# Working Draft

**T11.1 / Project XXXX-X / Rev 1.2**

**July 27, 1999**

# Information Technology - SCSI on Scheduled Transfer Protocol (SST)

Secretariat: National Committee for Information Technology Standardization (NCITS)

This is an internal working draft of T11.1, a Task Group of Technical Committee T11 of Accredited Standards Committee NCITS. As such, this is not a completed standard. The contents are actively being modified by T11.1.



**ABSTRACT**

This document specifies a mapping for transporting SCSI commands and responses using the Scheduled Transfer (ST) protocol (NCITS: T11.1, Project 1245-D: Scheduled Transfer) as the lower level protocol. The ST protocol is defined for a variety of network media including Ethernet, Gigabit Ethernet, ATM, Fibre Channel. The mapping of SCSI storage data onto the ST protocol layer enables storage area networking (SAN) implementations on a wide variety of common network infrastructures.

Contacts:

T11.1 Chairman
Roger Ronald
Power Micro Research
Suite 1100
1411 East Campbell Road
Richardson, TX 75081
Voice: 972-437-9461
FAX: 972-994-0888
E-mail: rronald@pmr.com

T11.1 Vice Chairman
Don Woelz
GENROCO, Inc.
255 Info Highway
Slinger, WI 53086
Voice: 414-644-2505
Fax: 414-644-6667
E-mail: don@genroco.com

Technical Editors
Don Woelz
GENROCO, Inc.
255 Info Highway
Slinger, WI 53086
Voice: 414-644-2505
Fax: 414-644-6667
E-mail: don@genroco.com

Reference number
ISO/IEC 11518-xx : 199x

Other Points of Contact:


| | T11 Chairman | T11 Vice-Chairman | NCITS Secretariat |
|---|---|---|---|
| | Kumar Malavalli | Edward L. Grivna | NCITS Secretariat, ITI |
| | Brocade Communications | Cypress Semiconductor | 1250 Eye Street, NW, Suite 200 |
| | 1901 Guadalupe Parkway | 2401 East 86th Street | Washington, DC 20005 |
| | San Jose, CA 95131 | Bloomington, MN 55425 | |
| Voice: | 408-487-8156 | 612-851-5046 | 202-737-8888 |
| FAX: | 408-524-8601 | 612-851-5087 | 202-638-4922 |
| E-mail: | kumar@brocade.com | elg@cypress.com | ncitssec@itic.nw.dc.us |


T11.1 E-mail Reflector (for HIPPI and ST technical discussions and notification of things on the web site)

| | |
|---|---|
| Internet address for subscribing to the reflector: | Majordomo@nsco.network.com |
| Message should contain a line stating… | subscribe hippi <*your E-mail address*> |
| Internet address for distribution via the HIPPI reflector | hippi@nsco.network.com |

T11 E-mail Reflector (for T11 meeting notices, agendas etc.)

| | |
|---|---|
| Internet address for subscription to the T11 reflector: | Majordomo@nsco.network.com |
| Message should contain a line stating… | subscribe T11 <*your E-mail address*> |
| Internet address for distribution via T11 reflector: | t11@nsco.network.com |

Web sites:

| | |
|---|---|
| HIPPI and ST Standards Activities | http://www.hippi.org/ |
| T11 Activities | http://www.t11.org/ |
| NCITS | http://www.ncits.org/ |

T11 Document Distribution:

Global Engineering
15 Inverness Way East
Englewood, CO 80112-5704
Voice: 303-792-2181 or 800-854-7179
FAX: 303-792-2192

**PATENT STATEMENT**

CAUTION: The developers of this standard have requested that holder's of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.

As of the date of publication of this standard and following calls for the identification of patents that may be required for the implementation of the standard, some such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

# Comments on Rev 1.0

This is a preliminary document undergoing lots of changes. Many of the additions are just placeholders, or are put there to stimulate discussion. Hence, do not assume that the items herein are correct or final – everything is subject to change. This page tries to outline where we are; what has been discussed and semi-approved, and what has been added or changed recently and deserves your special attention. This summary relates to changes since the previous revision. Also, previous open issues are outlined with a single box, new open issues ones are marked with a double bar on the left edge of the box.

Changes are marked with margin bars so that changed paragraphs are easily found, and then highlights mark the specific changes. The list below just describes the major changes, for detail changes please compare this revision to the previous revision. **The major technical changes are printed in bold.**

Please help us in this development process by sending comments, corrections, and suggestions to the Technical Editor, Don Woelz, GENROCO, Inc., at don@genroco.com. If you would like to address the whole group working on this document, send the comment(s) to hippi@nsco.network.com.

1. Changed dates version numbers on cover page.
2. Fixed capitalization in the title of Figure 1.
3. Changed Table 5 to a Figure and reorganized the data.
4. Changed Table 7 to a Figure and reorganized the data.
5. Renumbered Figures and Tables to coincide with changes.

**ANSI ®**
**NCITS xxx-199x**

American National Standard
for Information Technology –

# SCSI on Scheduled Transfer Protocol (SST)

Secretariat

**Information Technology Industry Council (ITI)**

Approved , 199x

**American National Standards Institute, Inc.**


## Abstract

This standard specifies a mapping for transporting SCSI commands and responses using the Scheduled Transfer (ST) protocol (NCITS: T11.1, Project 1245-D: Scheduled Transfer) as the lower level protocol. The ST protocol is defined for a variety of network media including Ethernet, Gigabit Ethernet, ATM, Fibre Channel. The mapping of SCSI storage data onto the ST protocol layer enables storage area networking (SAN) implementations on a wide variety of common network infrastructures.

# American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

**CAUTION NOTICE:** This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

**CAUTION:** The developers of this standard have requested that holder's of patents that may be required for the implementation of the standard disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.

As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, notice of one or more such claims has been received.

By publication of this standard, no position is taken with respect to the validity of this claim or of any rights in connection therewith. The known patent holder(s) has (have), however, filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. Details may be obtained from the publisher.

No further patent search is conducted by the developer or the publisher in respect to any standard in process. No representation is made or implied that this is the only license that may be required to avoid infringement in the use of this standard.

# CONTENTS

**Page**

# TABLES

**Page**



# FIGURES

**PAGE**

**Foreword** (This foreword is not part of American National Standard NCITS xxx-199x.)

This American National Standard specifies a mapping for transporting SCSI commands and responses using the Scheduled Transfer (ST) protocol (NCITS: T11.1, Project 1245-D: Scheduled Transfer) as the lower level protocol. The ST protocol is defined for a variety of network media including Ethernet, Gigabit Ethernet, ATM, Fibre Channel. The mapping of SCSI storage data onto the ST protocol layer enables storage area networking (SAN) implementations on a wide variety of common network infrastructures.

(List of NCITS Committee members, and other active participants, at the time the document is forwarded for public review, will be included by the Technical Editor.)

Technical Committee T11 on Device Level Interfaces, which reviewed this standard, had the following participants:

(List of T11 Committee members, and other active participants, at the time the document is forwarded for public review, will be included by the Technical Editor.)

Task Group T11.1 on the High-Performance Parallel Interface, which developed this standard, had the following participants:

(List of T11.1 Task Group members, and active participants, at the time of document is forwarded for public review will be included by the Technical Editor.)

## Introduction

The SCSI-3 family of standards is developed by NCITS T10 to facilitate the use of the SCSI command sets for different types of devices over a variety of physical interconnects. The master architectural document of the family of standards is X3.270-1996???. Information Technology - SCSI-3 Architecture Model (SAM). The SAM document contains a guide to other SCSI-3 documents.

