Working Draft

T10
Project xxxxD

Revision 1
March 11, 1997

Information technology
SCSI Transport *via* SBP-2

T10 Technical Editor:

Peter Johansson
Congruent Software, Inc.
3998 Whittle Avenue
Oakland, CA 94602
USA

(510) 531-5472
(510) 531-2942 FAX

pjohansson@aol.com

Reference numbers
ISO/IEC xxxxx:199x
ANSI NCITS.xxx-199x

Printed March 11, 1997

Points of contact

T10 Chair:

John B. Lohmeyer
Symbios Logic, Inc.
4420 Arrows West Drive
Colorado Springs, CO 80907
USA

(719) 533-7560
(719) 533-7036 FAX
john.lohmeyer@symbios.com

T10 Vice-chair:

Lawrence J. Lamers
Adaptec, Inc.
691 South Milpitas Boulevard
Milpitas, CA 95035

(408) 957-7817
(408) 957-7193 FAX
ljlamers@aol.com

NCITS Secretariat:

NCITS Secretariat
1250 I Street NW, Suite 200
Washington, DC 2000
USA

(202) 737-8888
(202) 638-4922 FAX

T10 Bulletin board: (719) 533-7950

T10 FTP: ftp.symbios.com/pub/standards/io/x3t10

T10 Home page: http://www.symbios.com/x3t10

T10 Reflector: scsi@symbios.com
majordomo@symbios.com (to subscribe)

IEEE 1394 Reflector: p1394@sun.com
bob.snively@sun.com (to subscribe)

Document distribution:

Global Engineering
15 Inverness Way East
Englewood, CO 80112-5704
USA

(800) 854-7179
(303) 792-2181
(303) 792-2192 FAX

ANSI®
NCITS.xxx-199x

American National Standard
for Information Systems –

SCSI Transport *via* SBP-2

Secretariat

Information Technology Industry Council

Not yet approved

American National Standards Institute, Inc.

Abstract

This standard specifies how to use Serial Bus Protocol 2 (SBP-2) for the transport of SCSI commands, data and status between devices connected by Serial Bus, a memory-mapped split-transaction bus defined by IEEE Std 1394-1995.

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

Published by

American National Standards Institute
1430 Broadway, New York, NY 10018

Printed in the United States of America

Contents

Page

Tables

Figures

Annexes

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

Serial Bus Protocol 2 (SBP-2) defines a transport protocol within the domain of IEEE Std 1394-1995 that is designed to permit efficient, peer-to-peer operation of input output devices (disks, tapes, printers, ***etc***.) by initiator(s) such as operating systems or embedded applications. SBP-2 does not specify the command sets used by the devices—only the mechanisms by which commands, data and status are transported between initiator(s) and target(s). This standard specifies how SBP-2 may be used to transport the SCSI family of commands and specifies a standard format for the return of SCSI status and sense data.

This standard was developed by T10 during 1997 in parallel with the completion of the Serial Bus Protocol 2 standard.

There is one annex in this standard, Annex A, which is informative and is not considered part of this standard.

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the NCITS Secretariat, Information Technology Industry Council, 1250 I Street NW, Suite 200, Washington, DC 20005-3922.

This standard was processed and approved for submittal to ANSI by National Committee for Information Technology Standardization (NCITS). Committee approval of this standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, NCITS had the following members:

James D. Converse, Chair
Donald C. Loughry, Vice-chair
Joanne M. Flanagan, Secretary

Organization Represented	*Name of Representative*
American Nuclear Society	Geraldine C. Main
AMP, Inc.	Edward Kelly
Apple Computer	Karen Higginbottom
Association of the Institute for Certification of Professionals	Kenneth Zemrowski
AT&T/NCR	Thomas W. Kern
Boeing Company	Catherine Howells
Bull HN Information Systems, Inc.	William George
Compaq Computer Corporation	James Barnes
Digital Equipment Corporation	Delbert Shoemaker
Eastman Kodak	James D. Converse
GUIDE International	Frank Kirshenbaum
Hewlett-Packard	Donald C. Loughry
Hitachi America, Ltd.	John Neumann
Hughes Aircraft Company	Harold L. Zebrack
IBM Corporation	Joel Urman
National Communication Systems	Dennis Bodson
National Institute of Standards and Technology	Robert E. Roundtree
Northern Telecom, Inc.	Mel Woinsky
Neville & Associates	Carlton Neville
Recognition Technology Users Association	Herbert P. Schantz
Share, Inc.	Gary Ainsworth
Sony Corporation	Michael Deese
Storage Technology Corporation	Joseph S. Zajaczkowski
Sun Microsystems	Scott Jameson
3M Company	Eddie T. Morioka

Unisys Corporation ..... John L. Hill
US Department of Defense..... William C. Rinehuls
US Department of Energy..... Alton Cox
US General Services Administration..... Douglas Arai
Wintergreen Information Services ..... Joun Wheeler
Xerox Corporation..... Dwight McBain

Technical Committee T10 on Lower Level Interfaces, which developed and reviewed this standard, had the following members:

John B. Lohmeyer, Chair
Lawrence J. Lamers, Vice-chair
Ralph O. Weber, Secretary

I. D. Allan
P. D. Aloisi
G. Barton
R. Bellino
C. Brill
J. Chen
R. Cummings
Z. Daggett
J. Dambach
J. V. Dedek
E. Fong
E. A. Gardner
L. Grantham
D. Guss
K. J. Hallam
N. Harris
E. Haske
S. F. Heil
S. Holmstead
P. Johansson
G. Johnsen
S. Jones
T. J. Kulesza
E. Lappin
R. Liu
B. McFerrin
J. McGrath
P. McLean
P. Mercer
G. Milligan
C. Monia
D. Moore
I. Morrell
J. Moy
S. Nadershahi
E. Oetting
D. Pak
G. Penokie
A. E. Pione
D. Piper
R. Reisch
S. D. Schueler
R. N. Snively
G. R. Stephens
C. E. Strang, Jr.
T. Totani
D. Wagner
D. Wallace
J. L. Williams
M. Wingard
M. Yokoyama

Revision history

Revision 1 (March 11, 1997)

First release of working draft.

