CONGRUENT SOFTWARE, INC.
3998 Whittle Avenue
Oakland, CA 94602
(510) 531-5472
(510) 531-2942 FAX

FROM: Peter Johansson

TO: T10 SBP-2 *ad hoc* Working Group

DATE: August 19, 1997

RE: Proposed SBP-2 Annex

Annex Q
(normative)

SCSI Transport *via* SBP-2

SBP-2 defines a protocol that permits initiator(s) to control the operation of devices (disks, tapes, printers, *etc.*), but it does not specify the command sets used by the devices—only the mechanisms by which commands, data and status are transported. This annex specifies how SBP-2 may be used for SCSI devices. This encompasses how SCSI command descriptor blocks (CDB's) are encapsulated, a standard format for SCSI status and sense data, the necessary configuration ROM entries and a mapping of SCSI Architecture Terminology to SBP-2.

Q.1 SCSI command encapsulation

SBP-2 provides for the transport of 6-, 10- and 12-byte SCSI CDB's within a normal command block ORB, as illustrated by Figure Q.1.

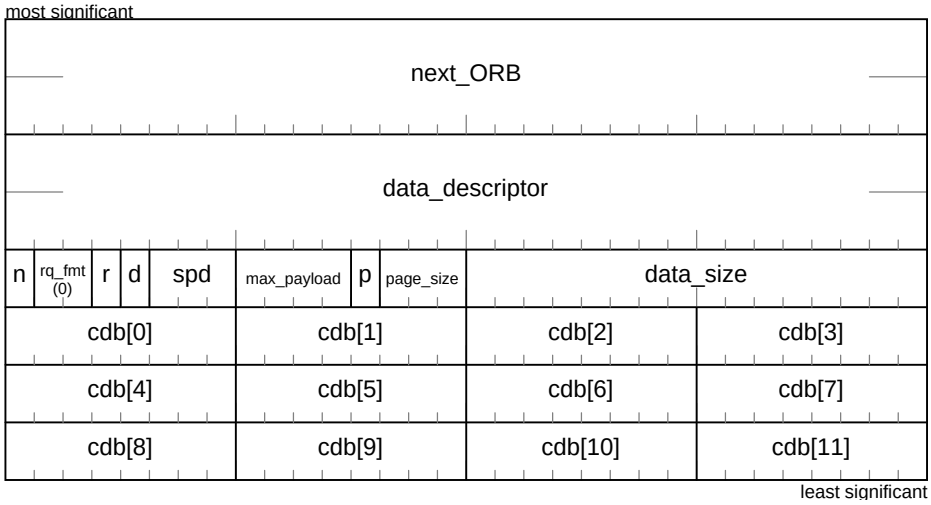

Figure Q.1 – SCSI command block ORB

A SCSI CDB may also be transported within a stream command block ORB (see 5.1.2.2).

The control byte (the last byte of a SCSI command descriptor block) is constrained to values illustrated below.

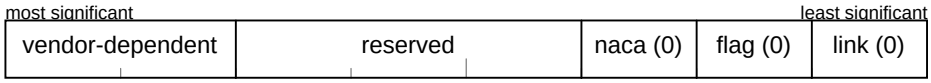

Figure Q.2 – Control byte

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

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

Q.2 SCSI status and 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.

