controlling target initiated operations in Parallel SCSI drives.

Table 15. Target Initiated Bus Control Modes

Value	Meaning
0	No target initiated modes (default)
1	Enable target initiated SDTR.
2	Enable target initiated WDTR.
3	Enable both target initiated SDTR and WDTR.

Disable Wide Bus Mode is a changeable field determining bus width in Parallel SCSI drives.

5.7 Library Application Layer Guidelines

Because the library is required to act as a SCSI initiator, it must implement the SCSI exception reporting mechanism. This includes recognizing and responding to the SCSI Status returned after the completion of each command and using the REQUEST SENSE command to determine the nature of exceptions.

5.7.1 General

The library must examine the Service Response and SCSI Status returned for every command. If the Service Response is not Command Complete (00h), then the Command packet was probably erroneous. If the Service Response is Command Complete but the SCSI Status is not Good Status (00h), then the library must take appropriate action. If the SCSI Status is Check Condition (02h), then the library must issue a REQUEST SENSE and take action depending upon the Sense Key (SK) and additional sense data (ASC/ASCQ) reported.

A Sense Key of Unit Attention (06h) is informative and can usually be ignored during early development of library functionality. When the library’s error detection becomes more sophisticated, it should check the ASC/ASCQ. For example, these will inform the library that a cartridge has been inserted, the loading or unloading process has reached the point where the cartridge memory can be read, that the drive is ready, or that the cartridge is completely ejected.

5.7.2 Powerup

The library should implement the following behavior at power up:

- Issue TEST UNIT READY and REQUEST SENSE and to clear the expected SK/ASC/ASCQ of Unit Attention / Power On, Reset, or Bus Device Reset Occurred.
- Issue TEST UNIT READY and REQUEST SENSE, which should show an ASC/ASCQ of Medium Not Present. If the drive powered up with a cartridge inside, then TEST UNIT READY will eventually return Good Status, unless a problem occurs.
- Issue MODE SENSE and MODE SELECT to check the drive's operating parameters and to change them if necessary. Note that changing some parameters, like the SCSI ID and baud rate require issuing a Target Reset task management request. That reset will cause a new Unit Attention (06/29/00).

5.7.3 Operations

When the drive is performing an operation, the library should poll for status by issuing INQUIRY for the Drive Status VPD page (DFh). The Drive State field will indicate what the drive is doing. When the drive reaches the expected stable state (e.g., DRIVE LOADED READY on a load or DRIVE LOADED HOLD or DRIVE MEDIA REMOVABLE on an unload, then the library should send a series of TEST UNIT READY/REQUEST SENSE command pairs to clear any Unit Attentions).

If the Drive Status field shows an error, then the drive should issue TEST UNIT READY/REQUEST SENSE command pairs to clear Unit Attentions and to get sense data containing a description of the exception.

To detect possible unrecoverable errors, the Library should also implement a high-level timer in case the drive continues to report an intermediate state without finishing the operation.

6. Transaction Layer

The transaction layer interfaces between the application layer and the link layer. Its primary purposes are to handle transmission errors reported by the link layer, detect higher-level errors peculiar to serial interfaces, and to present a SCSI-3 view of the lower layers to the application layer.

At this level, the library interface protocol is derived from the Fibre Channel Protocol for SCSI (FCP), because FCP deals with all of the issues which arise in performing SCSI commands over a serial interface, with its implicit disconnections between different information types. Furthermore, each information type is explicitly acknowledged, as in Fibre Channel Class 2 connections.

One simplification is that error recovery is less complex than in FCP. Because the serial interface is not used for data transfers that must maintain high throughput, it is acceptable to terminate a command in which an error has occurred and then retry it, rather than follow the FCP approach of performing low-level error recovery in an attempt to avoid retransmission of large blocks of data. Finally, the presence of only two nodes on a link and the lower throughput requirement mean that there is no need for a complex arbitration process; the nodes simply begin sending and retry in the event of a collision, in a manner analogous to Ethernet’s CSMA/CD protocol.

6.1 Classes of Transactions

Execution of a SCSI command or sending a drive alert requires a **transaction** of one or more **packets**, as shown in Figure 3. The node that starts the transaction is called the **requester** and the other node is called the **responder**. (For execution of a SCSI command, the library is the requester and the drive is the responder. For execution of an alert, the drive is the requester.) In each diagram of the figure, the packet is shown by an arrow. The direction of the arrow indicates which node is sender and which is receiver for that packet. Only the packet transmission is shown in this figure; the reply from the receiver is not shown, as that is part of the (lower-level) link protocol.

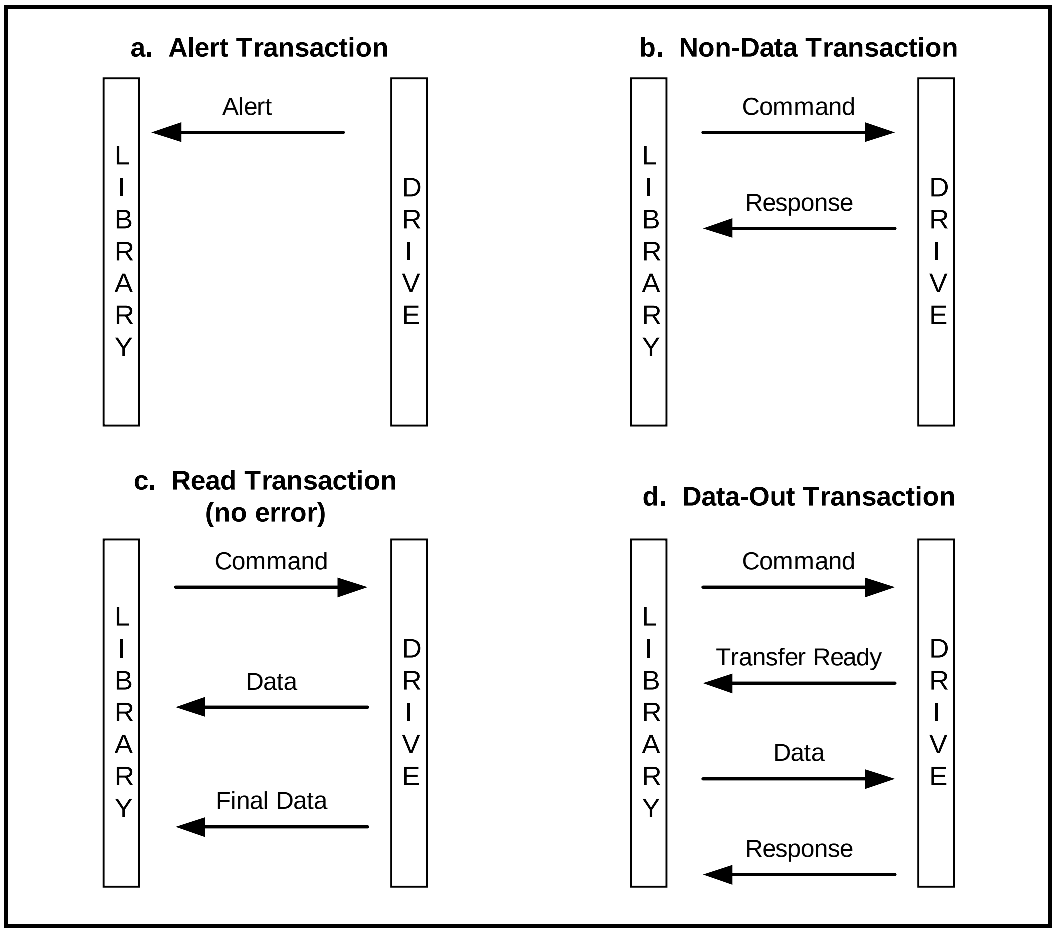


Figure 3. Transaction Protocol

6.1.1 Alert Transaction

The simplest case is when no data is transferred, such as when the drive sends an alert to the library. In this case, shown in **diagram a** of Figure 3, the library does not send a response packet to the drive; it will initiate some command in response to the alert. Alert transactions are never sent by the library.