American National Standard for Information Systems –

SCSI Transport *via* SBP-2

1 Scope and purpose

1.1 Scope

This standard defines methods, data structures and code values that enable Serial Bus Protocol 2 (SBP-2) to be utilized in the implementation of SCSI devices within the domain of IEEE Std 1394-1995, High Performance Serial Bus.

1.2 Purpose

As Serial Bus Protocol 2 (SBP-2) was developed, it evolved to become general-purpose, low-level interface for the transport and management of commands, data and status on IEEE Std 1394-1995, High Performance Serial Bus. SBP, the progenitor of SBP-2, specified the use of Serial Bus facilities for the transport of SCSI commands but SBP-2 has no connection with SCSI nor any other particular command set. This shift in focus for SBP-2 created the necessity for this document: a standard to map SCSI Architecture Model (SAM-2) requirements to their equivalent SBP-2 facilities. This encompasses how SAM-2 functions are expressed within SBP-2 and the format of data structures, such as status and sense data, that require standardization.

2 Normative references

The standards named in this section contain provisions which, through reference in this text, constitute provisions of this American National Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision; parties to agreements based on this American National Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

Copies of the following documents can be obtained from ANSI:

Approved ANSI standards;

Approved and draft regional and international standards (ISO, IEC, CEN/CENELEC and ITUT); and

Approved and draft foreign standards (including BIS, JIS and DIN).

For further information, contact the ANSI Customer Service Department by telephone at (212) 642-4900, by FAX at (212) 302-1286 or *via* the world wide web at http://www.ansi.org.

Additional contact information for document availability is provided below as needed.

2.1 Approved references

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

IEEE Std 1394-1995, Standard for a High Performance Serial Bus

ISO/IEC 13213:1994, Control and Status Register (CSR) Architecture for Microcomputer Buses

2.2 References under development

At the time of publication, the following referenced standards were still under development.

T10 Project 1155D, ANSI NCITS.xxx-199x, Serial Bus Protocol 2 (SBP-2)

T10 Project 1157D, ANSI NCITS.xxx-199x, SCSI Architecture Model 2 (SAM-2)

3 Definitions and notation

3.1 Definitions

3.1.1 Conformance

Several keywords are used to differentiate levels of requirements and optionality, as follows:

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

3.1.1.2 ignored: A keyword that describes bits, bytes, quadlets, octlets or fields whose values are not checked by the recipient.

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

3.1.1.4 reserved: A keyword used to describe objects—bits, bytes, quadlets, octlets and fields—or the code values assigned to these objects in cases where either the object or the code value is set aside for future standardization. Usage and interpretation may be specified by future extensions to this or other standards. A reserved object shall be zeroed or, upon development of a future standard, set to a value specified by such a standard. The recipient of a reserved object shall not check its value. The recipient of a defined object shall check its value and reject reserved code values.

3.1.1.5 shall: A keyword that indicates a mandatory requirement. Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this standard.

3.1.1.6 should: A keyword that denotes flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “is recommended.”

3.1.2 Glossary

The following terms are used in this standard:

3.1.2.1 byte: Eight bits of data.

3.1.2.2 doublet: Two bytes, or 16 bits, of data.

3.1.2.3 kilobyte: A quantity of data equal to 2^{10} bytes.

3.1.2.4 login: The process by which an initiator obtains access to a set of target fetch agents. The target fetch agents and their control and status registers provide a mechanism for an initiator to convey ORB’s to the target.

3.1.2.5 login ID: A value assigned by the target during a login process. For all logins, the login ID establishes a relationship between an initiator and a task set: an initiator ID. The login ID is used to identify subsequent requests from an initiator; in some cases the login ID is not present in the operation request block and its value is implicit.

3.1.2.6 octlet: Eight bytes, or 64 bits, of data.

3.1.2.7 operation request block: A variable length data structure fetched from initiator memory by a target in order to execute the command encapsulated within it.

3.1.2.8 quadlet: Four bytes, or 32 bits, of data.

3.1.2.9 status block: A fixed length data structure written to initiator memory by a target when an operation request block has been completed.

3.1.2.10 task: A task is an organizing concept that represents the work to be done by a target to carry out a command. In order to perform a task, a target maintains context information for the task, which includes (but is not limited to) the command, parameters such as data transfer addresses and lengths, completion status and ordering relationships to other tasks. A task has a lifetime, which commences when the task is signaled to the target, proceeds through a period of execution by the target and finishes when completion status is signaled to the initiator. While a task is active, it makes use of both target resources and initiator resources.

3.1.2.11 task set: A group of tasks available for execution by a logical unit of a target. There may be dependencies between individual tasks within the task set specified by this standard.

3.1.2.12 terabyte: A quantity of data equal to 2^{40} bytes.

3.1.2.13 unit: A component of a Serial Bus node that provides processing, memory, I/O or some other functionality. A SCSI peripheral is an example of a unit. Once the node is initialized, the unit provides a CSR interface that is typically accessed by device driver software at an initiator. A node may have multiple units, which normally operate independently of each other.

