NOTE: Copies of this document may be purchased from:
Global Engineering Documents, 2805 McGaw, Irvine, CA 92714
(800) 854-7179 or (714) 261-1455.

79
X3T9.2/92-~~5~~R1
Revision 1

Working draft proposed
American National Standard
for information systems -

SCSI-3 Architecture Model
(SAM)

June 19, 1992

Secretariat

Computer and Business Equipment Manufacturers Association

Abstract: This document is an working draft of the SCSI-3 Architectural Model. The purpose of the architecture is to define a model for device behavior that is independent of physical interconnect technology.

This is an internal working document of X3T9.2, a Task Group of Accredited Standards Committee X3. As such, this is not a completed standard. The contents are actively being modified by the X3T9.2 Task Group. This document is made available for review and comment only.

POINTS OF CONTACT:

X3T9.2 Chair	X3T9.2 Vice-Chair	Technical Editor
John B. Lohmeyer NCR Corporation 1635 Aeroplaza Dr. Colorado Springs, CO 80916	I. Dal Allan ENDL 14426 Black Walnut Court Saratoga, CA 95070	Charles Monia Digital Equipment Corporation 334 South Street Shrewsbury, MA 01545
Tel: (719) 596-5795 x 362 Fax: (719) 574-0424	Tel: (408) 867-6630 Fax: (408) 867-2115	Tel: (508) 841-6757 Fax: (916) 841-6400
Internet: jlohmeye@ncr-mpd.ftcollins.ncr.com	Internet: 2501752@mcimail.com	Internet: monia@starch.enet.dec.com

Draft working document SCSI-3 Architecture Model X3T9.2/92-79R1

STATUS OF EDITING:

Rev 0: Draft of glossary and outline

Rev 1: Completed SCSI Model Definition

i

63

2

X3T9.2/92-079 R1

1 Forward 4

2 Scope 5

3 Referenced Standards and Organizations 6

4 Glossary and Conventions 6
4.1 Glossary 6
4.2 Object Notation 9
4.3 Editorial Conventions 10

5 SCSI Architectural Model 10
5.1 Introduction 10
5.2 Request-Response Transactions 11
5.2.1 Orphan Detection 12
5.3 SCSI I/O Subsystem Elements 12
5.4 SCSI Domain 13
5.4.1 Logical Interconnect 14
5.5 SCSI Device 15
5.5.1 SCSI Transport Services 16
5.5.1.1 Transport-Initiated Resets 17
5.5.2 Initiator 18
5.5.3 Target 20
5.5.4 SCSI Transaction Example 22

6 I/O Control Operations 25
6.1 Command Processing Considerations and Exception Conditions 25
6.2 Contingent Allegiance 25
6.3: Auto Contingent Allegiance 25
6.4 Extended Contingent Allegiance Condition 25
6.5 Queued I/O Processes 25
6.6 Unit Attention Condition 25
6.7 Multiport SCSI Devices 25

1 Forward

With any technical document there may arise questions of interpretation as new products are implemented. The X3 Committee has established procedures to issue technical opinions concerning the standards developed by the X3 organization. These procedures may result in SCSI Technical Information Bulletins being published by X3.

These Bulletins, while reflecting the opinion of the Technical Committee which developed the standard, are intended solely as supplementary information to other users of the standard. This standard, ANS X3.xxx-199x, as approved through the publication and voting procedures of the American National Standards Institute, is not altered by these bulletins. Any subsequent revision to this standard may or may not reflect the contents of these Technical Information Bulletins.

Current X3 practice is to make Technical Information Bulletins available through:

Global Engineering Documents
2805 McGaw
Irvine, CA 92714
(800) 854-7179
(714) 261-1455

2 Scope

The documents under X3T9.2 jurisdiction are:

1. SCSI-3 Block Device Command Set
2. SCSI-3 Stream Device Command Set
3. SCSI-3 Other Device Command Set
4. SCSI-3 Architecture Model
5. SCSI-3 Common Access Method
6. SCSI-3 Interlocked Protocol
7. SCSI-3 Parallel Interface

The original Small Computer System Interface Standard, X3.131-1986, is referred to herein as SCSI-1. SCSI-1 was revised resulting in the Small Computer System Interface - 2 (X3.131-199x), referred to herein as SCSI-2. This standard, the SCSI-3 Interlocked Protocol, and the SCSI-3 Command Set are referred to herein as SCSI-3. The term SCSI is used wherever it is not necessary to distinguish between the versions of SCSI.

3 Referenced Standards and Organizations

4 Glossary and Conventions

4.1 Glossary

This section contains a glossary of special terms used in this standard.

4.1.1 ACA: Auto contingent allegiance

4.1.2 ACA breakthrough command: A tagged command that has been issued with the tag field set to ACA BREAKTHROUGH. Such commands are legal if and only if an ACA condition is present on the target LUN/TRN.