Note: Alerts are not currently implemented in the Viper drive.

6.1.2 Non-Data Transaction

In **diagram b**, the library sends a Command packet to the drive and the drive replies with a Response packet. An example would be a task management command to reset the drive.

6.1.3 Data-In Transaction

If the command is issued by the requester to obtain data from the responder, then **diagram c** applies. The requester sends a Command packet and the responder sends zero or more Data packets, followed by a Final Data-In packet. An example of this is a SCSI Mode Sense command.

Prior to issuing the command, the requester **must** allocate sufficient buffer space to receive the entire requested data payload. If the data length exceeds the tape drive's maximum packet payload size, there will be multiple Data packets; if the library has allocated the full buffer, then there will never be a buffer full response when the drive sends a Data packet.

Only if there is an error in execution of the command will the drive issue a Response packet in a data-in command. The Final Data-In packet implies command completion with good status.

6.1.4 Data-Out Transaction

If the purpose of the transaction is to send data to the responder, such as a SCSI Mode Select, then four packets are necessary, as shown in **diagram d**: a Command packet to the responder, a Transfer Ready packet whereby the responder tells the requester how much data to send, one or more Data packets from the requester to the responder, and finally a Response packet from the responder to the requester.

The use of the Transfer Ready packet allows the drive to manage flow control and avoid buffer overflow; if the library were to manage flow control, it would either have to learn the tape drive's maximum transaction payload size and prepare packets accordingly or retry sending write data when the drive sends a buffer full status.

6.1.5 Multiple Data Packets

When a transaction requires multiple data transfers, the first data packet shall contain the first byte in the command's logical data buffer, i.e., the one at offset zero. Subsequent data packets shall begin with the data byte immediately following the last one of the previous data packet. In other words, the data packets shall be contiguous and shall have increasing relative offset in the logical buffer. This restriction makes it unnecessary to have an explicit relative offset field in the transfer ready and data packets.

6.1.6 Command Forwarding Transactions

Because Command Forwarding transactions use a different type of command packet, they are shown separately, in the following figure. Note also that the Transfer Ready packet is used for Data-In transactions to prevent overflowing the drive's data buffer.

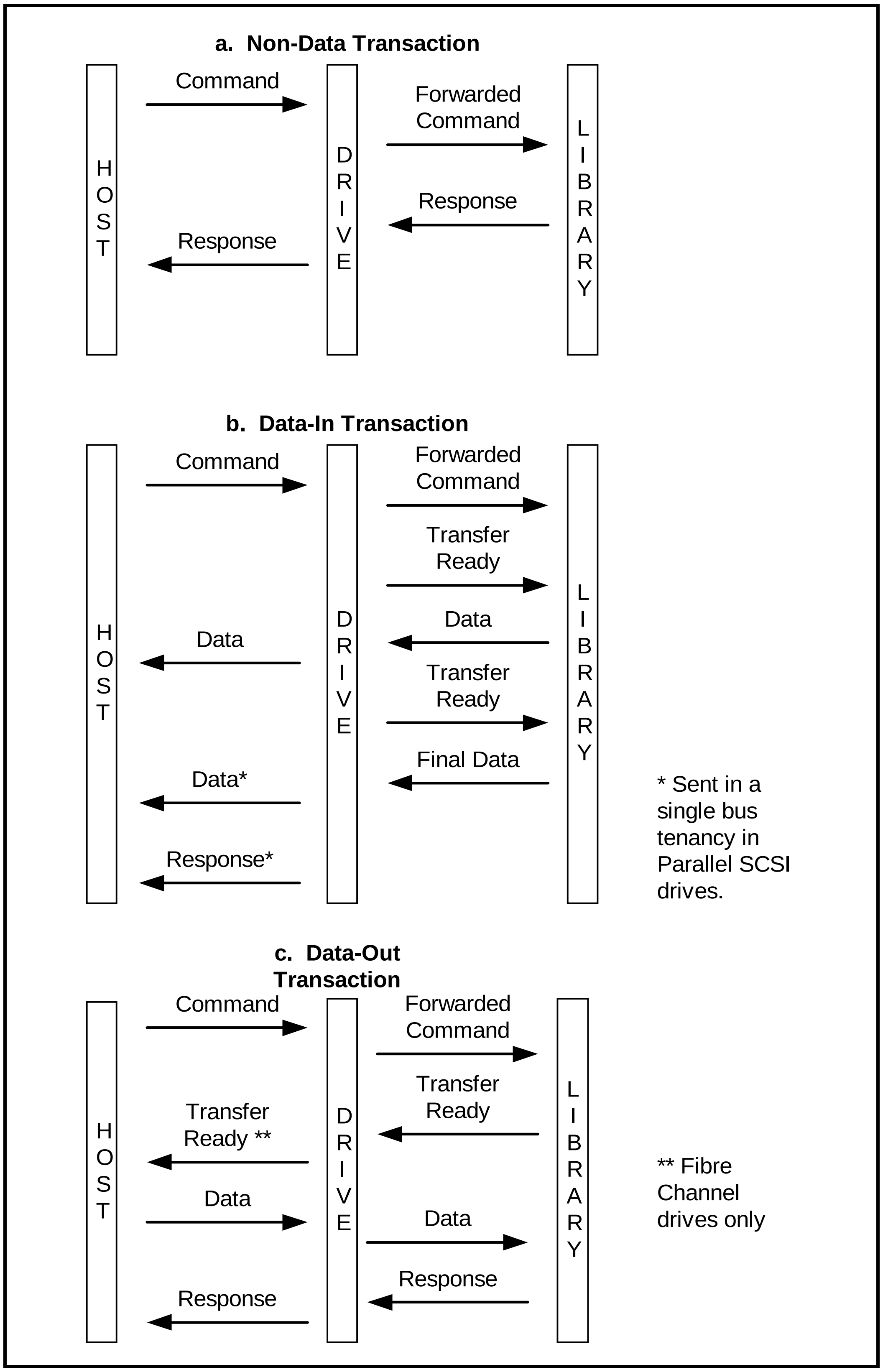


Figure 4. Command Forwarding Transactions

6.1.6.1 Read Data Flow Control

To avoid overrunning the drive's buffers when read data is being transferred to an initiator, the drive must send a Transfer Ready packet before the Library can send a Data packet. The amount of data in the Data packet must not exceed the Data Length specified by the Transfer Ready. The drive will not send a Transfer Ready after receipt of a Final Data-In packet.