3.1.2.14 unit architecture: The specification of the interface to and the behaviors of a unit implemented within a Serial Bus node. This standard is a unit architecture for SCSI targets that utilize SBP-2.

3.1.2.15 unit attention: A state that a logical unit maintains while it has unsolicited status information to report to one or more logged-in initiators. A unit attention condition shall be created as described elsewhere in this standard or in the applicable command set- and device-dependent documents. A unit attention condition shall persist for a logged-in initiator until a) unsolicited status that reports the unit attention condition is successfully stored at the initiator or b) the initiator's login becomes invalid or is released. Logical units may queue unit attention conditions. After the first unit attention condition is cleared, another unit attention condition may exist.

3.1.3 Abbreviations

The following are abbreviations that are used in this standard:

CDB	Command descriptor block
CSR	Control and status register
CRC	Cyclical redundancy checksum
EUI-64	Extended Unique Identifier, 64-bits
LUN	Logical unit number
ORB	Operation request block
SAM-2	SCSI Architecture Model 2
SBP-2	Serial Bus Protocol 2

3.2 Notation

The following conventions should be understood by the reader in order to comprehend this standard.

3.2.1 Numeric values

Decimal, hexadecimal and, occasionally, binary numbers are used within this standard. By editorial convention, decimal numbers are most frequently used to represent quantities or counts. Addresses are uniformly represented by hexadecimal numbers. Hexadecimal numbers are also used when the value represented has an underlying structure that is more apparent in a hexadecimal format than in a decimal format. Binary numbers are used infrequently and generally limited to the representation of bit patterns within a field.

Decimal numbers are represented by Arabic numerals without subscripts or by their English names. Hexadecimal numbers are represented by digits from the character set 0 – 9 and A – F followed by the subscript 16. Binary numbers are represented by digits from the character set 0 and 1 followed by the subscript 2. For the sake of legibility, binary and hexadecimal numbers are separated into groups of four digits separated by spaces.

As an example, 42, $2A_{16}$ and $0010\ 1010_2$ all represent the same numeric value.

3.2.2 Bit, byte and quadlet ordering

SBP-2 is defined to use the facilities of Serial Bus, IEEE Std 1394-1995, and therefore uses the ordering conventions of Serial Bus in the representation of data structures. This standard, which builds upon SBP-2, follows suite. In order to promote interoperability with memory buses that may have different ordering conventions, this standard defines the order and significance of bits within bytes, bytes within quadlets and quadlets within octlets in terms of their relative position and not their physically addressed position.

Within a byte, the most significant bit, *msb*, is that which is transmitted first and the least significant bit, *lsb*, is that which is transmitted last on Serial Bus, as illustrated below. The significance of the interior bits uniformly decreases in progression from *msb* to *lsb*.

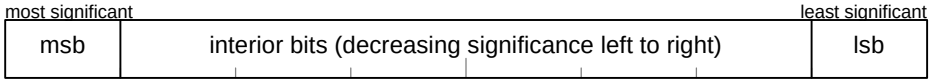

Figure 1 Bit ordering within a byte

Within a quadlet, the most significant byte is that which is transmitted first and the least significant byte is that which is transmitted last on Serial Bus, as shown below.

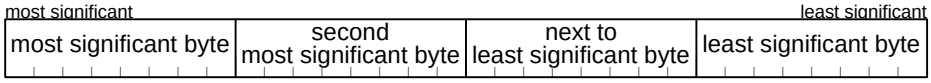

Figure 2 Byte ordering within a quadlet

Within an octlet, which is frequently used to contain 64-bit Serial Bus addresses, the most significant quadlet is that which is transmitted first and the least significant quadlet is that which is transmitted last on Serial Bus, as the figure below indicates.

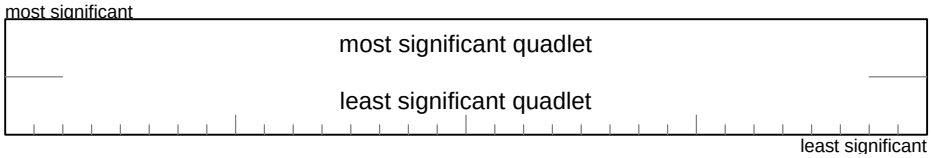

Figure 3 Quadlet ordering within an octlet

Increasing Serial Bus addresses for quadlets correspond to increasing addresses on other buses bridged to Serial Bus, but the correlation of addresses is problematical when block transfers take place that are not quadlet aligned or not an integral number of quadlets. In such cases, no assumptions can be made about the ordering (significance within a quadlet) of bytes at the unaligned beginning or fractional quadlet end of such a block transfer, unless an application has knowledge (outside of the scope of this standard) of the ordering conventions of the other bus.

4 Model (informative)

To be determined…

5 SCSI Architecture Model compliance

This section specifies how to make use of features of SBP-2 to implement SCSI devices that are compliant with the SCSI Architecture Model 2 (SAM-2). The features and the ways in which they shall be used to achieve compliance are set forth in the clauses that follow.

5.1 Object definitions

The SCSI Architecture Model defines objects within the SCSI domain. The equivalency of SBP-2 objects is enumerated below in those cases where the correlation may not be clear and in those cases where SBP-2 restricts the scope of an object

Initiator identifier: The *login_ID* returned by a target in response to a successful login is the initiator identifier for normal command block requests.

Logical unit number: SBP-2 restricts the scope of the logical unit number to 2^{16}. The *lun* field in management ORB's is one doublet.

NOTE – Neither the *login_ID* nor *lun* fields are present in normal command block ORB's, since the value of both is implicit in the CSR addresses of the target fetch agent to which the ORB's are signaled.