4.1.3 AEN: Asynchronous event notification.

4.1.4 Auto contingent allegiance: The state of a target LUN following the return of a CHECK CONDITION status, after which further I/O processing is suspended except for ACA breakthrough commands.

4.1.5 byte: In this standard, this term indicates an 8-bit construct.

4.1.6 CAM: Common Access Method

4.1.7 Command descriptor block: The structure used to communicate commands from an initiator to a target.

4.1.8 Common access method: A host architecture for interfacing to SCSI devices, as described in the document entitled "SCSI-2 Common Access Method, Transport and SCSI Interface Module".'

4.1.9 Current I/O Process: An I/O process that is sending or receiving information over the physical interconnect.

4.1.10 Device Model: The object, within a target LUN, that performs the set of device operations defined by one of the SCSI device specifications (SBC, SBS or SCCS).

4.1.11 Domain: The set of SCSI devices and physical interconnects that, collectively, comprise a single, fully-connected network. In such a network, all devices have the same view of the device configuration and each device may communicate with any other device in the domain. A device may belong to multiple domains.

4.1.12 function: An interface between objects residing on the same SCSI device.

4.1.13 initiator: An SCSI device (usually a host system) that requests an I/O process to be performed by another SCSI device (a target).

4.1.14 initiator I/O Process: An I/O process, residing in the initiator, which interacts with a cooperating target I/O process to perform an SCSI command. An initiator I/O process is created when an application command is received and is normally terminated following the receipt of COMMAND COMPLETE status message from the target or the transmission of a RELEASE RECOVERY I/O control command. An initiator I/O Process also ends whenever an ABORT, ABORT TAG, BUS DEVICE RESET, or CLEAR QUEUE I/O control command is sent.

4.1.15 interconnect: A physical pathway for the transfer of commands and data between SCSI devices in a domain.

4.1.16 I/O Process Control Object: An object residing in the initiator or target that executes incoming process control commands or services outgoing commands by interacting with a cooperating I/O process control object on another SCSI device.

4.1.17 I/O process: An object residing in the initiator or target LUN/TRN that executes an SCSI command or series of linked commands. Each SCSI command or series of linked commands causes the creation of a pair of cooperating I/O processes, within the initiator and target, which interact to execute the operation.

4.1.18 I/O Process Queue: A target-resident list of uncompleted I/O processes.

4.1.19 Linked SCSI Command: A SCSI I/O command consisting of two or more command descriptors, each of which has the LINK flag set in the CDB control field, as specified in the SCSI Command Standard.

4.1.20 Linked Command Element: A single CDB issued with the LINK flag asserted as part of a linked SCSI command.

4.1.21 logical interconnect: One or more physical interconnects that, together, constitute the data path for a domain. When a logical interconnect is made up of multiple physical interconnects, some combination of software and hardware is used to make the data path appear monolithic to higher layers.

4.1.22 logical unit: A physical or virtual peripheral device addressable through a target.

4.1.23 logical unit number: An encoded xxx-bit identifier for the logical unit.

4.1.24 LUN: Logical unit number.

4.1.25 mandatory: The referenced item is required to claim compliance with this standard

4.1.26 multipath interconnect: An interconnect comprised of several independent physical paths between the SCSI devices in a domain. These paths may be used to increase bandwidth, through "striping", and to improve availability through redundancy. In an SCSI environment, multipath interconnect management may performed by the transport service in a manner that is transparent to the device.

4.1.27 nexus: A relationship between cooperating initiator and target I/O processes that begins when the command descriptor block is sent and ends with termination of one or both processes.

4.1.28 object: An architectural abstraction that encapsulates data types, functions, or other objects.

4.1.29 optional: The referenced item is not required to claim compliance with this standard.

4.1.30 orphan: A transaction request or response addressed to or generated by a transactor that no longer exists.

4.1.31 packetized SCSI: An SCSI variant that uses a packetized interconnect as the transport mechanism for commands and data.

4.1.32 port: A single attachment to a physical interconnect from an SCSI device. SCSI devices may have multiple ports, each attached to the same or a different physical interconnect.

4.1.33 port address: The physical address of a port attached to an interconnect.

4.1.34 protocol: The rules governing the content and exchange of information passed between cooperating objects residing on different SCSI devices in a domain.

4.1.35 reserved: The term used for bits, fields, signals, and code values that are set aside for future standardization.

4.1.36 SCSI: Either SCSI-2 or SCSI-3.

4.1.37 SCSI-2: The Small Computer System Interface - 2 (X3.131-199X).

4.1.38 SCSI Device: A physical device in an SCSI domain.

4.1.39 SCSI device address: An address by which an SCSI device is referenced within a domain. The device address need not correspond to the port address. Each SCSI device within a domain may have one and only one SCSI device address.