The SCSI on ST (SST) specification defines a Protocol within the SCSI-3 SAM as shown in Figure 1. The SAM Interconnects to which the SST Protocol may attach are not defined within this specification, but rather, are any Interconnects or other protocols on which the basic ST protocol may operate.

Section 4 describes how SCSI operations are structurally mapped onto ST operations in the SST protocol.

Section 5 defines specific message formats for the SST protocol.

Section 6 defines error recovery procedures for the SST protocol.

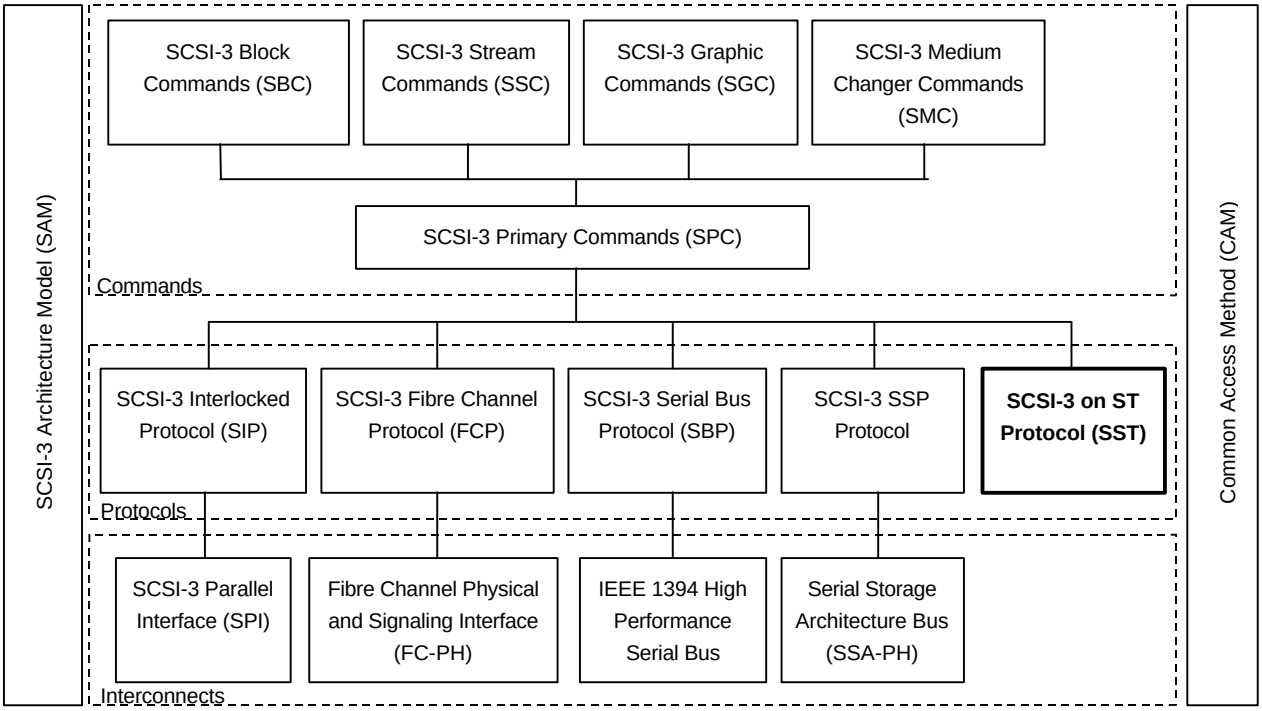



**Figure 1 – The Relationship of SST to SAM**

**American National Standard**
**for Information Technology –**

# SCSI on Scheduled Transfer Protocol (SST)

## 1 Scope

This American National Standard specifies a mapping for transporting SCSI commands and responses using the Scheduled Transfer (ST) protocol (NCITS: T11.1, Project 1245-D: Scheduled Transfer) as the lower level protocol. The ST protocol is defined for a variety of network media including Ethernet, Gigabit Ethernet, ATM, Fibre Channel. The mapping of SCSI storage data onto the ST protocol layer enables storage area networking (SAN) implementations on a wide variety of common network infrastructures.

Specifications are included for:

- Connection management
- Device management
- Task management
- SCSI command service requests using ST Request_To_Send, Request_To_Receive, and NOP operations
- SCSI data delivery requests using the ST Clear_To_Send operation
- SCSI data delivery actions using ST data operations
- SCSI command service responses using the ST Put operation to a Persistent Memory Region
- SCSI I/O operations using ST Read, Write, NOP, and Put Sequences
- Aborting connections and operations

## 2 Normative references

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

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

### 2.1 Approved references

ANSI NCITS 323-1998: Information Systems - High-Performance Parallel Interface - 6400 Mbit/s Physical Layer (HIPPI-6400-PH)

ANSI X3.183-1991 (R1996): Information Systems - High-Performance Parallel Interface - Mechanical, Electrical, and Signaling Protocol Specification (HIPPI-PH)

ANSI X3.210-1998: Information Systems - High-Performance Parallel Interface - Framing Protocol (HIPPI-FP)

ANSI X3.218-1997: Information Systems - High-Performance Parallel Interface - Physical Switch Control (HIPPI-SC)

ANSI X3.222-1993: Information Systems - High-Performance Parallel Interface - Encapsulation of ISO 8802-2 (IEEE Std 802.2) Logical Link Control Protocol Data Units (HIPPI-LE)

ANSI X3.131-1994: Information Systems - Small Computer System Interface – 2 (SCSI-2)

ANSI X3.270-1996: Information Systems - SCSI-3 Architecture Model (SAM)

### 2.2 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the document, or regarding availability, contact the relevant standards body or other organization as indicated. For information about obtaining copies of this document or for more information on the current status of the document, contact National Committee for Information Technology Standards, 1250 Eye Street, NW, Suite 200, Washington, DC 20005, 202-626-5746.

NCITS T11.1, Project 1231: High-Performance Parallel Interface - 6400 Mbit/s Physical Switch Control (HIPPI-6400-SC)

NCITS T11.1, Project 1245-D: Scheduled Transfer (ST)

## 3 Definitions and conventions

### 3.1 Definitions

For the purposes of this standard, the following definitions apply.

**Block:** An ordered set of one or more STUs within a Read, Write, Put, Get or FetchOp Sequence.

**Buffer Index (Bufx):** A 32-bit parameter identifying the starting address of a data buffer.

**Data Channel:** The logical channel that carries the data payload.

**Data operation:** A data transmission consisting of a Schedule Header and up to 4 gigabytes of user payload.

**Destination:** The end device that receives an operation or data.

**exposed:**

*Need to define exposed*

**Initiator:** The end device that starts a sequence of operations. This is typically a host computer system, but may also be a non-transparent translator, bridge, or router.

**Key:** A local identifier used to select and validate operations.

**lower-layer protocol (LLP):** A protocol below the Scheduled Transfer Protocol, e.g., a physical layer.

**Offset:** A parameter specifying the data's starting point relative to the start of a Bufx.

**Opaque data:** Four bytes of Source ULP to Destination ULP peer-to-peer information carried in a Data operation's Schedule Header separately from the data payload.

**Operation:** The procedure defined by the parameters in a Schedule Header, and any payload associated with that Schedule Header. The code in the Schedule Header's "Op" field identifies the operation's name/function.

**optional:** Characteristics that are not required by ST. However, if any optional characteristic is implemented, it shall be implemented as defined in ST.