(When the Encapsulated SCSI Protocol is used by the *library* to issue a data-in command *to* the drive, the Transfer Ready is not used, as the library is presumed to have allocated a large enough buffer to handle the requested data. This is not workable in command forwarding, because the data buffer in the initiator may exceed that in the drive.)

6.1.6.2 Autosense Data

Autosense data is transmitted on the Library interface in Response packets only in drives whose primary interface is Fibre Channel. Parallel SCSI drives do not allow autosense data in the Library interface to make the functioning of the two transports as similar – and the implementation as simple – as possible.

6.1.6.3 Link Failures

If the transfer of command, data, or status between the library and the drive fails, the drive will issue an Abort packet to the library for that transaction ID. It will then terminate the command with the initiator by returning a status of CHECK CONDITION and setting additional sense data of ABORTED COMMAND / LOGICAL UNIT COMMUNICATION FAILURE, 0B/08/00. If multiple forwarded commands are active, then only the one command will be affected.

6.1.6.4 Target Reset

When a Target Reset task management command or SCSI Bus Reset is received, both the drive and library must be reset. In Parallel SCSI drives, the drive will go bus free upon receipt of the Bus Device Reset. The drive will transmit a Target Reset task management command to the library before resetting itself.

6.1.6.5 Logical Unit Reset (LUN 1)

A Logical Unit Reset task management command to LUN 1 is transmitted to the library via the command forwarding mechanism. The drive will discard the state of any active LUN 1 commands, and the library will send status to the drive only for the LU Reset. It will then be forwarded to the initiator. If a race condition results in the drive's receiving status on an aborted command, it will be discarded.

6.2 Interface to Link Layer

The transaction layer's view of packet transmission and reception can be explained as a set of three functions:

- **Packet_Request()** is implemented by the link layer and is invoked by the transaction layer to send a packet that it has prepared.

- **Packet_Confirmation()** is implemented by the transaction layer and is invoked by the link layer to report the success or failure of an attempt to send a packet. Every invocation of **Packet_Request()** must result in an invocation of **Packet_Confirmation()**.
- **Packet_Indication()** is implemented by the transaction layer and is invoked by the link layer to report the successful arrival of a packet. See section 6.5 for errors that prevent invocation of **Packet_Indication()**.

6.3 Inter-Packet Rules

Packet transmissions must be atomic, i.e., one packet must be sent and acknowledged before the next can be sent. This is implemented by the link and transaction layers:

- The transaction layer may not submit a **Packet_Request()** until it has received a **Packet_Confirmation()** for its previous request.
- If the link layer is receiving a packet when a **Packet_Request()** is submitted, transmission of the new packet is held until reception of the incoming one is complete (or timed out as an underrun) and acknowledged.
- Two packets may not be transmitted in opposite directions at the same time; i.e., one node must acknowledge reception of a packet before it can transmit a new one.

This atomicity restriction reduces the complexity of the link layer, as there should never be any ambiguity as to which packet is being acknowledged. As the library serial interface is not required to support high data rates, the lack of overlap between packets in opposite directions is not a significant limitation.

Examples:

- In a non-data transaction, the responder must send the acknowledgement for the Command packet before sending the Response packet.
- In a data-in transaction, the responder must acknowledge the Command packet before sending the Data packet. If multiple read Data packets are used, the responder must wait for the requester's acknowledgement of each one before sending the next. And the responder must receive an acknowledgement for the next-to-last Data packet before sending the Final Data-In packet.
- In a data-out transaction, the responder must send the command acknowledgement before sending the first Transfer Ready, and the requester must acknowledge each Transfer Ready before sending the corresponding Data packet. The responder must then acknowledge each write Data packet before sending the next Transfer Ready packet or the Response packet.

6.4 Inter-Transaction Rules

Nodes may intermix packets of different transactions. In particular, if the drive recognizes an error condition while a transaction is in process, it may send an alert (which will be in a new transaction) without completing the first transaction. Similarly, the library need not wait for completion of any open transactions if it needs to send a Reset Target task management command (which will also be in a new transaction).

6.5 Error Detection and Handling

The transaction layer has the responsibility for handling certain types of errors. These fall into three classes:

- Arbitration errors, in which both nodes begin transmitting simultaneously (a collision) or in which the receiving node does not have an available buffer. As both of these conditions should be transitory, the link layer will attempt to re-send the packet. This is the only case of retransmission in the protocol; other errors are fatal to the transaction.
- Link errors may be detected by the receiver's link layer (erroneous Header, Length, or Checksum field, no receive buffer available, or failure to receive the number of bytes specified within a fixed time) and are reported to the sender as a transmission failure. The receiver's transaction layer is not informed of the failed reception. Loss of the acknowledgement from the receiver is detected by the sender's link layer and is reported to the truncation layer as an error. A link error is fatal for the transaction.
- Payload errors, in which the receiver's transaction layer detects a problem with an incoming packet's payload, such as a write data packet whose data length does not match the corresponding transfer ready's data length. In such cases, the encapsulation fields were all correct, and the link layer returned an acknowledgement to the sender and passed the packet to the transaction layer. The transaction is not performed, e.g., data is discarded. Payload errors are also fatal for the transaction.

If a fatal error occurs, the requester may then elect to retry the transaction; that is beyond the scope of this specification. The details of how this is handled can depend upon whether the transaction is an alert, a SCSI command, or a task management command, whether the sender or receiver detects the error, and whether the packet is part of an existing transaction.

This specification defines error handling to the extent necessary to ensure that both nodes understand that there has been an error and that the transaction is terminated. If there is a payload error in a transaction that is either a task management or SCSI command, the responder must return a response packet indicating failure. If it is the responder who detects an error in an on-going transaction, it will terminate processing of the command and send back an appropriate response packet (defined below). If it is the requester who detects the error in an on-going transaction, it must send an Abort Transaction packet to the responder, who will then terminate processing and send a response packet. If the requester detects the error in the initial packet (commands from the library, forwarded commands from the drive, or alerts from the drive), it will report failure to its application layer.

Table 16. Transaction Layer Error Handling

Error	Detector's Action	Other Node's Action
Requester detects packet transmission failure (i.e., gets bad acknowledgement or times out) in the packet of a SCSI Command, task management, or alert transaction.	Reports failure to application layer, which may retry. Resources are deallocated.	None. No resources remain allocated for the transaction.
Requester detects packet transmission failure (i.e., gets bad acknowledgement or times out) in an	Requester sends Abort Transaction.	Upon receiving Abort Transaction, Responder sends Transaction Status of Transaction Aborted and SCSI

ongoing SCSI Command transaction, or receives invalid packet. (See Note 1.)		Status of Check Condition with SK/ASC/ASCQ = 09h/45h/00h (Select or Reselect Failure).
Responder detects packet transmission failure or receives invalid packet in an ongoing SCSI Command or task management transaction. (See Note 1.)	Responder sends Transaction Status from Table 31 and SCSI Status of Check Condition with SK/ASC/ASCQ = 09h/45h/00h (Select or Reselect Failure).	Requester may re-start transaction.
Responder detects invalid packet not in an existing transaction.	Responder's link layer replies with appropriate error status (Table 17).	Requester may re-start transaction.
Task completes as an Abort Task management command is sent to abort that transaction.	Responder (drive) transaction layer finds invalid transaction ID. Sends Function Complete response packet.	Function Complete status is reported to application layer after SCSI status of completed command.