4.1.40 SCSI interlocked protocol: A protocol, defined by the SIP standard, that uses low-level bus signals to control and synchronize the states of the initiator and target. In the SIP protocol, such signals are used to initiate and transfer data associated with SCSI control, command and device I/O operations.

4.1.41 SCSI packetized protocol: Any SCSI device protocol that is designed to be implemented using a packetized interconnect technology. In a packetized protocol, message transactions replace the use of low-level bus signals as a way of initiating and passing data associated with control, command and device I/O operations. Except to control the transfer of data, bus signals play no part in an I/O transaction that uses a packetized protocol.

4.1.42 System Application: An entity, such as a CAM implementation, that is the source of SCSI I/O commands and I/O control commands.

4.1.43 target: An SCSI device that performs an I/O operation requested by an initiator.

4.1.44 target I/O Process: An I/O process, residing in the target device, that interacts with a cooperating initiator I/O process to perform an SCSI command. A target I/O process is created when a device command is received from the initiator and normally terminates by sending a COMMAND COMPLETE status message or when a RELEASE RECOVERY I/O control command is received. A target I/O Process also ends in the event of a transport-initiated reset, or whenever an ABORT, ABORT TAG, BUS DEVICE RESET, or CLEAR QUEUE I/O Control command is executed.

4.1.45 third-party command: An SCSI command, such as COPY, which requires the target device to assume the initiator role and transmit one or more commands to another device on behalf of the original initiator.

4.1.46 transport service: A combination of hardware and software that is responsible for the reliable delivery of request-response transactions between cooperating objects. When transmitting, this service maps the destination object address into the address of the associated physical port, decomposes data blocks into packets as required by the physical interconnect and transmits each packet to the destination port. When receiving, the transport service is responsible for the reassembly of packets into a replica of the transmitted data and delivery to the addressee.

When the device is attached to a multipath interconnect, the transport service manages the flow of traffic over the interconnect in a way that is transparent to the SCSI device.

4.1.47 transport-initiated reset: Hard reset initiated by the transport control services layer. This function is provided as a way to externally reset a device to it's power-on state when an error is detected that prevents the device from responding to I/O control commands.

4.1.48 word: In this standard, this term indicates a 1-byte, 2-byte, or 4-byte construct.

4.1.49 xx: Digits 0-9, except those used as section numbers, in the text of this standard that are not immediately followed by lower-case "b" or "h" are decimal values. Large Numbers are not separated by commas or spaces (e.g., 12345; not 12,345 or 12 345).

4.1.50 xxb: Digits 0 and 1 immediately followed by lower-case "b" are binary values.

4.1.51 xxh: Digits 0-9 and the upper-case letters "A"-"F" immediately followed by lower-case "h" are hexadecimal values.

4.2 Object Notation

The following notational conventions are used to describe composite objects:

+	"together with" as in A = B + C. Object A contains both B and C.
[|]	"select one of" as in A = [B|C]. Object A contains either B or C, but not both. This is equivalent to an "exclusive or" operation.
()	"optional" as in A = (B) + (C + D). Object A contains either B or C + D. This is equivalent to an "inclusive or" operation.

{ }	"Iterations of"	A set of objects enclosed within curly brackets may occur any number of times in a given instance. The brackets may be indexed: For example, M{}N indicates any number of instances from M to N. Thus: {...}3 implies 0, 1, 2 or 3 instances 3{...} implies 3 of more instances 3{...}3 implies exactly 3 iterations
"xxx"	literal	Symbols enclosed within quotes literally constitute the object.
...	range	Denotes a sequential set of literals. Thus: ["1"|....|"100"] implies one out of a set of literal integers between 1 and 100. ["A"|...|"Z"] implies one out of a set of literal alphabetic characters between "A" and "Z".

4.3 Editorial Conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in the glossary or in the text where they first appear. Names of signals and phases are in all uppercase. Lower case is used for words having the normal English meaning.

5 SCSI Architectural Model

5.1 Introduction

The purpose of the SCSI Architectural Model is to:

a. Specify required device behavior that is visible to a system application and common to all SCSI devices. A system application is an entity, such as a CAM implementation, that submits I/O requests to the SCSI subsystem.

b. Define SCSI I/O subsystem behavior in a manner that is independent of a specific physical bus implementation and compatible with both packetized and interlocked bus protocols.

This specification assumes that, except for differences in technology and performance, all SCSI interconnects, when supplemented by suitable hardware and firmware, are capable of providing functionally equivalent levels of service. The behavior defined in this document assumes that all device operations are controlled through information exchanges using the data transfer facilities of the bus. The required level of functionality is consistent with that provided by any bus capable of transferring data in packets.