**persistent:** Memory that is maintained for multiple Put, Get, and FetchOp operations.

**Put:** An operation to write data into a persistent memory region on a remote end device.

**Scheduled Transfer:** An information transfer, normally used for bulk data movement, where the end devices prearrange the transfer using the protocol defined in the ST standard.

**Scheduled Transfer Unit (STU):** The data payload portion of a Data operation. STUs are the basic components of Blocks and are the smallest units transferred.

**Sequence:** An ordered group of ST operations providing a particular function, e.g., Read, Write, Get, etc., between an Initiator and a Responder. The roles of Initiator and Responder are constant for all operations in the Sequence.

**ST Buffer Size:** The unit of memory addressed by ST for Bufx and Offset calculations and expressed as $2^{Bufsize}$ bytes.

**Transaction**: One or more distinct ST Sequences (e.g. Read Sequence, Write Sequence, Persistent Memory Region (PMR) Put Sequence), which are logically associated to implement a complete SCSI function (see section 4.1).

**Transfer:** An ordered set of one or more Blocks within a Scheduled Transfer.

**upper-layer protocol (ULP):** The protocol above ST. A ULP could be implemented in hardware or software, or could be distributed between the two.

**Virtual Connection:** A bi-directional logical connection used for Scheduled Transfers between two end devices. A Virtual Connection contains a logical Control Channel and one or more logical Data Channels in each direction.

### 3.2 Editorial conventions

A number of conditions, Sequence parameters, events, states, or similar terms are printed with the first letter of each word in uppercase and the rest lowercase (e.g., Block, Transfer). Any lowercase uses of these words have the normal technical English meaning.

The word *shall,* when used in this American National standard, states a mandatory rule or requirement. The word *should,* when used in this standard, states a recommendation.

Multiword parameters and field names are joined with an underscore, e.g., D_Port. A parameter associated with a particular end device uses a single letter prefix and a hyphen as a joiner, e.g., I-Key denoting the Initiator's Key.

All numbers are represented as unsigned integers.

Operations contained within <…> are conditional, and may not occur.

#### 3.2.1 Binary notation

Binary notation is used to represent relatively short fields. For example a two-bit field containing the binary value of 10 is shown in binary format as b'10'. An "x" in a bit position indicates a "don't care" value.

#### 3.2.2 Hexadecimal notation

Hexadecimal notation is used to represent some fields. For example a two-byte field containing a binary value of b'1100010000000011' is shown in hexadecimal format as x'C403'.

### 3.3 Acronyms and other abbreviations

| | |
|---|---|
| **FC** | Fibre Channel |
| **FTP** | File Transfer Protocol |
| **HIPPI** | High-Performance Parallel Interface |
| **id** | identifier |
| **IP** | Internet Protocol |
| **IEEE** | Institute of Electrical and Electronic Engineers |
| **KB** | kilobyte (i.e., 1024 bytes) |
| **LAN** | local area network |
| **LLP** | lower-layer protocol |
| **MAC** | Media Access Control |
| **MB** | megabyte (i.e., 1,048,576 bytes) |
| **num** | number, as in B_num |
| **PMR** | Persistent Memory Region |
| **RFC** | Request For Comment |
| **SCSI** | Small Computer System Interconnect |
| **SNAP** | SubNetwork Access Protocol |
| **SPMR** | Status Persistent Memory Region |
| **SST** | SCSI on Scheduled Transfer Protocol |
| **SSTVC** | SST Virtual Connection |
| **ST** | Scheduled Transfer Protocol |
| **STU** | Scheduled Transfer Unit |

**TM** Task Management
**ULA** Universal LAN address
**ULP** upper-layer protocol

# 4 Overview

## 4.1 Structure and Concepts

Scheduled Transfer (ST) is a data transfer protocol which may be implemented on a wide variety of Lower Layer Protocols (LLP), which supports a connection-oriented data transfer model with multiple outstanding data transfers per connection.

Three kinds of functional management are defined by the SST specification:

1. Connection management
2. Device management
3. Task management

For the purposes of describing this SST protocol, we define the term Transaction which consists of one or more distinct ST Sequences (e.g. Read Sequence, Write Sequence, Persistent Memory Region (PMR) Put Sequence), which are logically associated to implement a complete SCSI function from the list above.

The correspondence between SCSI functions, elements of the SST protocol, and elements of the ST protocol on which SST is built is shown in Table 1.

**Table 1 - Functional correspondence between SCSI, SST, and ST operations**

| SCSI | SST | ST |
|---|---|---|
| Command Service Request | Request_To_Send or<br>Request_To_Receive or<br>Request_Zero_Length | Request_To_Send Operation<br>Request_To_Receive Operation<br>NOP Operation |
| Data delivery request | Clear_To_Send operation | Clear_To_Send Operation |
| Data delivery action | Data operation (STU) | Data Operation |
| Command service response | Status Put | Put Operation |
| I/O Operation | Transaction | Read sequence and Put Sequence<br>or<br>Write sequence and Put Sequence<br>Or<br>NOP operation and Put Sequence |
| Task Management Operation | Transaction<br>or<br>Request_Zero_Length<br>or<br>End Sequence | NOP operation and Put Operation<br>or<br>NOP Operation<br>or<br>End Sequence |

# 5 SST Characteristics

## 5.1 Common ST Operation Characteristics

All ST Control operations used in the SST protocol shall have the ST Interrupt flag bit set.

Non-final ST Data operations of Read Sequences and Write Sequences shall have the ST Flag bits Interrupt = b'0', Last = b'0', and Silent = b'1'.

The final ST Data operation of Read Sequences and Write Sequences shall have the ST Flag bits Interrupt = b'1', Last = b'1', and Silent = b'0'.

All ST Data operations for an SST connection shall be sent on the ST Data Channel specified in the SST connection setup protocol (see section 5.3, Connection Management). ST Data Channel flag bits shall be appropriately set to reflect this.

The SST protocol does not use the operation pairs for reliable data movement as described in section 10.2 of the ST protocol specification. Instead, entire SST Transactions are retried as described in section 6.2, Transaction Timeouts.

## 5.3 Connection management

An SST Virtual Connection (SSTVC) is an extension of the basic ST Virtual Connection.

### 5.3.1 Connection Setup

An SSTVC shall be requested by sending an ST Request_Connect operation to the well known port for SST with the Out_Of_Order (O) flag bit cleared, and the SSTVC parameters in the optional payload (see section 6.1, Optional Payload for Connection_Request and Connection_Answer). SST requires that out of order operation not be requested in order to support the Clear_To_Send operation accounting protocol described in section 5.4.1, Underrun Residual Handling.

| *Note: A well-known port for SST must be established.* |
|---|

If the SSTVC request is acceptable to the SSTVC Responder, it shall respond with an ST Connection_Answer operation with the Out_Of_Order (O) flag bit = b'0', and with its SSTVC parameters in the optional payload.

If the SSTVC parameters are unacceptable to the SSTVC Responder, the responder shall reject the SSTVC attempt by sending an ST Connection_Answer operations with the ST Reject Flag = b'1' with an appropriate reason code as specified in section 6.1.6

If the SSTVC parameters are unacceptable to the connection Initiator, the Initiator shall Disconnect the SSTVC using the standard ST Virtual Connection Disconnect Sequence.