Note 1 When the transaction layer receives a packet with any of the following conditions in an ongoing transaction, it is regarded as an invalid packet:

- Write data payload length ≠ transfer ready data length
- Read data payload length plus previously-transferred read data > command packet data length
- Packet Type inappropriate at that point in the transaction

7. Link Layer

The process of the drive's executing a command from the library or of the drive's sending an alert to the library requires actions at several levels. At the highest level, the complete set of information transferred in both directions is called a transaction. A transaction might consist of a Log Sense command from the library to the drive, a log page from the drive to the library, and SCSI status from the drive to the library.

Each of these transfers within the transaction is called a **packet**. Each packet contains a distinct type of information, such as command, data, or response. Transfer of a packet requires two actions, transmission of the packet by the sender and transmission of an acknowledgement by the receiver. The receiver of the packet will validate all fields in the packet except the data and will send an appropriate acknowledgement.

This section describes the protocol at the transaction and packet levels and then presents the formats of the various packets.

7.1 Link Protocol

The link protocol is designed for reliable transmission of one packet across the interface. The node sending the packet is called the **sender** and the other node is called the **receiver**. During the course of a transaction, the requester and responder will each be a sender and a receiver, depending upon which packet is being sent. Simply, the sender will send a packet and the receiver will reply with an acknowledgement within a specified timeout.

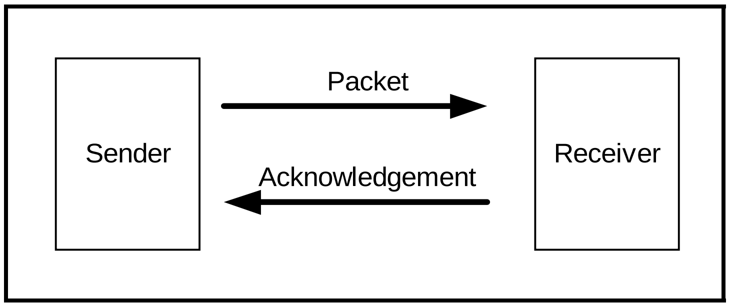


Figure 5. Link Protocol

The link protocol attempts to detect two classes of errors: loss of synchronization between sender and receiver, and data corruption. The link protocol will attempt to resynchronize and the sender will report an error to its upper layer. The receiver will not report the failed packet to its transaction layer. Recovery at the upper layer is handled as for any SCSI command that fails to connect or reconnect.

7.1.1 Special Characters

A number of special characters are recognized by the link protocol state machine. One is the header character, which is the first character in every packet. The others are acknowledgements, single-character replies from the packet receiver to the sender which indicate whether the packet transmission was successful.

Table 17 gives the meanings and values of these special characters. The values were chosen to have a Hamming distance of at least two from one another and from the common characters 00h and FFh; this avoids having one valid character changed into another valid – but incorrect – one by a single-bit error. This reduces the probability of confusion in the state machines.

Table 17. Special Characters

Character Name	Usage	Symbol	Value
Header	First character of packet	HDR	30h
No Error	Acknowledgement	S_OK	06h
Reserved (Note 2)	Not used	N/A	03h
Bad Type Field	Acknowledgement	S_TY	05h
Bad Checksum	Acknowledgement	S_CK	09h
Underrun (receiver timeout)	Acknowledgement	S_UR	0Ch
Buffer Full	Acknowledgement	S_BF	0Fh
Reserved	Not used	N/A	00h
Reserved	Not used	N/A	FFh

Note 2 The value 03h was formerly defined as a special character and is now reserved to avoid compatibility problems.

If any character other than HDR appears as the first character of a packet, it is an error. If any character other than S_* appears as an acknowledgement, it is an error.

7.1.2 State Machines

In the following sender (Figure 6), receiver (Figure 7), and resynchronization (Figure 8) state machines, the heavy vertical lines are states, and the horizontal arrows are transitions. The label above the arrow indicates the signal that triggers the transition, and the label below the arrow indicates the actions taken. Only the signals shown causing transitions from a state are valid. Any other signals received while in that state are ignored. The signals and actions are listed in

Table 16 Table 16. These are actually three proper subsets of the link state machine, rather than three separate state machines.

Table 18. State Machine Signals, Actions, and Flags

Signal, Action, or Flag	Type	Meaning
Pkt_Request( )	Signal	A request from the transaction layer that initiates a packet transmission by the link layer's sender state machine.
Pkt_Indication( )	Action	A signal from the receiver state machine to the transaction layer to indicate a newly-received transaction.
Pkt_Confirmation(status)	Action	A signal from the link layer sender state machine to the transaction layer to indicate the successful or unsuccessful completion of a Pkt_Request( ). Statuses are the Acknowledgement (S_*) special characters: No Error (S_OK) , Bad Type (S_TY), Bad Checksum (S_CK), Underrun (S_UR), and Buffer Full (S_BF).
Tmr_Start(*_TOV)	Action	The state machine starts the timer using the specified timeout value.
Tmr_Stop( )	Action	The state machine stops the running timer prior to its expiration.
Tmr_Timeout( )	Signal	Expiration of a timer; stopping the timer is then not necessary.
Byte_Request(n)	Action	Request from state machine for physical layer (hardware) to send the specified character. 'x' indicates an unspecified character.

Byte_Confirmation( )	Signal	Physical layer reports completion of the previous Byte_Request( ).
Byte_Indication( )	Signal	Physical layer reports reception of a character.
Last / Not Last	Flag	Byte sent was / was not the last byte of that packet.
Pending	Flag	TRUE value indicates that a packet is waiting to start or is in transmission.