The SCSI architecture is described in terms of objects, functions and protocols. The SCSI objects defined in this standard are abstractions that encapsulate a set of related operations, data types and other objects. Functions and protocols define interfaces between objects. A function is an interface between objects that

reside in the same SCSI device; a protocol is an interface between objects on different SCSI devices. The template for function and protocol interfaces is the request-response transaction described in section 5.2.

5.2 Request-Response Transactions

Protocol and function interfaces may be specified in a simple and general way by defining them as request-response transactions between cooperating objects. As shown in Figure 1a, such transactions consist of interactions between a requesting and a responding object, possibly through intervening objects, which may, themselves, interact via request-response transactions. In effect, such intermediaries are invisible and the net result is as if the requestor and responder were directly interfaced as shown in Figure 1B. The requestor begins the transaction then waits for a response or a notification that the transaction did not complete. ie: a notification that no response was received.

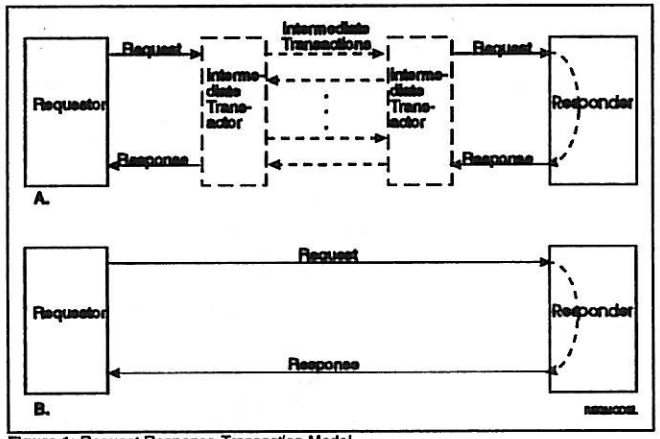

Figure 1: Request-Response Transaction Model

It is assumed that the following rules apply to request-response transactions:

a. Termination - All request-response transactions terminate either with a response or a notification of failure to complete.

b. Singularity - Delivery of a response to the requestor guarantees that the transaction was serviced once and only once by the responder.

c. Absence of after-effects - Once a response or failure notification has been delivered, no further responses associated with the completed transaction will be presented to the requestor.

d. The requestor can make no assumption about the operation associated with a failed (incomplete) transaction. The operation may never have been started, or may be partially or totally complete.

e. Each requestor will have no more than one request-response transaction pending at any time. A requestor will wait for a response or failure notification before issuing another request.

f. Responses are sent "in the blind". ie: with no confirmation of delivery. It is the requestor's responsibility, or the responsibility of some agent acting on behalf of the requestor to detect a response failure. Of course, an implementation may choose to provide such delivery guarantees.

g. If several requestors independently initiate concurrent transactions with a single responder, there are no guarantees about the order in which such transactions will be presented to the responder or the order in which responses will be received by requestors. All requestors must explicitly coordinate such requests when a deterministic sequence of transactions is required.

5.2.1 Orphan Detection

An orphan is data associated with a response or request addressed to or generated by an object that no longer exists. This situation may arise in a non-interlocked, packetized implementation when a target I/O process is aborted. If there are no implicit transaction ordering guarantees, it is possible for the target to service the abort command while a response or request for the terminated I/O process is in transit. Hence such transaction components can arrive after the addressee has been deleted.

Similarly, it is possible for the initiator to receive data associated with a response or request that was sent by a target I/O process prior to it's deletion.

It is the responsibility of the implementation to either avoid or detect and discard all orphans.

5.3 SCSI I/O Subsystem Elements

The SCSI architecture is defined in terms of an object hierarchy. The fundamental object is the *SCSI Domain* - a set of SCSI devices and a *logical interconnect*, as described in section 5.4, which, together, make up the essential elements of a functioning I/O subsystem. Each SCSI device may, itself, consist of an initiator and one or more target LUNs or target routines. Each of these, in turn, may be composed of other objects, such as I/O Processes, I/O Process queues and so forth. The following diagram shows a partial hierarchy, giving the relationship between the SCSI Domain, the interconnect and SCSI devices.

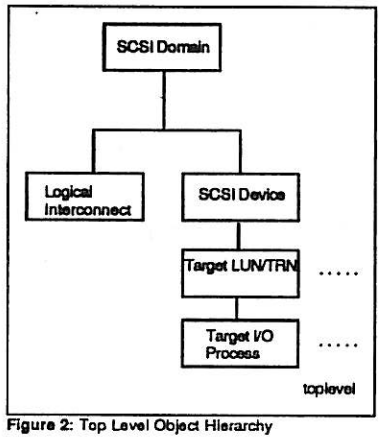

Figure 2: Top Level Object Hierarchy

The following sections describe SCSI objects in detail, using the notation for composite objects specified in section 4.2

5.4 SCSI Domain

An SCSI domain is a set of SCSI devices and a logical interconnect that, together, comprise the fully connected network shown in Figure 3. A SCSI Domain is a generalization of the SCSI-2 subsystem topology, extended to allow the use of any suitable interconnect technology and to accommodate any number of devices.