If the SSTVC parameters indicate that the connection Initiator can function as a SCSI Initiator and the SSTVC Responder intends to function as a SCSI Target, then the SSTVC Responder shall use an ST Request_Memory_Region operation to request that the SSTVC Initiator expose a Persistent Memory Region, called the Status Persistent Memory Region (SPMR), for the ST Data Channel specified in the Request_Connect Optional Payload. The SPMR size shall be specified in the Request_Connect optional payload. The SSTVC Initiator shall expose the SPMR and respond with an appropriate Memory_Region_Available operation. The initial Offset of the SPMR shall be a multiple of 512 bytes, the SPMR segment size.

When the Connection_Answer operation is received by the SSTVC Initiator, and the connection parameters specify that the SSTVC Responder can function as a SCSI Initiator, and the SSTVC Initiator intends to function as a SCSI Target, then the SSTVC Initiator shall request that the SSTVC Responder expose an SPMR as described above.

An SSTVC endpoint shall not initiate any SCSI operations until the SPMR has been requested and exposed, and the ST Memory_Region_Available operation sent. An SST Initiator shall keep a timer awaiting the ST Request_Memory_Region operation, and if none arrives within the duration of the timer,

disconnect the SSTVC using the standard ST Virtual Connection Disconnect Sequence and retry the SSTVC Sequence.

An SST Target shall respond to any Transaction initiating operation (Request_To_Send, Request_To_Receive or Request_Zero_Length) using an ST Request_Answer with a reason code of SPMR Not Established until the ST Memory_Region_Available operation is received.

*Note: An **SPMR Not Estalibshed** reason code must be assigned.*

If the initial offset returned in the ST Memory_Region_Available operation for the SST SPMR is not a multiple of the Status Segment Size, 512 bytes, an SST Target shall respond to any data transfer request using an ST Request_Answer with a reason code of Status Persistent Memory Region Unaligned.

*Note: An **SPMR Unaligned** reason code must be assigned.*

If an SSTVC endpoint intends to function as a SCSI Initiator and the remote endpoint indicates that an authentication handshake is required, the SCSI Initiator endpoint shall begin the authentication handshake as described in section ???.

*Note: A description of the authentication handshake must be written*

If authentication is required, an SSTVC endpoint shall not initiate any SCSI operations until the authentication handshake has been successfully performed. An SST Target shall respond to any data transfer request (Request_To_Send, Request_To_Receive, or Request_Zero_Length) using an ST Request_Answer with a reason code of Authentication Required until the SST authentication handshake is performed. An SST Target may adopt any suitable policy for reclaiming SSTVC resources if authentication is required and a successful SST authentication Sequence is not performed by an SST Initiator. Specifically, such a policy should protect against denial of service attacks by unauthorized SST connection Initiators.

*Note: An **SPMR Authentication Required** reason code must be assigned.*

### 5.3.2 Connection Disconnect

An SSTVC shall be disconnected using the standard ST Virtual Connection Disconnect Sequence.

### 5.3.3 Virtual Connection Keep Alive

An SST endpoint that behaves as a SCSI Target shall implement the ST Virtual Connection keep-alive Sequence as described in the ST specification (section 10.4). This keep-alive Sequence is the primary mechanism for SST Targets to reclaim resources associated with a broken Virtual Connection.

An SST endpoint that behaves as a SCSI Initiator may implement the ST Virtual Connection keep-alive Sequence as described in the ST specification (section 10.4).

## 5.4 Device management

An application client begins an SST I/O Transaction when it provides to the SST layer a request for an Execute command service. A single request or a list of linked requests may be presented to the software interface. The SST layer then performs the following actions using ST services to perform the SCSI command.

The ST endpoint that is the SCSI Initiator for the command starts a Transaction by sending an ST Request_To_Send, ST Request_To_Receive, or SST Request_Zero_Length operation to begin an SST device management Sequence (see section 6.2, ST NOP Operations in SST). A Request_To_Send is sent if the SCSI command involves data transfer from the SCSI Initiator to the SCSI Target. A Request_To_Receive is sent if the SCSI command involves data transfer from the SCSI Target to the SCSI Initiator. An SST Request_Zero_Length is sent if the command involves no data transfer.

The ST Request_To_Send, ST Request_To_Receive or SST Request_Zero_Length operation has an optional payload including some command control flags,

addressing information, and the SCSI Command Descriptor Block (CDB).

This ST Request_To_Send, ST Request_To_Receive or SST Request_Zero_Length operation is the Execute Command service request and starts the I/O Operation. The Transaction that is started is identified by the ST Initiator Sequence Identifier qualified by the ST Virtual Connection. The ST Initiator Sequence Identifier used on the ST Request_To_Send, ST Request_To_Receive, or SST Request_Zero_Length operation which begins an SST Transaction is called the Initiator Transaction Identifier and is used appropriately on all ST operations associated with the Transaction.

When the Target for the command has completed the interpretation of the command, has determined that a data transfer is required, and is prepared to request the data delivery service, it shall indicate this to the Initiator, and data is transferred as described below. Note that the amount of data transferred is the minimum of:

- The transfer length specified in the ST Request_To_Send or Request_To_Receive operation which initiated the Transaction; or,
- The amount of data the Target requires to be transferred as specified in the SCSI CDB.

If the Transaction was initiated with an ST Request_To_Send operation, the Target shall send one or more ST Clear_To_Send operations for the amount of data that it is prepared to receive.

The Initiator shall then respond with one or more ST Data operations to satisfy the outstanding Clear_To_Send operations received from the Target. The last ST Data operation in each Sequence shall set ST Interrupt = b'1' and Last = b'1', and ST Silent = b'0'.

When the Target is prepared to receive additional data, it may send one or more additional ST Clear_To_Send operations to which the Initiator will respond with additional Data operations until the required amount of data has been transferred.

If the Transaction was initiated with an ST Request_To_Receive operation, the Target shall respond to the ST Request_To_Receive operation with an ST Request_To_Send operation. Data is then transferred from the Target to the Initiator as above, except that the Initiator will send ST Clear_To_Send operations and the Target will send ST Data operations.

After all of the data has been transferred, the Target shall transmit the Execute Command service response by sending an ST Put Sequence to the SPMR defined by the connection setup protocol at an address (Bufx/Offset) computed from the Initiator Transaction Identifier for the Transaction. This ST Put Sequence shall contain the SCSI status and, if an unusual condition has been detected, the SCSI REQUEST SENSE information and the SST Response information that describes the condition. The SST Status Put shall terminate the command. The SCSI logical unit determines whether additional commands will be performed in the SST I/O Operation. If this is the last or only command executed in the SST I/O Operation, the SST I/O Operation and the Transaction shall be terminated.

When the command is completed, returned information is used to prepare and return the Execute Command service confirmation information to the software that requested the operation. The returned status shall indicate whether or not the command was successful. The successful completion of the command indicates that the SCSI device performed the desired operations with the transferred data and that the information was successfully transferred to or from the SCSI Initiator.

If the command is linked to another command, the ST Put payload shall contain the proper status indicating that another command will be executed. The Initiator shall continue the same Transaction with an ST Request_To_Send, ST Request_To_Receive, or SST Request_Zero_Length operation beginning the next SCSI command. All SCSI commands linked in the SST I/O Operation except the last

shall be executed in the manner described above.

Note that when an SST Transaction executes more than one linked SCSI command, the same Initiator Transaction Identifier shall be used for all ST Sequences in the SST Transaction.

The number of SST I/O Operations that may be active at one time depends on the queuing capabilities of the particular SCSI devices and the number of concurrent exchanges supported by the ST endpoints.