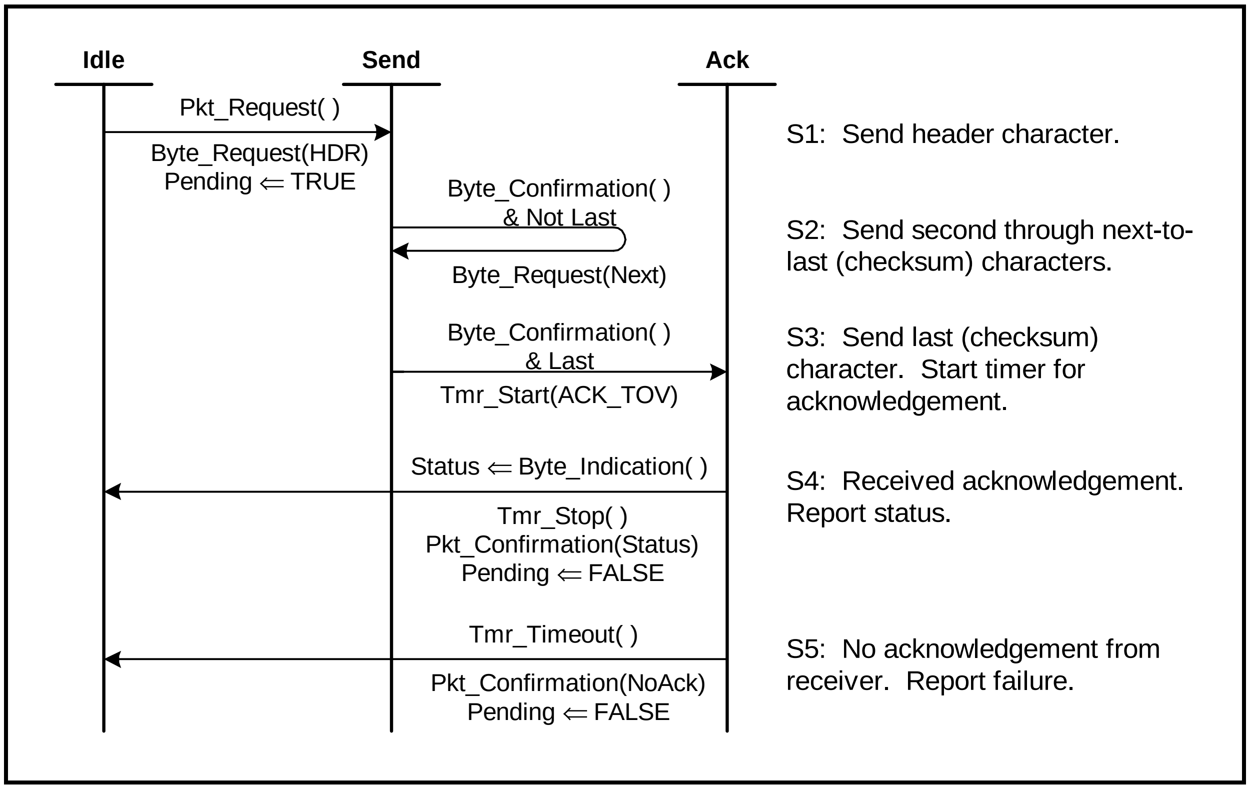
There is a single timer for use by the state machine via the Tmr_*( ) actions and signal. Table 20 summarizes the timeout values that may be set.

7.1.2.1 Sender

The following table describes the transmission of a complete packet in terms of state transitions in the sender and receiver machines. In this example, all of the required bytes are sent and received, although there may be errors internal to the packet.

Table 19. Complete Packet Transmission Scenario

Character Sent	Transmitting Node	Sender Transition	Receiver Transition
Header character (# 0)	Sender	S1	R1
Characters (# 1..n-1)	Sender	S2	R2
Last character (# n: checksum)	Sender	S3	R3
Acknowledgement (no error)	Receiver	S4	R7
Acknowledgement (error)	Receiver	S4	R8



Idle
Send
Ack
Pkt_Request( )
Byte_Request(HDR)
Pending ⇐ TRUE
Byte_Confirmation( )
& Not Last
Byte_Request(Next)
Byte_Confirmation( )
& Last
Tmr_Start(ACK_TOV)
Status ⇐ Byte_Indication( )
Tmr_Stop( )
Pkt_Confirmation(Status)
Pending ⇐ FALSE
Tmr_Timeout( )
Pkt_Confirmation(NoAck)
Pending ⇐ FALSE
S1: Send header character.
S2: Send second through next-to-last (checksum) characters.
S3: Send last (checksum) character. Start timer for acknowledgement.
S4: Received acknowledgement. Report status.
S5: No acknowledgement from receiver. Report failure.

Figure 6. Sender State Machine

7.1.2.2 Receiver

The receiver state machine (Figure 7 below) contains additional transitions to handle a Packet_Request( ) that may arrive during reception. Transition R5 or R6 is taken in this case and the Pending flag is set. When transmission of the acknowledgement is confirmed, either R9 or R10 will be taken to send the header character of the newly-requested packet. The former is taken if there was no error in reception, the latter if there was an error.