The *logical interconnect* is some combination of physical transport implementations, designed and configured to meet specific performance, availability, connectivity and cost requirements. As described in ?, a combination of device-resident hardware and firmware makes the interconnect subsystem appear as a single integrated transport mechanism.

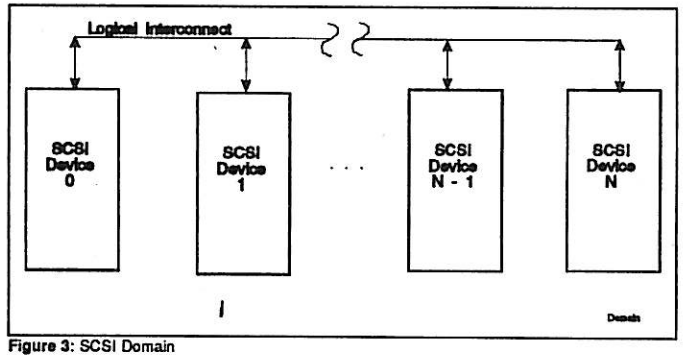

Figure 3: SCSI Domain

Object Definition 1 SCSI Domain

$$SCSI\_Domain = 2[SCSI\_Device]256 + Logical\_Interconnect$$

In this topology, every device in the domain may communicate with all other devices and each sees the same configuration. These properties allow a third-party I/O operation to be requested between any two SCSI devices in the domain.

5.4.1 Logical Interconnect

A logical interconnect provides the data pathway between SCSI devices and is made up of one or more physical interconnects that are used for the transfer of commands and data between devices in an SCSI domain. A simple example of a logical interconnect is a single-ended, interlocked SCSI bus. A more complex implementation may consist of a multi-path system with several redundant physical interconnects that are used to enhance data throughput or system availability.

One example of a multi-path implementation might be a collection of dual-ported SCSI devices, with separate physical interconnects connected to each port for increased availability. In this configuration, the initiator could dynamically select a bus for arbitration based on whether or not the associated physical interconnect was in use. As described in ?, a bus reset or other failure on one bus would not effect data transfers occurring on the other.

A logical interconnect may also be comprised of mixed interconnect technologies, such as short and long-haul physical interconnects, that are interfaced via bridges or other devices such that a seamless view of the data path is presented to all devices in the domain.

In the SCSI architecture model, functions related to managing and transferring data over the logical interconnect are delegated to the **Transport Services** object as described in Section 5.5.1. The model assumes that objects which transfer data over the logical interconnect do so using the services of this object and therefore need have no knowledge of the physical interconnect configuration.

The formal definition of a logical interconnect is given in Object Definition 2

Object Definition 2: Logical Interconnect

Logical_Interconnect = 1 { *Physical_Interconnect* }

5.5 SCSI Device

An SCSI device is a physical element containing the following objects:

a. A Transport Services object,

b. An initiator and zero or more target LUN/TRN objects or,

c. An optional initiator and one or more target LUN/TRN objects.

The device model is shown graphically in Figure 4 and described formally in Object Definition 3

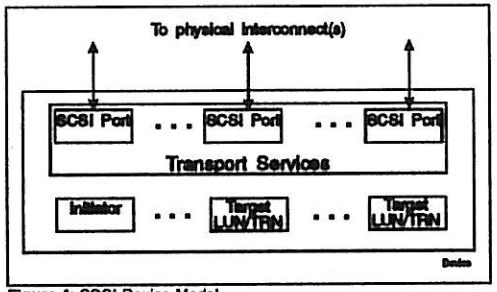

Figure 4: SCSI Device Model

Object Definition 3: SCSI Device

SCSI_device = *Transport_Services* + [*Initiator* + 0{*Target_LUN/TRN*256

| (*Initiator*) + 1{*Target_LUN/TRN*256 } + *Device_identifier*

Device_identifier = *Byte*+["0"|...|"255"]

The *Device Identifier* shall be unique within an SCSI Domain.