A brief way of summarizing this is that an SST Transaction consists of:

- an ST Read, Write or NOP Sequence; followed by
- an ST Put Sequence to the SPMR containing status

for each command in a single SCSI Execute service request. Note that commands may be linked.

### 5.4.1 Underrun Residual Handling

If the amount of data the Target requires to be transferred as specified in the SCSI CDB is less than the length of the ST data transfer specified in the ST Request_To_Send or Request_To_Receive operation, either as a result of a successful or unsuccessful SCSI command completion, this is called an Underrun Residual.

If an ST Write Sequence in an SST Transaction results in an Underrun Residual, the SST Target shall receive all data it has requested with outstanding ST Clear_To_Send operations before performing the SST Status Put operation which signifies the completion of the SCSI command.

If an ST Read Sequence in an SST Transaction results in an Underrun Residual and the SST Initiator has sent ST Clear_To_Send operations for this ST Read Sequence for data beyond the final block in which the SST Target has transferred data, the SST Initiator shall ensure that all Clear_To_Send operations have been received by the SST Target by sending an ST End operation with an optional payload that indicates the highest block number of Clear_To_Send operations which have been sent (see section 6.5). After all outstanding Clear_To_Send operations have been received, the SST Target shall respond to the ST End operation with an ST End_Ack operation as described by the ST protocol. In order to support this Clear_To_Send accounting protocol, an SST Initiator shall only send Clear_To_Send operations for a contiguous sequence of Blocks for SST Read Transactions.

### 5.4.2 Third Party SCSI Commands

Certain third-party SCSI commands and parameters specify a 64-bit field that is defined to access other SCSI devices addressable from that port. These commands include COPY, RESERVE, and several others.

The ST protocol does not specify any form of addressing. However, it is usually run on an LLP that does specify an address format. Therefore, the addressing formats for third party SCSI commands in the SST protocol are a function of the LLP addressing format. Addressing formats are beyond the scope of this specification.

## 5.5 Task Management

An application client requests a Task Management (TM) function when a task or some group of tasks must be aborted or terminated.

SCSI TM functions are mapped onto SST and the underlying ST protocol as shown in Table 2.

**Table 2 - SCSI Task Management Function Mapping**

| SCSI | SST | ST | Required |
|---|---|---|---|
| ABORT TASK | End Sequence | End Sequence | Y |
| TERMINATE TASK | Request_Zero_Length with Terminate Task bit set | NOP Operation | N |
| TARGET RESET | Transaction with Target Reset bit set | NOP Operation and Put Operation | Y |
| ABORT TASK SET | Transaction with Abort Task Set bit set | NOP Operation and Put Operation | Y |
| CLEAR TASK SET | Transaction with Clear Task Set bit set | NOP Operation and Put Operation | Y |
| CLEAR ACA | Transaction with Clear Auto Contingent Allegiance bit set | NOP Operation and Put Operation | Y (if ACA is supported) |

The SST Request_Zero_Length operation sent to initiate a TARGET RESET, ABORT TASK SET, CLEAR TASK SET or CLEAR ACA TM function shall be sent as the first operation in a new Transaction. As with all other Transactions, the target shall respond to TARGET RESET, ABORT TASK SET, CLEAR TASK SET and CLEAR ACA TM functions with an SST Status Put using the Initiator Transaction Identifier specified in the Request_Zero_Length operation that requested the TM function.

The SST Request_Zero_Length operation sent to initiate a TERMINATE TASK TM function shall be sent using the Initiator Transaction Identifier of the task to be terminated. Unlike other TM functions initiated with an SST Request_Zero_Length operation, the target will not respond to a TERMINATE TASK TM request with an SST Status Put operation specifically for the TERMINATE TASK TM function. However, when the requested task is terminated, the target shall, as always, perform an SST Status Put operation with the status of the terminated task.

### 5.5.1 Abort Task

The SCSI Abort Task TM function may be used to terminate, prematurely, an outstanding SCSI function. The SST Abort Task protocol used to implement the SCSI Abort Task TM function ensures that in addition to aborting the SCSI function, all outstanding ST Operations for that SST Transaction have either reached their destination or been discarded before the SST Abort Task protocol is complete.

Once the SST Abort Task protocol is complete, both the SST Initiator and SST Target are free to reuse the ST Sequence Identifiers, including the SST Transaction Identifier, associated with the SST Transaction.

For the purpose of the Abort Task TM function, the Abort Task Timeout shall be selected as the sum of the maximum time required for a target to perform internal functions associated with aborting the Transaction, and the maximum lifetime of an ST operation on the network. Generally, the Abort Task Timeout can be the same as the Transaction Timeout for a Transaction, since it should not take longer to abort a Transaction than it would take to complete the Transaction normally.

To abort an SST Write Transaction, the Initiator shall:

a - stop sending data
b - send an ST End for the Transaction to abort
c - For every block for which a Clear_To_Send has been received, ensure that an in-order STU with ST Flags Last=1, Interrupt=1 and Silent=0 has been sent
d - Wait for an End_Ack for the Transaction being aborted
e - If an End_Ack is not received within the Abort Task Timeout, determine the viability of the Virtual Connection and:

- If the Virtual Connection is not viable, declare the Virtual Connection closed; otherwise,
- Retry steps b through d an appropriate number of times after which the Write Transaction shall be declared aborted

And the Target shall:

a - if a Transaction with the Initiator Transaction Identifier specified in the End is not in progress, send End_Ack
b - otherwise:
  - For each Clear_To_Send operation sent, wait for a STU with the ST flag bit Last = b'1'
  - send End_Ack

To abort an SST Read Transaction, the Initiator shall:

a - Stop sending Clear_To_Send operations
b - Send an End operation indicating block number of the last Clear_To_Send operation in the optional payload (section 6.5, Optional Payload For ST End Operations)
c - Wait for a STU with the ST Flag Last=1 for every Clear_To_Send operation sent, and an End_Ack operation for the Transaction being aborted
d - If the operations awaited in c are not received within the Abort Task Timeout, determine the viability of the Virtual Connection and:
  - if the Virtual Connection is not viable, declare the Virtual Connection closed
  - otherwise, retry steps b and c an appropriate number of times, after which, the Read Transaction shall be declared aborted

And the Target shall:

a - if a Transaction with the Initiator Transaction Identifier specified in the End is not in progress, send End_Ack
b - Otherwise:
  - stop sending data
  - for each Clear_To_Send operation reported sent by the Optional Payload of the End operation, ensure that an in-order STU with ST Flags Last=1, Interrupt=1 and Silent=0 has been sent
  - send End_Ack

To abort an SST Request_Zero_Length Transaction, the Initiator shall:

a - send End
b - wait for an End_Ack operation for the Transaction being aborted

And the Target shall:

a - send End_Ack

# 6 SST Protocol Operation Formats

## 6.1 Optional Payload for Connection_Request and Connection_Answer

Figure 2 shows the format of the Optional Payload data for the Connection Request and Connection Answer operations.

**Figure 2 – Optional Payload for Connection_Request and Connection_Answer**

| | | | Byte |
|---|---|---|---|
| ST_OC | LENGTH | MAX_TARGET | 00-03 |
| FLAGS | | | 04-07 |
| SPMR_SIZE | | | 08-11 |

### 6.1.1 ST_OPTION_CODE

ST_OPTION_CODE = x'03', indicating that the ST optional payload field is valid and contains a ULP parameter.

### 6.1.2 LENGTH

LENGTH = x'0C', indicating that this ST optional payload field is 12 bytes long.