Tag: The Serial Bus address of an ORB is the tag by which the task is identified. This mandates that initiator memory allocated to a request shall not be released or reused while the task is active within a task set. The scope of an SBP-2 tag is that of a Serial Bus address, 2^{64}, and equal to the scope defined by SAM-2.

Target identifier: Targets are identified by means of a unit unique ID, or EUI-64, found in the target's configuration ROM. When a unit unique ID leaf is not present in configuration ROM, the value of the unit unique ID shall be construed to be equal to the node unique ID. The scope of a target identifier, 2^{64}, is identical to the scope of a target identifier specified by SAM-2.

Task set: An SBP-2 task set consists of the linked list of normal command block ORB's that are managed by a single target fetch agent. There is no provision in SBP-2 for untagged tasks; an SBP-2 task set always consists of zero or more tagged tasks.

SBP-2 delimits the extent of a task set in a way that is compatible with SAM-2 but that differs from the definition given in SAM-2 of "…a group of tasks *within* a target…" (emphasis added). By way of contrast with SAM-2, an SBP-2 task enters the task set when it is linked into an active request list. The extent of an SBP-2 task set includes all the uncompleted ORB's linked into a request list in initiator memory, not solely the requests already fetched by the target.

Untagged task: SBP-2 does not define untagged tasks.

5.2 Command descriptor block

SBP-2 provides for the transport of 6-, 10- and 12-byte SCSI CDB's but constrains the control byte (the last byte of a SCSI command descriptor block) to values illustrated below.

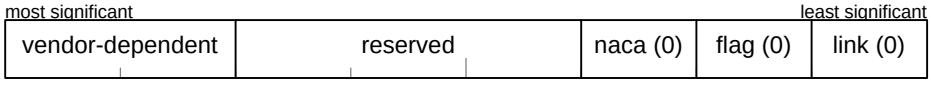

Figure 4 Control byte

The *naca*, or normal ACA, bit shall be zero; SBP-2 targets implement SCSI-2 contingent allegiance.

The *flag* and *link* bits shall be zero; SBP-2 targets do not support linked commands.

5.3 Status

SCSI status is reported in the *status* field, which is part of the status block defined in 5.6.1.

5.4 Command delivery services

SAM-2 requires that four protocol services be defined to support the Execute Command remote procedure call. The SBP-2 facilities used to provide these services are specified below.

5.4.1 Send SCSI Command

The formal arguments of the Send SCSI Command service are:

Task address,
CDB,
[Task Attribute],
[Data-out buffer],
[Command byte count],
[Autosense request]

The task address argument is composed of the address of the ORB and the logical unit for which it is intended. The logical unit is implicit in the target fetch agent to which the ORB is signaled. The request is signaled to a target fetch agent by the methods specified by SBP-2.

The CDB argument is encapsulated within the ORB and fetched by the target from initiator memory.

The task attribute argument, either SIMPLE or ORDERED, is implicit in the target implementation. The Logical_Unit_Characteristics entry in the unit directory indicates which task attribute is implemented. When the *ordered* bit in this entry is zero, the all tasks have an implicit attribute of SIMPLE. Otherwise, if *ordered* is set to one, the task attribute is ORDERED for all tasks.

The data-out buffer argument, if present, is specified by the *data_descriptor* and *data_size* fields in the ORB.

SBP-2 provides no means by which the command byte count argument may be communicated to the target. The target may determine the length of the command by an examination of the operation code in the CDB.

The SBP-2 status block provides for the return of autosense data, although the initiator is expected to reformat the information as SCSI sense data before it is presented to the application client.

5.4.2 SCSI Command Received

The formal arguments of the SCSI Command Received service are:

Task address,
[Task Attribute],
CDB,
[Autosense request]

The task address argument is composed of the address of the ORB and the logical unit for which it is intended. The logical unit is implicit in the target fetch agent to which the ORB is signaled. A Serial Bus write indication for the AGENT_RESET register signals the target fetch agent that there may be a new SCSI command. The details of this indication are specified in SBP-2.

The task attribute argument is implicit in the target implementation and may be determined by an examination of the *ordered* bit in the Logical_Unit_Characteristics entry in configuration ROM.

The CDB argument is encapsulated within the ORB and fetched by the target from initiator memory.

The *status_FIFO* address, provided by the initiator as part of the login procedure, is the address for the return of autosense data.

5.4.3 Send Command Complete

The formal arguments of the Send Command Complete service are:

Task identifier,
[Sense data],
Status,
Service response

The task identifier argument is derived from the address to which the status information is stored. The ORB specified the address for the status block, either implicitly by means of a fixed offset from the address of the ORB or explicitly by means of the *status_FIFO* field. In either case, the initiator shall ensure that the address at which status is stored is sufficient to uniquely correlate the status with the task identifier.

SCSI sense data may be returned in the status block upon completion of a SCSI command.

The service response argument is encoded within the status block by *resp* and *sbp_status* as summarized below.

Table 1 SAM-2 Service responses

Service response	*resp*	*sbp_status*	Description
Task complete	0	0	The task has ended with a completion status indicated by *status*
Linked command complete	—	—	Not supported by SBP-2
Linked command complete (with flag)	—	—	Not supported by SBP-2
Function complete	0	0	Used by task management functions
Service delivery or target failure	1	Various (as defined by SBP-2)	The command has completed because of a Serial Bus service failure or a target malfunction
Function rejected	0	9	Used by task management functions

5.4.4 Command Complete Received

The formal arguments of the Command Complete Received service are:

Task address,
[Data-in buffer],
[Sense data],
Status,
Service response

The task address argument is derived from the address to which the status information was stored. The ORB specified the address for the status block, either implicitly by means of a fixed offset from the address of the ORB or explicitly by means of the *status_FIFO* field. In either case, the initiator shall ensure that the address at which status is stored is sufficient to uniquely correlate the status with the task identifier.