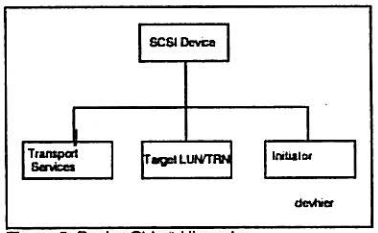

Figure 5: Device Object Hierarchy

5.5.1 SCSI Transport Services

The Transport Services object shown in Figure 4 is a facility for communication between objects residing on different SCSI devices; it provides the following services:

a. Conveys request-response transactions between cooperating objects with a very high degree of reliability, notifying the requestor whenever a transaction fails to complete. Such failures are considered to be unrecoverable.

b. Optionally, implements a transport-initiated reset mechanism as described in section 5.5.1.1

The Transport Services object contains one or more *SCSI Ports*, each of which attaches to a physical interconnect. The port represents the point at which data from a physical interconnect enters or leaves the device and it's function is the port-to-port transfer of data over the physical bus.

As shown in Figure 6, the input from the local transactor is a request or a response to be forwarded to a transactor in another SCSI device. The local Transport Services object decomposes this into a series of port-to-port protocol transactions, which are reassembled by the remote Transport Services object and presented to the remote transactor.

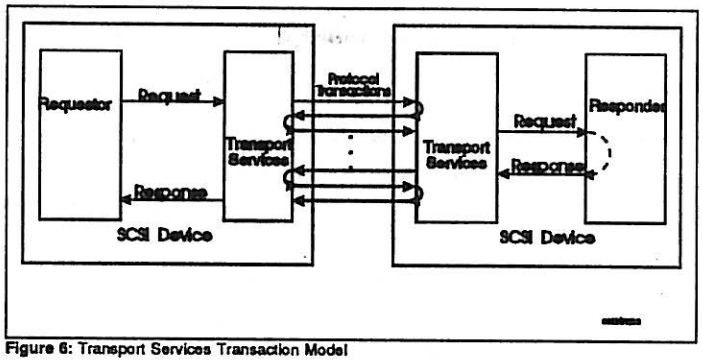

Figure 6: Transport Services Transaction Model

On output, the local Transport Services object must:

a. In a multi-port implementation, select the SCSI port to be used.

b. Decompose outgoing data into segments as required by the physical interconnect protocol.

c. Map the destination object address to a remote port address.

d. Transmit the segmented data, interacting with the port and the receiver as necessary to recover from transmission errors.

e. When servicing a locally initiated request, report a transaction completion failure to the requestor.

The receiving Transport Services Object is required to:

a. Accept incoming data from the port.

b. Interact with the sender as needed to recover from transmission errors.

c. Reassemble incoming segments into a replica of the transmitted data and deliver that data to the addressee.

5.5.1.1 Transport-Initiated Resets

A transport-initiated reset is a signal to the SCSI device, conveyed via the logical interconnect, which instructs the receiving device to execute a BUS DEVICE RESET operation. Support for this function is an interconnect implementation option.

The purpose of this function is to trigger a device reset without having to rely on the normal mechanisms for delivering and executing I/O commands. When recovering from an error it is used as a last resort if

there is reason to believe that the request delivery mechanism may be broken. ie: when there is no way to deliver a BUS DEVICE RESET command.

The behavior of a transport-initiated reset is interconnect-specific. An example is the HARD RESET, implemented as part of the SCSI Interlocked Protocol, which has the effect of broadcasting a BUS DEVICE RESET to all connected devices.

5.5.2 Initiator

An initiator object originates I/O service requests by sending commands to SCSI target devices. To the system application, the only visible initiator features are the command-response interface, the interface for passing command data and an optional sense data interface. The target-visible features are all implementation protocol-dependant and are described in the appropriate SCSI protocol standard.

The set of objects described here are specified to complete the behavioral model and need not be present in an implementation. The only requirements are that:

a. The behavior, as seen by the System Application, complies with the behavior described in this standard and

b. The behavior, as seen by the target device, complies with the requirements of the appropriate protocol standard.

An initiator may be composed of the following objects which are paired with corresponding objects in a target device:

a. Initiator I/O process object - Interacts with a cooperating target I/O process to execute a command or series of linked commands. An initiator I/O process consists of the procedures for servicing an I/O request, a data buffer, if required by the command, and an optional buffer for sense data.

b. Initiator I/O Process control object - interacts with a target I/O process control object to create or terminate I/O processes.