### 6.1.3 MAX_TARGET

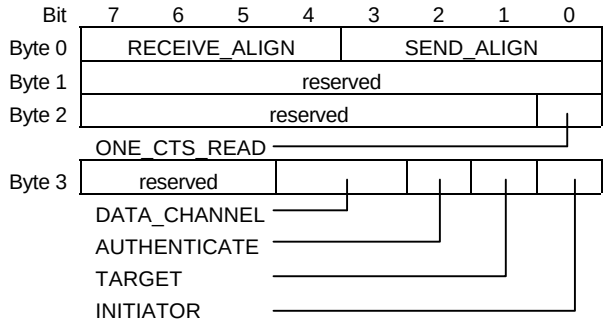
Maximum supported value of the TARGET field (see section 5.3.3) in the Optional Payload for ST Request_To_Send, an ST Request_To_Receive, or and SST

Return_Zero_Length operations. Note that the maximum number of targets is MAX_TARGET + 1.

### 6.1.4 FLAGS

The FLAGS field contains a number of supported and required feature flags as shown in Figure 3 and described in the following text.

**Figure 3 – Optional Payload Flags for Connection Operations**




RECEIVE_ALIGN is the log, base 2, of the minimum required alignment of the offset parameter of any ST Data operations sent to the SST Virtual Connection endpoint specifying the RECEIVE_ALIGN value. This necessarily implies that a SST endpoint will never perform a Clear_To_Send operation with an initial offset that does not conform to the RECEIVE_ALIGN value it specified during the SST Connection setup.

SEND_ALIGN is the log, base 2, of the minimum required alignment of the offset parameter of any ST Clear_To_Send operations sent to the SST Virtual Connection endpoint specifying the SEND_ALIGN value.

ONE_CTS_READ = b'1' indicates that an SST Target requires an Initiator to send only a single ST Clear_To_Send operation to cover the entire data transfer of an SST Read Transaction.

DATA_CHANNEL indicates the ST Data Channel for which to request the SST Status Persistent Memory Region, and on which all SST Data operations, including SST Status Put Sequences, shall be performed.

AUTHENTICATE = b'1' indicates that an authentication handshake must be performed during an SST connection setup before data may be transferred.

TARGET = b'1' indicates that the SST endpoint is capable of performing SCSI Target functions.

INITIATOR = b'1' indicates that the SST endpoint is capable of performing SCSI Initiator functions.

### 6.1.5 SPMR_SIZE

This field indicates the number of 512-byte Status Persistent Memory Region segments an SST Target should request that an SST Initiator should expose.

In other words, the size of the Status Persistent Memory Region requested shall be:

SPMR_SIZE * 512 bytes

### 6.1.6 SST Virtual Connection Reject Reason Codes

If the parameters of an SST Virtual Connection are unacceptable, the Virtual Connection request shall be rejected or closed with an appropriate reason code as shown in Table 3.

**Table 3 – SST Connection Reject Reason Codes**

| Definition | Value |
|---|---|
| Single CTS Read unsupported | TBD |
| Target function unsupported | TBD |
| Busy (no resources) | TBD |

## 6.2 ST NOP Operations in SST

SST distinguishes different uses of the ST NOP operation using the 16-bit Param field as an operation code as shown in Table 4. ST considers this field to be opaque.

**Table 4 – SST NOP Operations**

| Definition | Value |
|---|---|
| Request_Zero_Length | x'0' |

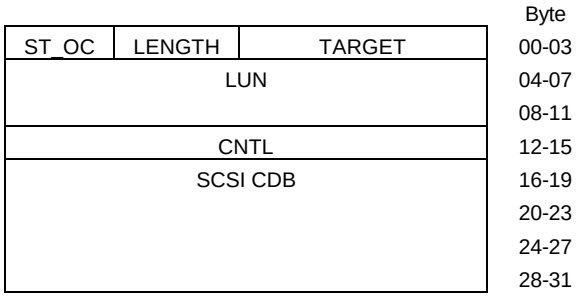
The 32-bit S_id field of an SST Request_Zero_Length operation contains the Initiator Transaction Identifier. ST considers this field to be opaque.

The remainder of the NOP fields (which ST considers opaque) in the SST Request_Zero_Length operation are unused. The Optional Payload of the Request_Zero_Length operation contains the Optional Payload for SST command operations as described in section 6.3.

### 6.3 Optional Payload for SST Command Operations

The Optional Payload for ST Request_To_Send, ST Request_To_Receive, and SST Request_Zero_Length operations is organized as shown in Figure 4.

**Figure 4 – Optional Payload for SST Command Operations**




#### 6.3.1 ST_OPTION_CODE

SST_OPTION_CODE = x'03' indicates that the ST Optional Payload field is valid and contains a ULP parameter

#### 6.3.2 LENGTH

LENGTH = x'20' indicates that the ST optional payload field is 32 bytes long.

#### 6.3.3 TARGET

Selects one of a set of individual SCSI Targets addressable through a single SST Virtual Connection. The maximum allowable value of the TARGET field is established in the SST connection setup protocol.

#### 6.3.4 LUN

The SST Logical Unit Number (LUN) is the address of the desired logical unit in the attached subsystem. The LUN field is specified by X3.230.

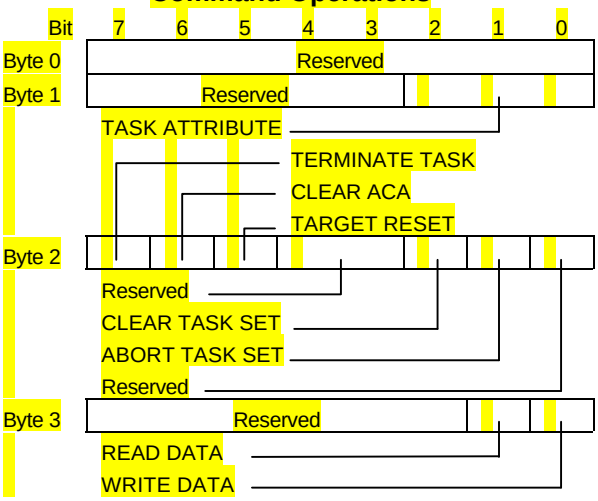
Each SST Target shall accept an INQUIRY command to the logical unit addressed by LUN = x'0000 0000 0000 0000'. Using the SCSI INQUIRY information, the Initiator can determine the SCSI device type, manufacturer, and model of the logical unit. If the logical unit at LUN 0 specifies a device model that has a defined addressing structure, the Initiator can use that information to determine what other logical units are implemented in the Target. The structure of the LUN field is not specified by the SST protocol. The structure is specified by the SCSI device model and may be vendor unique for device models that do not have a defined address structure.

If the LUN address locates a valid logical unit, the command shall be executed according to standard SCSI behavior. Behavior may include successful execution of the command, presentation of errors associated with the command, or rejection of the command. If the addressed logical unit does not exist, the responses shall follow the SCSI-3 rules for selection of invalid logical units.

#### 6.3.5 CNTL

The CNTL field contains a number of control flags and control bits as shown in Figure 5.

**Figure 5 – CNTL field definitions for SST Command Operations**




### 6.3.5.1 Task Codes, Byte 1

TASK ATTRIBUTE shall be selected as defined in Table 5 (refer to X3.270).