SCSI sense data may be returned in the status block upon completion of a SCSI command.

The service response argument is encoded within the status block by *resp*, as specified by Table 1 – SAM-2 Service responses

.

5.5 Data transfer services

SAM-2 requires that four protocol services be defined to support data transfer necessary for the Execute Command remote procedure call. The SBP-2 facilities used to provide these services are specified below.

5.5.1 Send Data-in

The formal arguments of the Send Data-in service are:

Task identifier,
Device server buffer,
Application client buffer offset,
Request byte count

The task identifier argument is the Serial Bus address of the ORB for the active task. It is expected that target implementations reference a copy of the ORB maintained in the device's local memory, although nothing precludes a fetch of the information from the initiator memory occupied by the ORB.

The device service buffer argument is vendor-dependent. The data available in the device server buffer shall be formed into Serial Bus write transactions, as described below.

The application client buffer offset is a value maintained by the device server to correlate medium locations with locations in the application client buffer. The base of the application client buffer is specified by the *data_descriptor* field supplied by the initiator.

The request byte count is determined by the device server.

The target shall use one or more Serial Bus quadlet or block write requests to store the requested data into the application client buffer. For the sake of efficiency, it is expected that the target uses the largest block write requests permitted by the *max_payload* field in the ORB and transmit these requests at the speed mandated by the *spd* field in the ORB.

5.5.2 Data-in Delivered

The formal arguments of the Data-in Delivered service are:

Task identifier

Upon completion of each of the quadlet or block write requests initiated as a result of Send Data-in, the target shall receive Serial Bus write response confirmations. It is the target's responsibility to correlate the Serial Bus addresses (for which write responses are received) with the task identifier in order to provide the Data-in Delivered confirmation.

5.5.3 Receive Data-out

The formal arguments of the Receive Data-out service are:

Task identifier,
Application client buffer offset,
Request byte count,
Device server buffer

The task identifier argument is the Serial Bus address of the ORB for the active task. It is expected that target implementations reference a copy of the ORB maintained in the device's local memory, although nothing precludes a fetch of the information from the initiator memory occupied by the ORB.

The application client buffer offset is a value maintained by the device server to correlate medium locations with locations in the application client buffer. The base of the application client buffer is specified by the *data_descriptor* field supplied by the initiator.

The request byte count is determined by the device server.

The device service buffer argument is vendor-dependent. The data obtained from Serial Bus read responses shall be moved to the device server buffer, as described below.

The target shall use one or more Serial Bus quadlet or block read requests to fetch the requested data from the application client buffer. For the sake of efficiency, it is expected that the target uses the largest block read requests permitted by the *max_payload* field in the ORB and transmit these requests at the speed mandated by the *spd* field in the ORB.

5.5.4 Data-out received

The formal arguments of the Data-out received service are:

Task identifier

Upon completion of each of the quadlet or block read requests initiated as a result of Receive Data-out, the target shall receive Serial Bus read response confirmations and their accompanying data. It is the target's responsibility transfer the data to the device server buffer and to correlate the Serial Bus addresses (for which read responses are received) with the task identifier in order to provide the Data-out Received confirmation.

5.6 Contingent allegiance

SBP-2 targets implement SCSI-2 contingent allegiance and do not support CDB's whose *naca* bit in the control byte is one. The contingent allegiance condition shall exist within a task set when a logical unit stores a status block for a command where *status* is set to CHECK CONDITION or COMMAND TERMINATED. Since SBP-2 targets implement autosense *via* the return of the status block, the contingent allegiance condition is automatically cleared.

At the time a contingent allegiance condition is created, the logical unit shall:

a) immediately halt the operations of the fetch agent for the faulted initiator;

b) abort the task set in the same fashion as if an ABORT TASK SET task management function had been signaled to the target;

c) clear the contingent allegiance condition.

Because the faulted initiator's fetch agent has been halted, it is necessary for the initiator to reset and reinitialize the fetch agent before any commands may be signaled to the target.

5.6.1 Sense data

Upon completion of a command, if the *notify* bit in the ORB is one or if there is exception status to report, the target shall signal the initiator by storing all or part of the status block shown below at the *status_FIFO* address provided by the initiator as part of the login request.

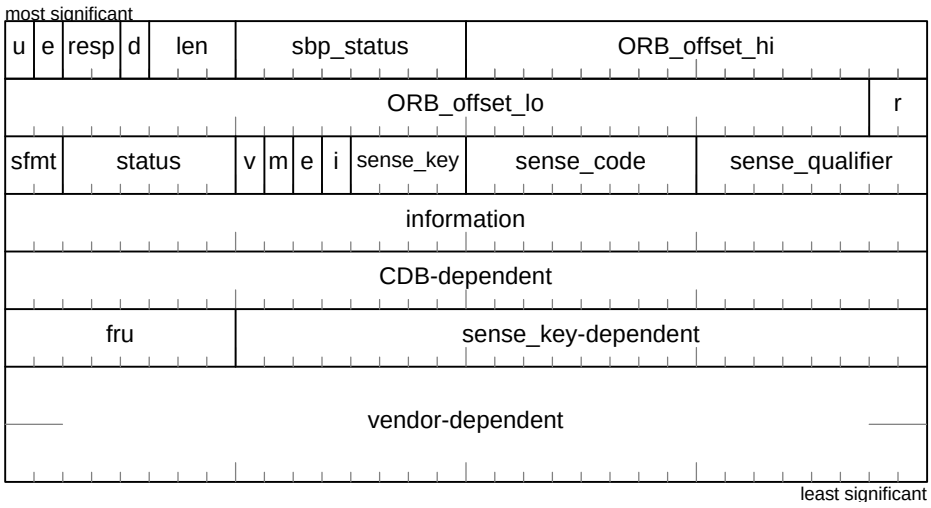

Figure 5 Status block format for SCSI sense data

When a command completes with GOOD status, only the first two quadlets of the status block shall be stored at the *status_FIFO* address; the *len* field shall be one. Otherwise, both SCSI status and sense data shall be stored in a status block that conforms to the format illustrated above.