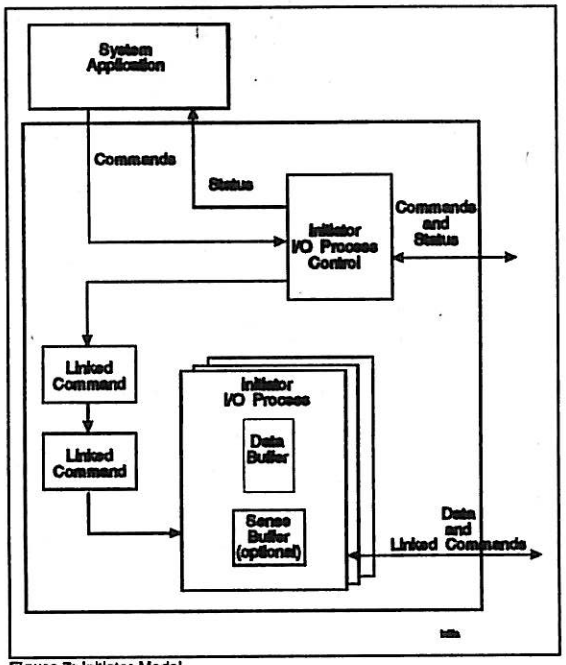

Figure 7: Initiator Model

Object Definition 4: Initiator Model

Initiator = 1{*Initiator_I/O_process*} + *Initiator_I/O_process_control*

Initiator_I/O_process = *Data_buffer*+{*Sense_buffer*) + *Initiator_I/O_Process_Procedure*

As shown in Figure 7, System Application commands are serviced by the Initiator I/O Process Control object, which translates them into protocol-dependant I/O Control requests, serviced by the Target I/O Process Control object.

A single system application command may consist of one SCSI CDB or a series of two or more linked CDBs as shown above. In the case of a linked command, all command elements shall be passed to the initiator in a single operation. Once such a command is started, the system application is not notified until either all linked command elements have been processed or the operation terminates because of an error. There will be no way for a System Application to intervene based on any status returned after the completion of an intermediate linked command element.

5.5.3 Target

The target consists of the following objects:

a. Target I/O process control object - Creates, deletes and queues processes as requested by the initiator,

b. One or more Target LUNs or Target Routines.

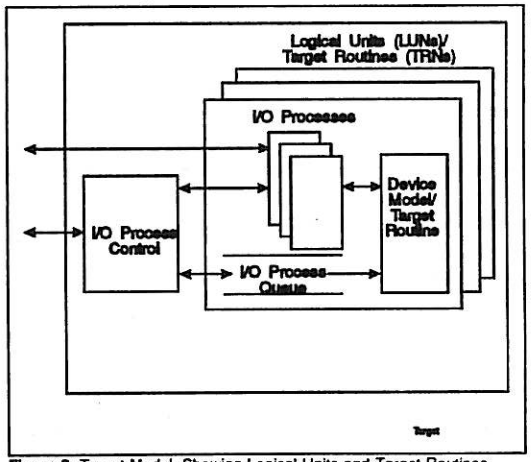

Figure 8: Target Model, Showing Logical Units and Target Routines

Object definitions 5, 6 and 7 describe the Target, Target LUN/Target Routine and Target I/O Process models.

Object Definition 5: Target Model

Target = Target_I/O_process_control + 1{Target_LUN/TRN}256

Object Definition 6: Target LUN/TRN Model

Target_LUN/TRN = 1{I/O_Process_Slot}256+0{Target_I/O_Process}256 + I/O_Process_Queue

+ [Device_Model|Target_Routine_Model]

+Target_LUN/TRN_I/D

Target_LUN/TRN_I/D = Byte + [LUN_I/D|TRN_I/D] + Device_Identifier

LUN_I/D = ["0"|...|"127"]

TRN_I/D = ["128"|...|"255"]

Note that the *I/O Process Slot* represents target LUN resources that can be allocated to store I/O process state. As indicated in the above definition, a target LUN/TRN object shall have enough resources to create at least one target I/O process, although, at any time, no target I/O processes might exist.

Object Definition 7: Target I/O Process

Target_I/O_Process = Target_I/O_Process_Procedure + Target_I/O_Process_I/D

Target_I/O_Process_I/D = Target_I/O_Process_Tag + Target_LUN/TRN_I/D

Target_I/O_Process_Tag = Byte + ["0"|...|"255"]

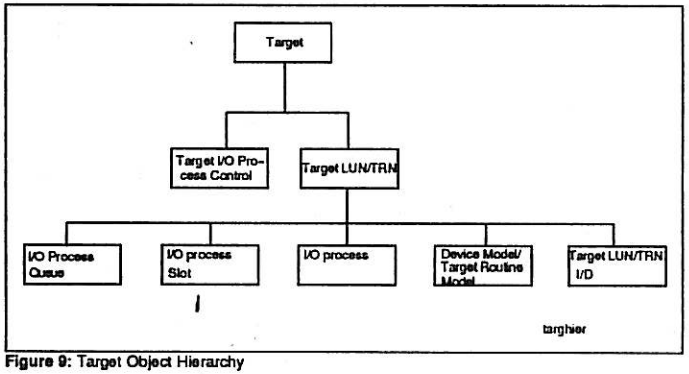

Figure 9: Target Object Hierarchy

5.5.4 SCSI Transaction Example

Figure 10 shows how the various components of the model interact to perform an SCSI Read request. The horizontal arrows denote *effective* protocol or request-response transactions between cooperating objects in the target and initiator. That is, even though the actual exchange may have required the intervention of other transactors, such as the Transport Services object, the transaction is considered to have occurred directly between the requesting and responding objects.