**Table 5 – TASK ATTRIBUTE definitions for SST Command Operations**

| Value, bits 2-0 | TASK ATTRIBUTE |
|---|---|
| b'000' | SIMPLE_Q |
| b'001' | HEAD_OF_Q |
| b'010' | ORDERED_Q |
| b'100' | ACA_Q |
| b'101' | UNTAGGED |
| others | reserved |

SIMPLE_Q requests that the task be managed according to the rules for a SIMPLE task attribute.

HEAD_OF_Q requests that the task be managed according to the rules for a HEAD OF QUEUE task attribute.

ORDERED_Q requests that the task be managed according to the rules for an ORDERED task attribute.

ACA_Q requests that the task be managed according to the rules for an Automatic Contingent Allegiance (ACA) task attribute.

UNTAGGED requests that the task be managed according to the rules for an untagged task. Only one untagged task can exist for each logical unit identifier / Initiator pair. Requesting a second untagged command for the same logical unit identifier / Initiator pair shall be treated as an overlapped command. An untagged task is scheduled according to the rules for a SIMPLE task attribute.

### 6.3.5.2 Task Management Flags, Byte 2

If any TM flag is set to b'1', the CDB and CDB related CNTL flags (task codes and execution management codes) are not valid and shall be ignored. No more than one task management flag shall be set to b'1' in any SST Request_Zero_Length Transaction. No TM flags shall be set for an ST Request_To_Send or ST Request_To_Receive operation.

TERMINATE TASK = b'1' requests that the specified task be terminated without corrupting the medium. This bit is optional for SST Targets. The TERMINATE TASK operation shall be sent by the initiator using the Initiator Transaction Identifier for the task to be terminated.

CLEAR ACA = b'1' causes the ACA condition to be cleared. When the task manager clears the Auto Contingent Allegiance condition, any task within that task set may be completed subject to the rules for task management specified by X3.270.

The use of the ACA bit in the CDB control field and the implementation of ACA are described in X3.270.

If the ACA bit in the CDB control field is set to b'0', the automatic sense operation performed by the presentation of the SST Status Put operation shall clear the ACA condition. Depending on the MODE SELECT parameters that have been established, additional SST I/O operations may have to be aborted by the recovery abort

TARGET RESET = b'1' performs a reset to the SCSI device as defined in X3.270. TARGET RESET resets all tasks for all Initiators and

resets all internal states of the Target to their initial power-on and default values. A unit attention condition is created for all Initiators. The Initiator and Target clear all resources.

CLEAR TASK SET shall cause all tasks from all Initiators in the specified task set to be aborted as defined in X3.270. A unit attention condition is created for all Initiators other than the Initiator that sent the CLEAR TASK SET that had tasks in the task set.

ABORT TASK SET shall cause all tasks in the task set from the Initiator requesting the ABORT TASK SET to be aborted as defined in X3.270.

#### 6.3.5.3 Execution management codes, Byte 3

READ DATA = b'1' shall only be set on an ST Request_To_Receive operation and specifies that the Initiator expects data for the task to be in the direction opposite to the direction of the ST Request_To_Receive operation. This is a SCSI Read operation.

WRITE DATA = b'1' shall only be set on an ST Request_To_Send operation and specifies that the Initiator expects data for the task to be in the same direction as the ST Request_To_Send operation. This is a SCSI Write operation.

If READ DATA = b'0' and WRITE DATA = b'0', there shall be no data and the ST Request_To_Send or ST Request_To_Receive transfer length shall be 0.

### 6.3.6 CDB

The CDB field shall contain the actual CDB to be interpreted by the addressed logical unit. The maximum CDB length is 16 bytes. The CDB is not valid and is ignored if any task management flag is set to 1. The command byte of the CDB shall be located at the first byte transmitted of the CDB field. The first byte of the CDB defines the length and basic format of the remainder of the CDB as described in X3.270. Bytes beyond the last byte of the CDB are not defined by SST, shall be ignored by the Target, and may have any value.

## 6.4 SST Status Put Sequence

The SST Status Put Sequence shall be directed to an address (Bufx/Offset) which is computed as:

> Status Put Bufx = address / Initiator bufsize
>
> Status Put Offset = address % Initiator bufsize

Where:

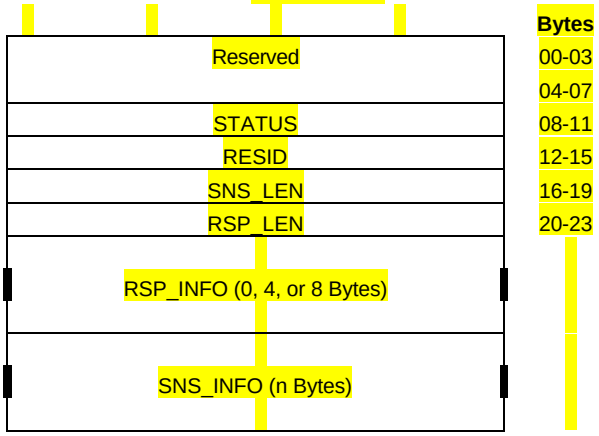
> address = (SPMR initial Bufx * Initiator bufsize + SPMR initial offset) + 512 * Transaction Initiator Identifier

The Transaction Initiator Identifier shall be sent in the ST S_id field (opaque in the ST specification) of the ST Data operations of the SST Status Put. This is a departure from all other ST operations for a Transaction where the Target places the Initiator Transaction Identifier in the ST D_id field. The D_id field of SST Status Put operations contains the Responder Identifier for the Persistent Memory Region identified in the ST Memory_Region_Available operation received as part of the SST connection setup protocol.

The SST Status Put Sequence shall consist of a single ST Data STU with the entire SST Status Put payload. The SST Status Put Data STU shall have ST Interrupt = b'1' and Last = b'1', and ST Silent = b'0'.

The payload of the SST Status Put Sequence is shown in Figure 6.

**Figure 6 - Payload Parameters for an SST Status Put**

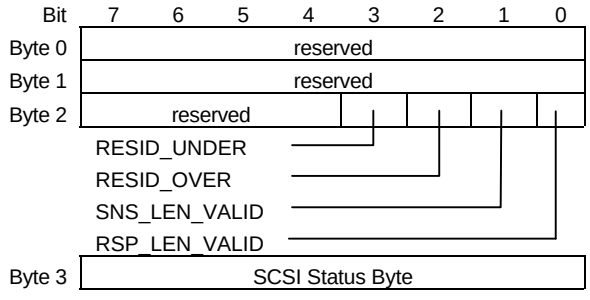



### 6.4.1 STATUS

STATUS = x'00 00 00 00' normally upon successful completion of an SST I/O Operation. Even if command linking is being performed, an SST Status Put shall be performed for each command executed. A value of x'00 00 00 00' means no error, and in addition indicates that no other information is present in the SST Status Put. For linked commands, INTERMEDIATE status or INTERMEDIATE CONDITION MET status indicates successful completion of a command with no other information valid if all other fields are 0.

Figure 7 shows the format of the STATUS field.

**Figure 7 – STATUS field definitions for SST Status PUT**




RESID_UNDER = b'1' indicates that the RESID field is valid and contains the count of bytes that were expected to be transferred, but were not transferred.

RESID_OVER = b'1' indicates that the RESID field is valid and contains the count of bytes that could not be transferred because the ST transfer length was not sufficient.

SNS_LEN_VALID = b'1' indicates that the SNS_LEN field is valid and contains the count of bytes in the SNS_INFO field.

RSP_LEN_VALID = 'b'1' indicates that the RSP_LEN field is valid and contains the count of bytes in the RSP_INFO field.