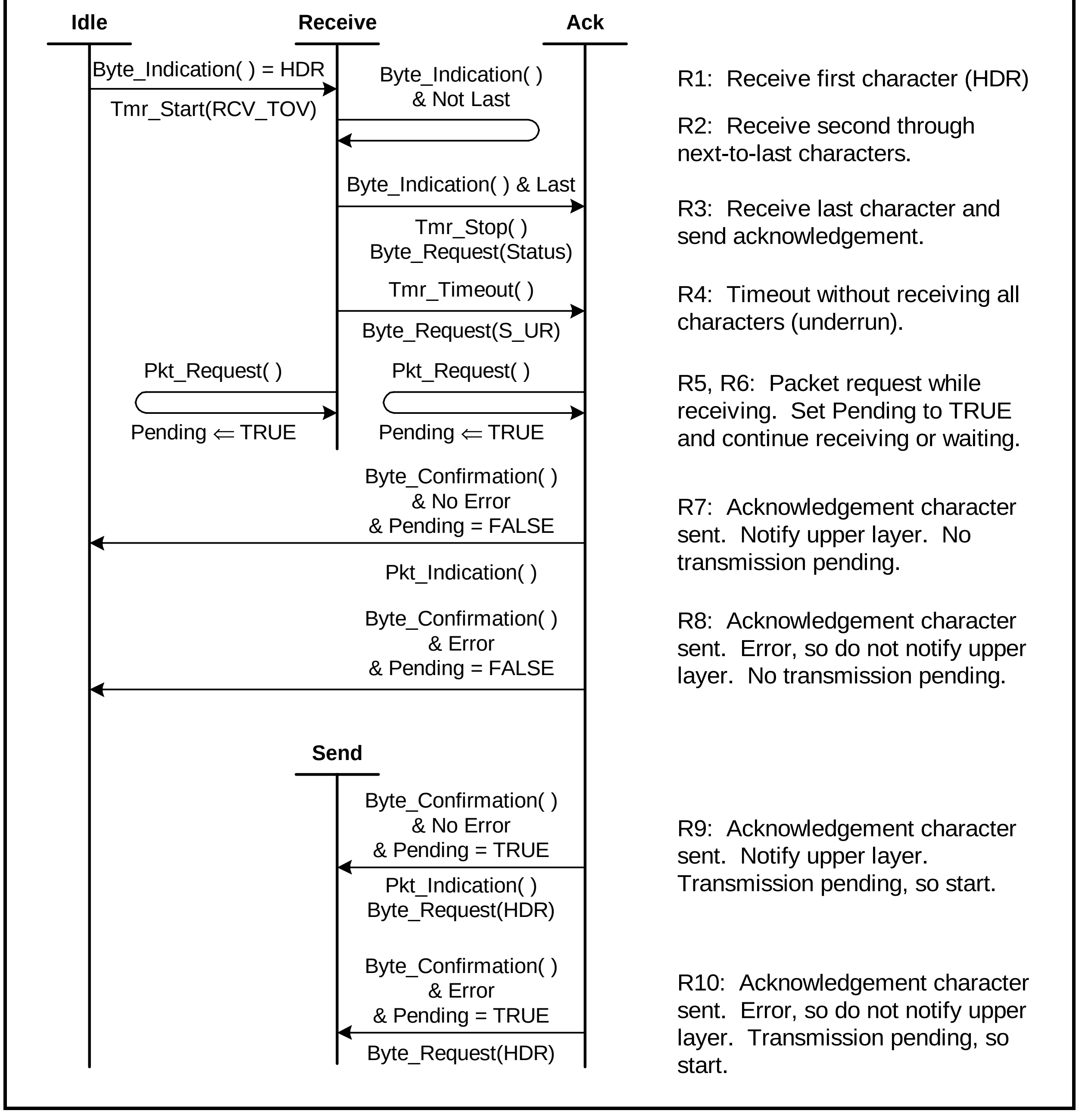

Idle
Receive
Ack
Byte_Indication( ) = HDR
Tmr_Start(RCV_TOV)
Byte_Indication( ) & Not Last
Byte_Indication( ) & Last
Tmr_Stop( ) Byte_Request(Status)
Tmr_Timeout( )
Byte_Request(S_UR)
Pkt_Request( )
Pending ⇐ TRUE
Pkt_Request( )
Pending ⇐ TRUE
Byte_Confirmation( ) & No Error & Pending = FALSE
Pkt_Indication( )
Byte_Confirmation( ) & Error & Pending = FALSE
Send
Byte_Confirmation( ) & No Error & Pending = TRUE
Pkt_Indication( ) Byte_Request(HDR)
Byte_Confirmation( ) & Error & Pending = TRUE
Byte_Request(HDR)
R1: Receive first character (HDR)
R2: Receive second through next-to-last characters.
R3: Receive last character and send acknowledgement.
R4: Timeout without receiving all characters (underrun).
R5, R6: Packet request while receiving. Set Pending to TRUE and continue receiving or waiting.
R7: Acknowledgement character sent. Notify upper layer. No transmission pending.
R8: Acknowledgement character sent. Error, so do not notify upper layer. No transmission pending.
R9: Acknowledgement character sent. Notify upper layer. Transmission pending, so start.
R10: Acknowledgement character sent. Error, so do not notify upper layer. Transmission pending, so start.

Figure 7. Receiver State Machine

7.1.2.3 Resynchronization

Resynchronization occurs when the receiver and sender state machines may have lost synchronization:

1. The first character received is not the header character, possibly indicating that reception began in the middle of a packet. (E1)
2. The sender received a character before it sent the last byte of the packet, possibly indicating simultaneous sending by both nodes (a collision). (E2)

The general philosophy of resynchronization is for the node or nodes detecting either of the above conditions to discard all characters received during a period slightly longer than the time to transmit a maximum-length packet (IDLE_TOV). (E3)

At the end of this time, the node returns to the Idle state (E5). If there is a packet to send (E6), then the node will wait in Idle for the retransmission time (RETRANS_TOV). When the timer expires, it will begin sending the packet (E7). As the Library and the Drive use different values of RETRANS_TOV, in the event of a collision they should not begin retransmission at the same time. This is analogous to the random backoff used in Ethernet, but is simplified by the existence of only two nodes of distinct types.

Table 20. Timeout Value Definitions

Name	Abbreviation	Value	Meaning
Packet Transmission	RCV_TOV	Time for maximum length packet plus 100 bytes (2154 bytes)	Time that receiver waits for the last byte of a packet after receiving the first byte of the packet. If the complete packet is not received in this time, the Receiver sends S_UR acknowledgement.
Recovery Idle	IDLE_TOV	RCV_TOV + 100 msec	Time that sender ignores received characters after detecting a character from the receiver while packet transmission is in progress.
Acknowledgement Reception	ACK_TOV	IDLE_TOV + 100 msec	Time that Sender waits for acknowledgement from receiver after sending last byte of packet. If an acknowledgement is not received in this time, the Sender initiates recovery.
Retransmission	RETRANS_TOV_DRV	ACK_TOV + 100 msec	Time that Drive waits for incoming bytes, before attempting to re-send a packet that experienced a collision. Library and drive use different values to avoid a new collision.
	RETRANS_TOV_LIB	ACK_TOV + 200 msec	Time that Library waits for incoming bytes, before attempting to re-send a packet that experienced a collision. Library and drive use different values to avoid a new collision.

To satisfy the requirements, the values must satisfy the following conditions:

1. *IDLE_TOV > RCV_TOV*: This ensures that all of the packet's bytes will be sent before the receiver begins looking at incoming bytes again. RCV_TOV is the longest time allowed for all of the bytes of a packet to be transmitted.
2. *IDLE_TOV < Minimum Packet Time + ACK_TOV* and
   IDLE_TOV + RETRANS_TOV > Maximum Packet Time + ACK_TOV: These ensure that the

sender times out its wait for an acknowledgement (ACK_TOV) during the other node's retransmission time, so that if the sender begins sending immediately, it will probably not collide with a new transmission by the other node. This implies that:
ACK_TOV < RETRANS_TOV.

The following table summarizes the values in milliseconds of these timeouts for the supported baud rates. The values are rounded to the next higher multiple of 10 msec.

Table 21. Numerical Timeout Values in Milliseconds

Baud Rate / **Value**	**4,800**	**9,600**	**19,200**	**38,400**	**57,600**	**112,500**
RCV_TOV	4,490	2,250	1,130	570	380	200
IDLE_TOV	4,590	2,350	1,230	670	480	300
ACK_TOV	4,690	2,450	1,330	760	580	400
RETRANS_TOV_DRV	4,790	2,550	1,430	860	680	500
RETRANS_TOV_LIB	4,890	2,650	1,530	960	780	600