The function interfaces that generate these exchanges are shown as vertical arrows.

For this example, it is assumed that the following functions and protocol request-response transactions are implemented, which mirror the command, data and status transactions of the SCSI interlocked protocol. Since the intent is to convey a general picture of model behavior, many details have been elided.

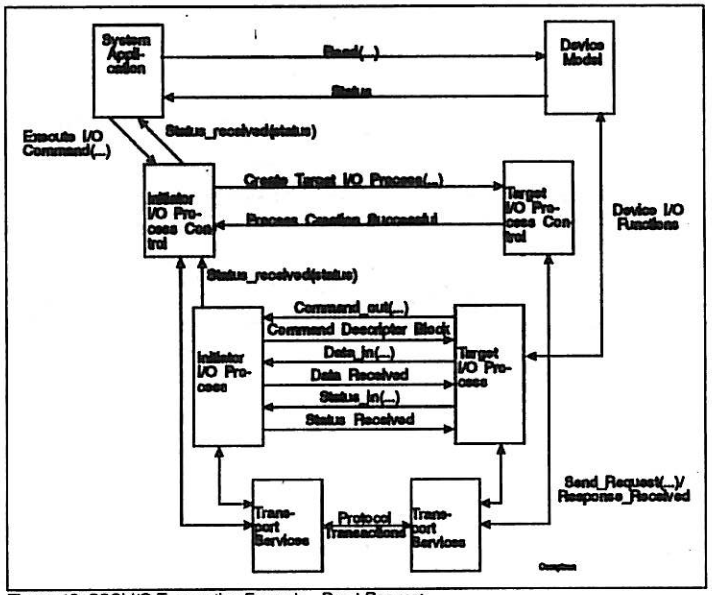

Figure 10: SCSI I/O Transaction Example - Read Request

I/O Process Control Functions:

Request: Execute_I/O_Command (Command_Descriptor, Target_I/O_Process_I/D ...)

Response: Status_Received(status)

Description: Executes the command specified in the command descriptor block and returns command status to the System Application.

I/O Process Control Protocol Transactions:

Request: Create_target_I/O_Process (Target_I/O_Process_I/D)

Response: Process creation successful
Process creation failure.....

Description: Creates a target I/O Process to service a command.

Target I/O Process Protocol Transactions:

Request: Command_out(...)

Response: Success - Command descriptor block sent.
Failure.....

Description: Solicits the transfer of an SCSI command descriptor block from the initiator I/O process. This operation is functionally analogous to the act of passing the command descriptor during the SIP Command Out phase.

Request: Data_in(data_block_size, data_buffer_offset)

Response: Success - Requested data block received.
Failure.....

Description: Delivers a block of data to the initiator I/O process data buffer. The block is deposited at the specified offset relative to the start of the buffer. This function corresponds to the delivery of command data during a Data In phase

Transport Services Transactions:

Request: Send_request(addressee, request_descriptor, response_descriptor....)

Response: Response successfully received,
Response failure....

Description: Sends the specified request to the addressee and returns the response to the requestor (or notifies the requestor of a response failure).

As shown in Figure 10, the System Application invokes the initiator I/O process control object with a request for I/O command execution. The process control object, in turn, creates an initiator I/O process then requests the creation of a target I/O process to execute the operation.

The work associated with the command is performed through a series of process-to-process transactions, consisting of the command descriptor transfer, followed by one or more data_in requests and culminating in the transfer of command status. On completion of the status transfer, both the initiator and target I/O processes are deleted. Delivery of command status to the System Application marks the completion of the I/O command.

6 I/O Control Operations

[Ed. Note - See SCSI-2, section 6.]

6.1 Command Processing Considerations and Exception Conditions

[ed. note - Based on SCSI-2, extensively revised for SCSI-3.]

6.2 Contingent Allegiance

[ed. note - will this be defined for SCSI-3?]

6.3: Auto Contingent Allegiance

[ed, note - New for SCSI-3. Content based on the outcome of the SCSI-3 the queuing model discussions]

6.4 Extended Contingent Allegiance Condition

[ed. note - will this be defined for SCSI-3?]

6.5 Queued I/O Processes

[Ed. note - SCSI-3 Queuing Model goes here]

6.6 Unit Attention Condition

[ed. note - As modified for SCSI-3]

6.7 Multiport SCSI Devices

[ed. note - specifies the behavior of multiport devices.]