NOTE – SBP-2 permits the return of a status block between two and eight quadlets in length. When a truncated status block is stored, the omitted quadlets shall be interpreted as if zero values were stored.

The *unsolicited*, *end_of_list* and *dead* bits (abbreviated as *u*, *e* and *d*, respectively, in the figure above), as well as the *resp*, *len*, *sbp_status*, *ORB_offset_hi* and *ORB_offset_lo* fields, are as previously described in SBP-2.

The *sfmt* field shall specify the format of the status block and shall additionally indicate whether the error condition associated with *sense_key* is current or deferred. The table below defines permissible values for *sfmt*.

Value	Description
0	Current error; status block format defined by this standard
1	Deferred error; status block format defined by this standard
2	Reserved for future standardization
3	Status block format vendor-dependent

The *status* field shall specify SCSI status information as defined by SAM-2, with the exceptions noted in the table below.

Value	Description
0	GOOD
2	CHECK CONDITION
4	CONDITION MET
8	BUSY
10_{16}	Not supported by SBP-2 devices
14_{16}	Not supported by SBP-2 devices
18_{16}	RESERVATION CONFLICT
22_{16}	COMMAND TERMINATED
28_{16}	Not supported by SBP-2 devices
30_{16}	Not supported by SBP-2 devices
All other values	Reserved for future standardization

The *valid* bit (abbreviated as *v* in the figure above) shall specify the content of the *information* field. When the *valid* bit is zero, the contents of the information field are not specified. When the *sfmt* field has a value of zero or one and the *valid* bit is one, the contents of the information field shall be as defined by SPC or the relevant command set standard.

The meanings of the *mark*, *eom* and *illegal_length_indicator* bits (abbreviated as *m*, *e* and *i*, respectively, in the figure above) are device-type dependent within the command set standard(s).

The *sense_key*, *sense_code* and *sense_qualifier* fields shall specify command completion information defined by SPC or the relevant the command set standard. These fields correspond to the sense key, additional sense code and additional sense code qualifier fields defined by SPC for sense data.

The contents of the *information* field are unspecified if either the *valid* bit is zero or the *sfmt* field has a value of three. For *sfmt* values of one or two, the contents of the *information* field are device-type or command dependent and, if the *valid* bit is one, are defined within SPC or the appropriate standard for the command. Characteristic uses of the *information* field are for:

- the unsigned logical block address associated with *sense_key* and the command; or
- the least significant 32-bits of the unsigned logical block address associated with *sense_key* and the command; or
- the residue of the requested data transfer length minus the actual data transfer length, in either bytes or blocks as determined by the command. Negative values are indicated in two’s complement notation.

The contents of the *CDB*-dependent field are device-type or command dependent and are defined within the appropriate standard for the command.

Nonzero values in the *fru* field may be used to identify a device-dependent, field replaceable mechanism or unit that has failed. A value of zero in this field shall indicate that no specific mechanism or unit has been identified to have failed or that the data is unavailable. When *fru* is nonzero, the format of the information is not specified by this standard.

When *sfmt* has a value of zero or one, the contents of the *sense_key*-dependent field are defined by SPC or the relevant the command set standard. In this case the most significant bit of the *sense_key*-dependent field is the *sksv* bit defined by SPC. When *sfmt* is equal to three, the contents of *sense_key*-dependent are unspecified.

5.7 Asynchronous event reporting

SBP-2 does not support asynchronous event reporting as defined by SAM-2.

5.8 Autosense

SBP-2 supports autosense through the return of a status block to the address specified by the login parameter *status_FIFO*. The status block is always stored in the event of an exception condition, *e.g.*, CHECK CONDITION or TERMINATE TASK.

5.9 Hard reset

A Serial Bus reset shall cause a target to execute a hard reset, as defined by SAM-2.

5.10 Task management functions

SBP-2 targets implement different levels of task management functions, according to the queuing model supported. A target may support a basic or full task set model. The basic and full task management models are as specified by SAM-2.

The relationship between the task management model and the task management functions supported is given by the table below.

Function	Basic	Full	Comments
ABORT TASK	Required	Required	
ABORT TASK SET	Required	Required	May also be performed directly through the AGENT_RESET register
CLEAR ACA	—	—	SBP-2 devices implement SCSI-2 contingent allegiance
CLEAR TASK SETS	Optional	Required	
TARGET RESET	Required	Required	May also be performed directly through the RESET_START register
LOGICAL UNIT RESET	Optional	Optional	Functions as TARGET RESET but scope is limited to a single logical unit
TERMINATE TASK	Not supported	Optional	The basic model provides no control over individual tasks

An initiator may consult the Logical_Unit_Characteristics entry in the unit directory in configuration ROM to determine which task management model is implemented.