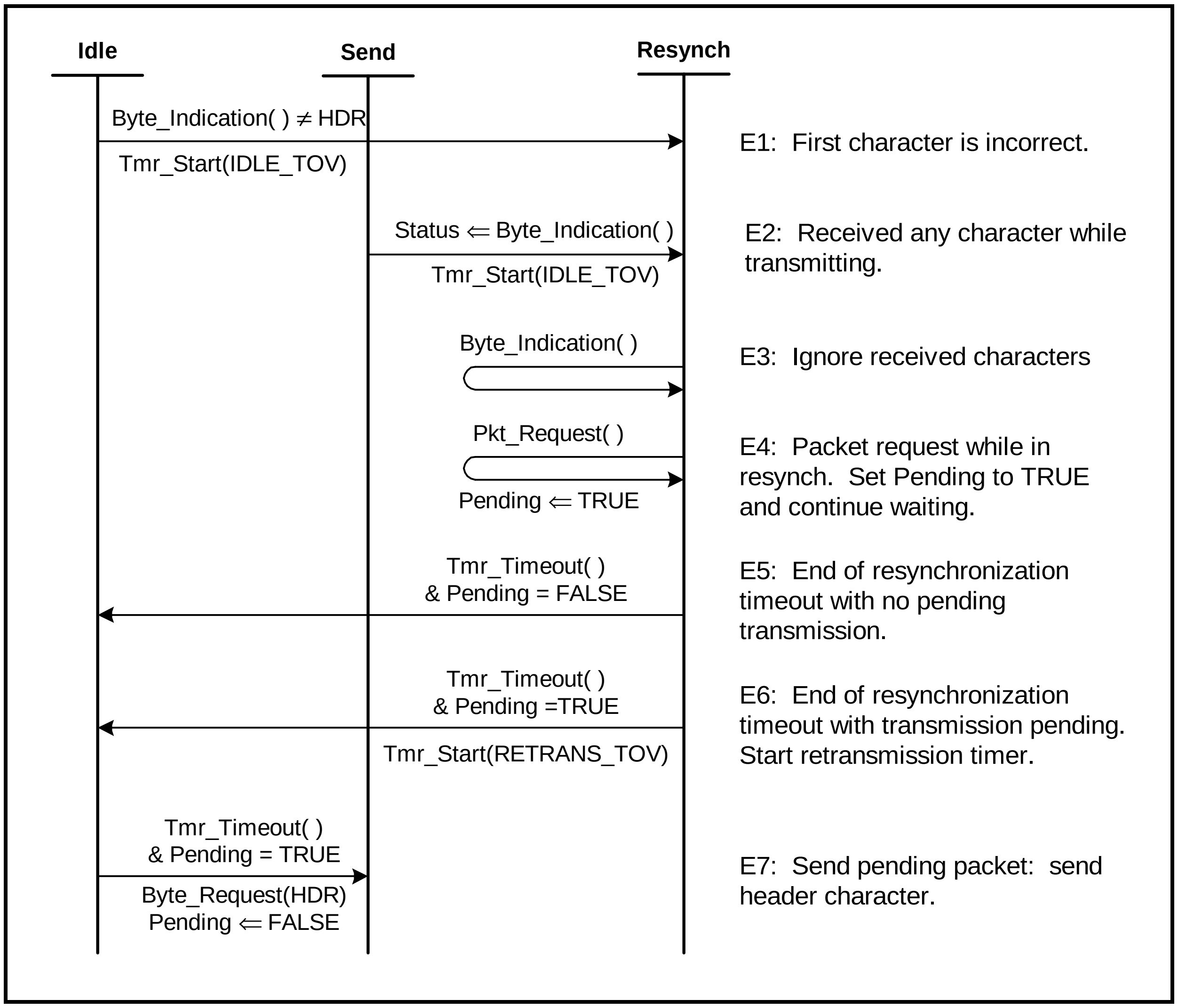

Idle
Send
Resynch
Byte_Indication( ) ≠ HDR
Tmr_Start(IDLE_TOV)
E1: First character is incorrect.
Status ⇐ Byte_Indication( )
Tmr_Start(IDLE_TOV)
E2: Received any character while transmitting.
Byte_Indication( )
E3: Ignore received characters
Pkt_Request( )
Pending ⇐ TRUE
E4: Packet request while in resynch. Set Pending to TRUE and continue waiting.
Tmr_Timeout( ) & Pending = FALSE
E5: End of resynchronization timeout with no pending transmission.
Tmr_Timeout( ) & Pending =TRUE
Tmr_Start(RETRANS_TOV)
E6: End of resynchronization timeout with transmission pending. Start retransmission timer.
Tmr_Timeout( ) & Pending = TRUE
Byte_Request(HDR)
Pending ⇐ FALSE
E7: Send pending packet: send header character.

Figure 8. Error (Resynchronization) State Machine

7.2 Packet Format

The packet is the basic unit of information that is transmitted across the Library Interface. A packet is a wrapper for various types of information transmitted between the library and a drive; this information is contained in a variable-length payload. The wrapper includes (i) information necessary to implement the link layer protocol to correctly deliver the payload: header character, length, payload, and checksum, and (ii) information common to all packets: type and transaction ID. In this section, the packet format is explained; specific payloads are in section 7.3.

Table 22. Packet Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Header Character (30h)							
1	Transaction ID							
2	(MSB) Payload Length (n-5)							
3	Payload Length (n-5) (LSB)							
4	Type							
5	Payload							
…	Payload							
n-1	Payload							
n	Checksum (XOR(0..n-1))							

7.2.1 Header Character

This byte is HDR (30h). Having the first character be a fixed value helps the receiver recognize a loss of synchronization between the sender and receiver state machines.

7.2.2 Transaction ID

This one byte field contains an unsigned integer identifying the transaction; all packets in the same transaction contain the same transaction ID. The transaction ID is assigned by the requester that is responsible for assigning unique IDs. Transactions initiated by the library use IDs in the range [1..127] and those initiated by the drive use IDs in the range [128..254]. This means that there can never be more than 256 transactions active at one time. IDs 0 and 255 are reserved.

7.2.3 Payload Length

This is the number of bytes in the payload. The minimum value is zero, for a packet with no payload. Placing the length in a fixed location in every packet simplifies the reception routine, allowing reception of the complete packet without complex decoding of its contents. The maximum value of the Payload Length field is 2048.

7.2.4 Type

This is a one-byte field that indicates the type of information in the packet. The details of each packet type are explained below.

Table 23. Packet Types

Type	Value	Sender	Payload Size	Implemented
Command	0	Library	24 (Note 3)	Yes
Transfer Ready	1	Both	2	Yes
Data	2	Both	[1..2048]	Yes
Response	3	Both	≥ 2 (Note 4)	Yes
Alert	4	Drive	1	No
Abort	5	Library	0 (Note 5)	Yes
Forwarded Command	6	Drive	32 (Note 3)	Yes
Final Data-In	7	Both	[1..2048]	Yes
Reserved	8 – 255			

Note 3	If the CDB is longer than 16 bytes, this value will be larger.
Note 4	A two-byte payload is a SCSI status and a transaction status with no autosense data. If the size is three or greater, the third and subsequent bytes are autosense data, that may be up to the maximum of 255. There are no other flags that indicate whether autosense data is included.
Note 5	Transaction ID is that for the transaction being aborted.

7.2.5 Payload

This is an optional field containing information whose length and format will depend upon the packet type. Some packet types have no payload. Contents of this field are explained in section 7.3.

7.2.6 Checksum

The checksum (last byte of the packet) is computed as the bit-wise exclusive OR of bytes 0 through n-1.

7.3 Packet Payloads

The packet payload contains both information destined for the application layer – such as a command queuing specifier and a SCSI CDB – as well as information used by the transaction layer to determine the correctness of the protocol – such as the transaction class and command data length

In the following figures, the byte offset is relative to the beginning of the packet; the first byte of every payload is at offset 5.

7.3.1 Command

This is a variable-length packet containing a SCSI CDB or Task Management command for execution by the drive. The format of the CDB is defined in the appropriate SCSI specification.

Table 24. Command Payload

Byte \ Bit	7	6	5	4	3	2	1	0
5	LUN (0)							
6	Task Attribute							
7	Function							
8	Reserved (0)						Transaction Class	
9	Command Descriptor Block							
…								
24								
	Additional CDB							
n-3	(MSB) Data Length							
n-2								
n-1								
n								(LSB)

7.3.1.1 LUN

The logical unit number (**LUN**) field is provided for compatibility with future drives that may support multiple logical units. In the current drive, this field must be zero.

7.3.1.2 Task Attribute

The **Task Attribute** field is meaningful if the transaction Type is Command; then it determines the queuing rules under which the SCSI task is to be managed.

Table 25. Task Attributes

Value	Attribute	Supported
0	Simple	Yes
1	Head of Queue	Yes
2	Ordered	Yes
3	ACA	Yes
4	Untagged	Yes
5 – 255	Reserved	N/A

These attributes are explained in section 7.6 of SAM-2. For Alert transactions, this field is not meaningful and must have a value of zero.

7.3.1.3 Function

This field specifies whether the command is a task management command or a SCSI command descriptor block. Not all task management commands are supported at this time.

Table 26. Function Values

Value	Command	Supported
0	SCSI CDB	Yes
1	Abort Task	No
2	Abort Task Set	Yes
3	Clear ACA	Yes
4	Clear Task Set	Yes
5	Logical Unit Reset	Yes
6	Target Reset	Yes
7	Forwarded Command	Yes
8 – 255	Reserved	N/A

The Abort Task task management request is supported via a packet with a Type of Abort. This permits the abort packet to use the transaction ID of the command being aborted, without requiring the Transaction Layer to inspect the contents of the payload.