Byte 3 contains the status byte from the SCSI logical unit. The status byte codes are defined by X3.270.

### 6.4.2 RESID

If RESID_UNDER = b'1' or RESID_OVER = b'1', the RESID field contains a count of the number of residual data bytes that were not transferred for this SCSI command. Upon successful completion of an SST I/O Operation, the residual value is normally 0 and the RESID value is not valid. Devices having indeterminate data lengths may have a nonzero residual byte count after completing valid operations. Targets are not required to verify that the data length implied by the contents of the CDB will create an overrun or underrun before beginning execution of an SCSI command.

If RESID_UNDER = b'1', a transfer that did not fill the buffer to the expected displacement specified by the ST transfer length was performed and the value of RESID is a number equal to:

> ST transfer length - highest offset of any byte transmitted

A condition of RESID_UNDER may not be an error for some devices and some commands.

If RESID_OVER = b'1', the transfer was truncated because the data transfer required by the SCSI command extended beyond the transfer length specified in the ST Request_to_Send or Request_to_Receive operations. Those bytes that could be transferred without violating the ST transfer

length value may be transferred. The RESID is a number equal to:

(Transfer length required by command) - ST transfer length

If a condition of RESID_OVER is detected, the termination state of the SST I/O operation is not certain. Data may or may not have been transferred and the SCSI status byte may or may not provide correct command completion information.

If the RESID_UNDER and the RESID_OVER bits are 0, the RESID field is not meaningful and may contain any value.

### 6.4.3 SNS_LEN

If SNS_LEN_VALID = b'1', the SNS_LEN field specifies the number of valid bytes of SNS_INFO.

If SNS_LEN_VALID = b'0', the SNS_LEN field is not valid and no SNS_INFO is provided.

The SNS_LEN field shall always be included in the SST Status Put.

### 6.4.4 RSP_LEN

If RSP_LEN_VALID = b'1', the RSP_LEN field specifies the number of valid bytes of RSP_INFO. The number of valid bytes shall be x'00 00 00 00', x'00 00 00 04', or x'00 00 00 08'. Other values of length are reserved for future standardization.

RSP_LEN = b'00 00 00 00' specifies that no bytes of response information are being provided.

If RSP_LEN_VALID = b'0', the RSP_LEN field is not valid and no RSP_INFO is provided.

The RSP_LEN field shall always be included in the SST Status Put.

### 6.4.5 RSP_INFO

The RSP_INFO field contains information describing only the protocol failures detected during the execution of an SST I/O operation. The RSP_INFO does not contain SCSI logical unit error information since that is contained in the SNS_INFO field as proscribed in 5.4.6. The RSP_INFO field shall contain valid information if the Target detects any of the conditions indicated by a RSP_CODE. The format of the RSP_INFO field is shown in Figure 8.

**Figure 8 – RSP_INFO field definitions for SST Status PUT**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 0 | reserved | | | | | | | |
| Byte 1 | reserved | | | | | | | |
| Byte 2 | reserved | | | | | | | |
| Byte 3 | RSP_CODE | | | | | | | |

The valid RSP_CODE values are shown in Table 6.

**Table 6 – RSP_CODE definitions for SST Status PUT**

| RSP_CODE definition | Value |
|---|---|
| No failure or Task Management function complete | x'00' |
| reserved | x'01' |
| SST CTS or RTS payload fields invalid | x'02' |
| reserved | x'03 |
| Task Management Function Not Supported | x'04 |
| Task Management Function Failed | x'05' |
| Nonexistent Target | x'06' |
| Busy (no resources) | x'07' |
| reserved | X'08' – x'FF' |

### 6.4.6 SNS_INFO

The SNS_INFO field contains the information specified by X3.131 for presentation by the REQUEST SENSE command. The proper SNS_INFO shall be presented when the SCSI status byte of CHECK CONDITION or COMMAND TERMINATED is presented as

specified by X3.270. SST implements the autosense mechanism as described in X3.270.

### 6.5 Optional Payload for ST End Operations

When the ST End or End_Ack operation is required by the SST protocol to specify the block number of the last ST Clear_To_Send operation sent, the ST End or End_Ack operation's optional payload shall be as shown in Figure 9.

**Figure 9 – Optional Payload for ST End Operations**

| | | | Byte |
|---|---|---|---|
| ST_OC | LENGTH | reserved | 00-03 |
| B_NUM | | | 04-07 |

#### 6.5.1 ST_OPTION_CODE

ST_OPTION_CODE = x'03' indicates that the ST optional payload field is valid and contains a ULP parameter.

#### 6.5.2 LENGTH

LENGTH = x'08' indicates that the ST optional payload field is 8 bytes long.

#### 6.5.2 B_NUM

Indicates the ST Block number of the last ST Clear_To_Send operation sent by the data sink.

Since the SST protocol requires in-order request of ST data Blocks, the ST Block number of the last ST Clear_To_Send operation permits the data source to ensure that all ST Clear_To_Send operations for a Transaction have been received before reusing the ST Sequence Identifiers associated with the Transaction. This permits the endpoints to avoid aliasing of ST Clear_To_Send operations across Transactions when Sequence Identifiers are reused.

## 7 ERROR RECOVERY PROCEDURES

### 7.1 Virtual Connection Keep Alive

An SST endpoint Target shall implement ST Virtual Connection keep-alive Sequences as described in the ST specification (section 10.4). An SST endpoint Initiator may implement ST Virtual Connection keep-alive Sequences as described in the ST specification.

If an ST Virtual Connection is discovered to be dead by a failure of the keep-alive test, all SST Transactions on the Virtual Connection shall be declared dead and their resources reclaimed, as well as the resources for the Virtual Connection itself.

Determining the viability of a Virtual Connection consists of performing a Request_State/Request_State_Response Sequence with an appropriate number of retries.

For a very low loss medium, a single retry would be an appropriate default for any step in the SST Abort Task protocol that requires retries.

### 7.2 Transaction Timeouts

An SST Initiator shall maintain timeouts on all outstanding device and task management operations from the time a Request_To_Send, Request_To_Receive or Request_Zero_Length operation is sent until the final Data and SST Status Put operations are received.

The Transaction Timeout shall be at least the sum of:

- the maximum time required to perform the SCSI command and transfer the associated data
- the maximum time an ST operation can remain in the network

The actual determination of the Transaction Timeout value is beyond the scope of this document.

This timeout ensures that no ST operations remain in the network or will be subsequently generated by an SST Target for an SST Transaction which guarantees that an SST Initiator may reuse the SST Transaction Identifier (ST I-id) associated with the SST Transaction immediately.

If a Transaction Timeout expires, the Initiator shall first determine the viability of the SST Virtual Connection using an ST Request_State / Request_State_Response Sequence. If a Request_State_Response operation is not received within an appropriate period of time, the Request_State / Request_State_Response Sequence shall be retried an appropriate number of times after which the Virtual Connection shall be declared closed and its resources reclaimed (refer to section 6.1).

If a Request_State_Response is received, the operation that timed out shall be aborted with the SST Abort Task protocol as described in section 4.5.1, Abort Task.

An SST Target shall not maintain timeouts on outstanding Transactions. An SST Target's resources for a Transaction shall only be reclaimed when:

- the Target completes the Transaction
- the Initiator ends the Transaction with the SST Abort Task protocol
- the SST Virtual Connection is ended
- target – see Steph!!!!

The requirement that the SST Initiator perform the SST Abort Task protocol for timed out operations ensures that an SST Target can reclaim resources associated with a failed Transaction.