SAM-2 additionally requires protocol services to support the task management functions, enumerated below. In all of the definitions for the task management function protocol services, the following apply:

- The object address is as specified by SAM-2 and modified by the SBP-2 object definitions;
- The function identifier is one of the six task management functions defined by SAM-2; and
- The service response is one of Function complete, Function rejected or Service delivery or target failure. These service responses are encoded by the *resp* and *sbp_status* fields in the status block stored by the target upon completion of a request. See Table 1 for the numeric values that encode the service responses.

5.10.1 Send task management request

The formal arguments of the Send task management request service are:

Object address,
Function identifier

Subsequent to the creation of a task management ORB in initiator memory, the initiator signals the request to the target management agent by the methods described in SBP-2.

5.10.2 Task management request received

The formal arguments of the Task management request received service are:

Object identifier,
Function identifier

When a Serial Bus write indication is received for the target's DOORBELL register, the target may then fetch the request from initiator memory.

5.10.3 Task management function executed

The formal arguments of the Task management function executed service are:

Object identifier,
Service response

The target signals the completion of the task management function by storing an 8-byte status block at the address specified by *status_FIFO* in the ORB.

5.10.4 Received Function-executed

The formal arguments of the Received Function-executed service are:

Object address,
Service response

When the initiator receives a Serial Bus write indication for data addressed to the *status_FIFO* address, it may examine the status block to determine the service response from *resp*.

6 Configuration ROM

All nodes that implement SCSI target(s) as an SBP-2 unit architecture shall implement general format configuration ROM in accordance with ISO/IEC 13213:1994, IEEE Std 1394-1995, ANSI NCITS.xxx-199x Serial Bus Protocol 2 and this standard. General format configuration ROM is a self-descriptive structure as illustrated below. The bus information block and root directory are at fixed locations; all other directories and leaves are addressed by entries in their parent directory.

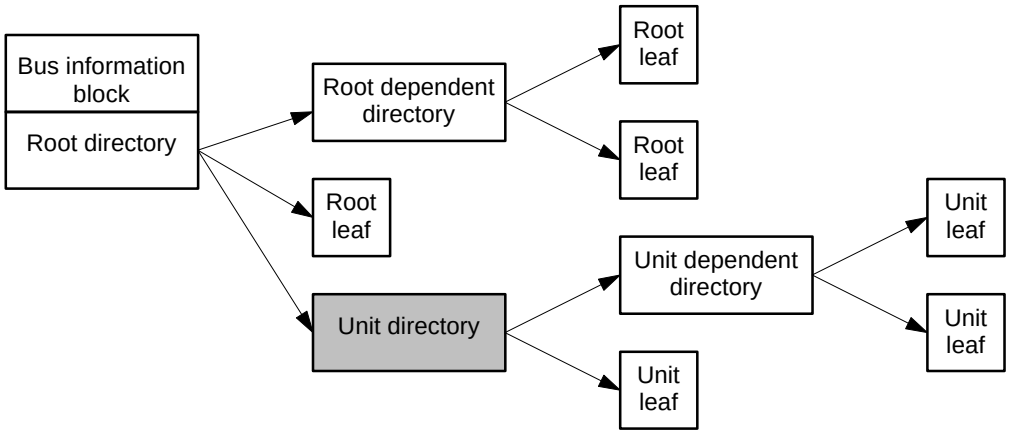

Figure 6 Configuration ROM hierarchy

A SCSI target is described by its unit directory (shown highlighted in the figure above). In addition to the configuration ROM entries required by SBP-2, the target shall implement the unit directory entries described in the clauses that follow.

Targets shall implement at least one logical unit, logical unit zero. Additional logical units may be implemented. A logical unit is described by entries in the unit directory or by entries in a logical unit directory dependent upon the unit directory or by entries taken in combination from both places.

6.1 Command_Set_Spec_ID entry

The Command_Set_Spec_ID entry is an immediate entry in the unit directory that specifies the organization responsible for the command set definition for the target. Figure 7 shows the format of this entry.

Figure 7 Command_Set_Spec_ID entry format

38_{16} is the concatenation of *key_type* and *key_value* for the Command_Set_Spec_ID entry.

00 $609E_{16}$ is the *unit_spec_ID* obtained by NCITS from the IEEE/RAC. The value indicates that the NCITS Secretariat is responsible for the command set definition.

6.2 Command_Set entry

The Command_Set entry is an immediate entry in the unit directory that in combination with the *command_set_spec_ID* specifies the command set implemented by the target. Figure 8 shows the format of this entry.

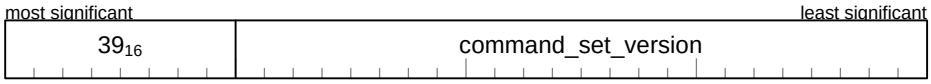

Figure 8 Command_Set entry format

39_{16} is the concatenation of *key_type* and *key_value* for the Command_Set entry.

The meaning of *command_set_version* shall be specified by the owner of *command_set_spec_ID*.

6.3 Logical_Unit_Characteristics entry

The Logical_Unit_Characteristics entry is an immediate entry that, when present in either the unit directory or a logical unit directory, specifies characteristics of the target implementation. Figure 9 shows the format of this entry.

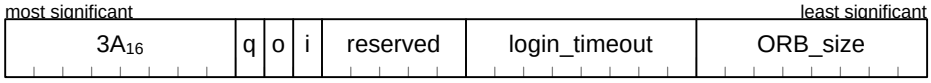

Figure 9 Logical_Unit_Characteristics entry format

$3A_{16}$ is the concatenation of *key_type* and *key_value* for the Logical_Unit_Characteristics entry.

The *q* bit shall specify the task management (queuing) model implemented by the target. If *q* is zero, the target implements the basic task management model defined by SBP-2. When *q* is one, the task management model is dependent upon the command set specified by the Command_Set_Spec_ID and Command_Set_Version entries.