7.3.1.4 Transaction Class

This field indicates whether the transaction will have read, write, or no data packets. It is not a protocol error for a transaction specifying read or write to complete with only a command packet and a response packet. This would occur if a check condition were encountered.

Table 27. Transaction Class Values

Value	Class	Commands Using this Class
00b	Non-data	SCSI non-data (e.g., Test Unit Ready), Task Management
01b	Read	SCSI read (e.g., Inquiry)
10b	Write	SCSI write (e.g., Mode Select)
11b	Reserved	None

This information duplicates information found elsewhere in the packet. However, placing it here removes the need for the responder to inspect the SCSI CDB's opcode (first byte) to determine whether the transaction will read, write, or neither.

7.3.1.5 Command Descriptor Block

This is the CDB defined in the SCSI-2, SPC, SSC, and SMC standards. A space of sixteen bytes is always reserved for it.

7.3.1.6 Additional CDB

This is a variable-length field to accommodate CDBs longer than sixteen bytes; it may be of length zero. The length of this field is equal to the payload length minus 24. The contents are the portion of the CDB in excess of sixteen bytes.

7.3.1.7 Data Length

This field indicates the total number of data bytes to be transferred by the command. It is used by the transaction layer to insure that the receiving buffer is not overrun; if the total amount of data transferred would be larger than that originally specified, the entire data packet is discarded and the transaction is terminated. None of the data in the packet is passed to the application layer. It is not an error for a command to transfer less data than originally specified, as in a SCSI Inquiry command.

7.3.2 Transfer Ready

This packet is sent by the drive to inform the library that it is ready to receive data. It has a two-packet payload that is the maximum amount of data that may be transferred in the next data packet. This is analogous to the disconnect in parallel SCSI, and is used to avoid overflowing data buffers. If the data cannot be transferred in a single data packet, then there will be additional transfer ready / data packet pairs.

Table 28. Transfer Ready Payload

Byte \ Bit	7	6	5	4	3	2	1	0
5	(MSB)	Data Length						
6								(LSB)

7.3.3 Data

This is a variable-length payload containing either outbound or inbound data required by SCSI commands such as Write Buffer, Request Sense, etc. The format of this data is as defined in the

SCSI-2 specification, with vendor-specific extensions as defined in section 5. The maximum length of a data payload is 2048 bytes.

Table 29. Data Payload

Byte \ Bit	7	6	5	4	3	2	1	0
5	Data							
…								
n-1								

7.3.4 Response

The response payload consists of two or more bytes comprising three mandatory and one optional field.

Table 30. Response Payload

Byte \ Bit	7	6	5	4	3	2	1	0
5	Service Response							
6	SCSI Status							
7	Autosense Data (optional)							
…								
n-1								

7.3.4.1 Service Response

The **Service Response** field is defined in the following table. If the response packet is for a SCSI command, this field contains zero.

Table 31. Service Responses

Value	Response Name	Meaning
0	Command Complete	SCSI or task management command completed correctly
1	Function Rejected	Task management command not supported by responder
2	Function Failed	Task management command supported but failed
3	Invalid ID	Transaction ID is invalid (used for Abort only)
4	Invalid Command	Command payload field(s) invalid
5	Write Length Mismatch	Length of write data did not match transfer ready
6	Read Data Failure	Read data packet transmission error; transaction ended
7	Status Failure	Response packet transmission error; transaction ended. SCSI status will be the same as in the original response packet.
8	Transaction Aborted	Requester sent Abort Transaction

7.3.4.2 SCSI Status

The SCSI status byte is defined in the SCSI-2 specification. If the response packet is for a task management command, this field contains zero.

7.3.4.3 Autosense Data

This is a variable-length field, as defined in [SAM-2]. The length of this field is determined from the packet **Length** field. Autosense data is not supported in the Library interface on parallel SCSI drives.

7.3.5 Alert

The alert payload has a one byte field, **Alert Type**, which explains the reason for the alert. The alert is a single-packet transaction. While there will be an acknowledgement character from the library, there is no packet sent to the drive as a response.

Note: Alerts are not currently implemented in the Viper drive.

Table 32. Alert Types

Alert Type Value	Meaning	Recommended Action by Library
0	Unit Attention	Issue Request Sense
1	Detect Deferred Errors	Issue Request Sense
2	Tape Alert	Issue Log Select for Tape Alert page, 2Eh

7.3.6 Abort

This command is generated by the transaction layer to recover from protocol errors, such as an undeliverable, invalid, or lost packet. It is different from the Task Management Abort Task command, in that the latter is generated by the application layer to abort commands that have experienced no protocol errors.

When the requester (library) detects a protocol error, such as an error sending a data packet, it will send Abort Transaction to the responder to cause it to terminate command execution and return status. Other details of this error handling are in section 5.7.

7.3.7 Forwarded Command

Command forwarding requires that the library be notified of the identity of the initiator issuing the command. For Parallel SCSI, this is a one-bye SCSI ID; for Fibre Channel, this is a 64-bit WorldWide Name. Therefore forwarded command packets must have an additional eight-byte field in the payload to contain the initiator ID. The Type field in the packet header will contain the new value **FORWARDED COMMAND**, which is different from that of a standard command issued by the library to the drive. Both modes of operation will use the **FORWARDED COMMAND** packet.

Table 33. Forwarded Command Payload

Byte \ Bit	7	6	5	4	3	2	1	0
5	(MSB)							
...				Initiator ID				
12								(LSB)
13				LUN (1)				
14				Task Attribute				
15				Function				

16	Reserved (0)	Transaction Class
17	(MSB) Command Descriptor Block	
...		
32		(LSB)
...	Additional CDB	
n-5	(MSB) Data Length	
n-4		
n-3		
n-2		(LSB)
n-1	Command Reference Number	
n	Checksum (XOR(0..n-1))	

InitiatorID identifies the initiator which sent the command. For Parallel SCSI, the initiator's SCSI ID is placed in the LSB (byte 12) and the previous seven bytes are zero. For Fibre Channel, this field contains the World Wide Name of the initiator.

The **LUN** field will always be one.

In Fibre Channel drives, the **Command Reference Number** contains the CRN sent by the initiator in the COMMAND REFERENCE NUMBER field of the FCP command information unit. This field is for Parallel SCSI drives.

All other fields are as defined in section 7.3.1, Command payload.

7.3.8 Final Data-In

The Final Data-In packet will be used to complete a data-in transaction when the last data packet is ready to be transmitted and the responder has determined that the command will have a status of GOOD STATUS. Except for the **Type** field, this packet is identical to a Data packet. The payload is the data and the **Type** field implies that the Service Response is Command Complete and the SCSI Status is GOOD STATUS. This is the final packet of the transaction.

8. Physical Layer

- Levels: RS-422
- Data rate(s): 9600 baud default; settable to 4800, 9600, 19200, 38600, 57600, or 115200 baud.
- Framing: 8 data bits, selectable one or two stop bits, no parity.
- Cable and connector: see Seagate Removable Storage Solutions tape drive specification.