The *ordered* bit (abbreviated as *o* in the figure above) specifies the manner in which the target executes tasks signaled to the normal command block agent. If the target executes and reports completion status without any ordering constraints, the *ordered* bit shall be zero. Otherwise, if the target both executes all tasks in order and reports their completion status in the same order, the *ordered* bit shall be one.

The *isochronous* bit (abbreviated as *i* in the figure above) specifies whether or not the target supports isochronous operations. When *isochronous* is one, create stream requests, stream command block requests and stream control requests shall all be supported. If the *isochronous* bit is one, the *irmc*, *cmc* and *isc* bits in the bus information block shall also be one, as described in IEEE Std 1394-1995.

The *login_timeout* field shall specify, in units of 500 milliseconds, the maximum time an initiator allows for a target to store a status block in response to the initiator's login request.

The *ORB_size* field shall specify, in quadlets, the fetch size used by the target to obtain ORB's from initiator memory. The initiator shall allocate, on a quadlet aligned boundary, at least this much memory for each ORB signaled to the target.

6.4 Logical_Unit_Number entry

The Logical_Unit_Number entry is an immediate entry that, when present in either the unit directory or a logical unit directory, specifies the peripheral device type and logical unit number of a logical unit implemented by the target. Figure 10 shows the format of this entry.

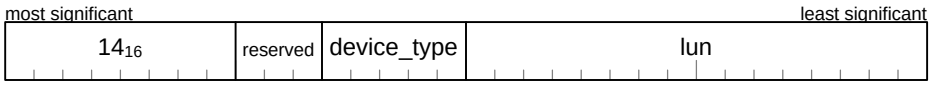

Figure 10 Logical_Unit_Number entry format

14_{16} is the concatenation of *key_type* and *key_value* for the Logical_Unit_Number entry.

The *device_type* field indicates the peripheral device type implemented by the logical unit. This field shall contain a value specified by the table below.

Value	Peripheral device type
$0 – 1E_{16}$	The value of *device_type* shall have the same meaning as the peripheral device type field returned in INQUIRY data as specified by SPC
$1F_{16}$	Unknown device type

The *lun* field shall identify the logical unit to which the information in the Logical_Unit_Number entry applies.

Annex A
(informative)

Sample configuration ROM

Configuration ROM is located at a base address of FFFF F000 0400_{16} within a node's initial memory space. The requirements for general format configuration ROM for targets are specified in section 6. This annex contains an illustration of a typical configuration ROM for a simple target.

most significant

4	0014_{16}	ROM CRC (calculated)
3133 3934_{16} (ASCII "1394")		
node_options (00FF 2000_{16})		
node_vendor_ID		chip_ID_hi
chip_ID_lo		
4		Root directory CRC (calculated)
03_{16}	module_vendor_ID	
$0C_{16}$	node_capabilities (00 $83C0_{16}$)	
$8D_{16}$	2	
$D1_{16}$	4	
2		Leaf CRC (calculated)
node_vendor_ID		chip_ID_hi
chip_ID_lo		
7		Unit directory CRC (calculated)
12_{16}	unit_spec_ID (00 $609E_{16}$)	
13_{16}	unit_sw_version (01 0483_{16})	
38_{16}	command_set_spec_ID (00 $609E_{16}$)	
39_{16}	command_set_version	
54_{16}	csr_offset (00 4000_{16})	
$3A_{16}$	01 $0A08_{16}$	
14_{16}	00 0000_{16}	

least significant

Figure A.1 Sample configuration ROM

The ROM CRC in the first quadlet is calculated on the twenty quadlets of ROM information that follow.

A.1 Root directory

The *node_options* field represents a collection of bits and fields specified in IEEE Std 1394-1995. The value shown, 00FF 2000_{16}, represents basic characteristics of a device that is not isochronous capable. This value is composed of a *cyc_clk_acc* field with a value of FF_{16} and a *max_rec* value of two. The *max_rec* field encodes a maximum payload of eight bytes in block write requests addressed to the target.

The Node_Capabilities entry in the root directory, with *key_type* and *key_value* fields of $0C_{16}$, has a value where the *spt*, *64*, *fix*, *lst* and *drq* bits are all one. This is a minimum requirement for targets.

The Node_Unique_ID entry in the root directory, with *key_type* and *key_value* fields of $8D_{16}$, has an *indirect_offset* value of two that points to the node unique ID leaf.

The Unit_Directory entry in the root directory, with *key_type* and *key_value* fields of $D1_{16}$, has an *indirect_offset* value of four that points to the unit directory.

A.2 Unit directory

The Command_Set_Spec_ID and Command_Set_Version entries, with *key_type* and *key_value* fields of 38_{16} and 39_{16}, respectively, define the command set used by the target.

The Management_Agent entry in the unit directory, with *key_type* and *key_value* fields of 54_{16}, has a *csr_offset* value of $00\ 4000_{16}$ that indicates that the management agent CSR has a base address of FFFF F001 0000_{16} within the node's initial memory space.

The Logical_Unit_Characteristics entry in the unit directory, with *key_type* and *key_value* fields of $3A_{16}$, has an immediate value of $01\ 0A08_{16}$. This indicates a target that implements the basic task management model, may reorder tasks without restriction and does not support isochronous operations. In addition, the target is expected to complete a login within five seconds and fetches 32-byte ORB's.

The Logical_Unit_Number entry in the unit directory, with *key_type* and *key_value* fields of 14_{16}, has an immediate value of zero that indicates a direct-access device with a logical unit number of zero.