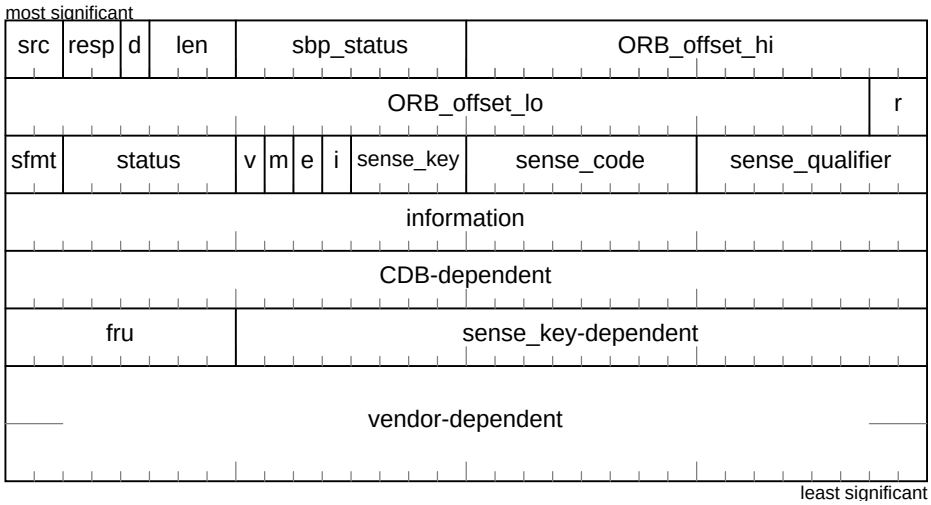

Figure Q.3 – Status block format for SCSI sense data

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 *src*, *resp*, *len*, *sbp_status*, *ORB_offset_hi* and *ORB_offset_lo* fields, as well as the *dead* bit (abbreviated as *d* in the figure above), are as previously described in 5.3.

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.

Q.3 Access control

TO BE DETERMINED – Move A.4 and the definition of the SET PASSWORD management ORB here from the mass storage profile?

Q.4 Configuration ROM

SCSI targets shall implement configuration ROM in accordance with section 7 and this annex. At least one logical unit, logical unit zero, shall be implemented; additional logical units may be implemented. A logical unit is described by entries in a unit directory or by entries in a logical unit directory dependent upon the unit directory or by entries taken in combination from both places.

Mandatory and optional configuration ROM entries for SCSI targets, and the directories in which they may occur, are summarized by the table below.

Entry	Unit directory	Logical unit directory	Comments
Unit_Spec_ID	Mandatory	—	
Unit_SW_Version	Mandatory	—	
Command_Set_Spec_ID	Mandatory (in at least one directory)		See precedence rules below.
Command_Set	Mandatory (in at least one directory)		See precedence rules below.
Command_Set_Revision	Optional	Optional	See precedence rules below.
Firmware_Revision	Optional	—	
Management_Agent	Mandatory	—	
Logical_Unit_Characteristics	Mandatory (in at least one directory)		See precedence rules below
Logical_Unit_Directory	Optional	—	
Logical_Unit_Number	Mandatory (in at least one directory)		See precedence rules below.
Unit_Unique_ID	Optional	—	

For the entries which may be present in either a unit directory or a logical unit directory, the entry in the logical unit directory takes precedence over that in the parent unit directory (if any). If there is no entry in the logical unit directory, the value is inherited from the corresponding entry in the parent unit directory.

The presence of one or more Logical_Unit_Number entries in a unit directory makes the Command_Set_Spec_ID, Command_Set and Logical_Unit_Characteristics entries mandatory in the unit directory.

Within a logical unit directory, at least one Logical_Unit_Number entry is required and the Command_Set_Spec_ID, Command_Set and Logical_Unit_Characteristics entries are all mandatory unless the corresponding entry is present in the parent unit directory.

Q.4.1 Command_Set_Spec_ID entry

The Command_Set_Spec_ID entry is an immediate entry in either a unit or logical unit directory that specifies the organization responsible for the command set definition for the target. The format of this entry is specified by 7.5.3.

SCSI targets shall have a *command_set_spec_ID* value of $00\,609E_{16}$, which indicates that NCITS is responsible for the command set definition.

Q.4.2 Command_Set entry

The Command_Set entry is an immediate entry in either a unit or logical unit directory that, in combination with the *command_set_spec_ID*, specifies the command set implemented by the target. The format of this entry is specified by 7.5.4.

SCSI targets shall have a *command_set* value of $01\,04D8_{16}$, which indicates that the target's command set is specified by SCSI Primary Commands (SPC) and related command set standard(s)—as determined by the target's peripheral device type(s). In addition, this *command_set* value specifies that the target conforms to all requirements of this annex.

Q.4.3 Logical_Unit_Number entry

The Logical_Unit_Number entry is an immediate entry in either a unit or logical unit directory that specifies the peripheral device type and logical unit number of a logical unit implemented by the SCSI target. The format of this entry is specified by 7.5.10.

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

Q.4.4 Sample configuration ROM for a SCSI target

Configuration ROM is located at a base address of $FFFF\,F000\,0400_{16}$ within a node's initial memory space. This clause contains an illustration of a typical configuration ROM for a simple SCSI target and includes the optional textual information leaves.

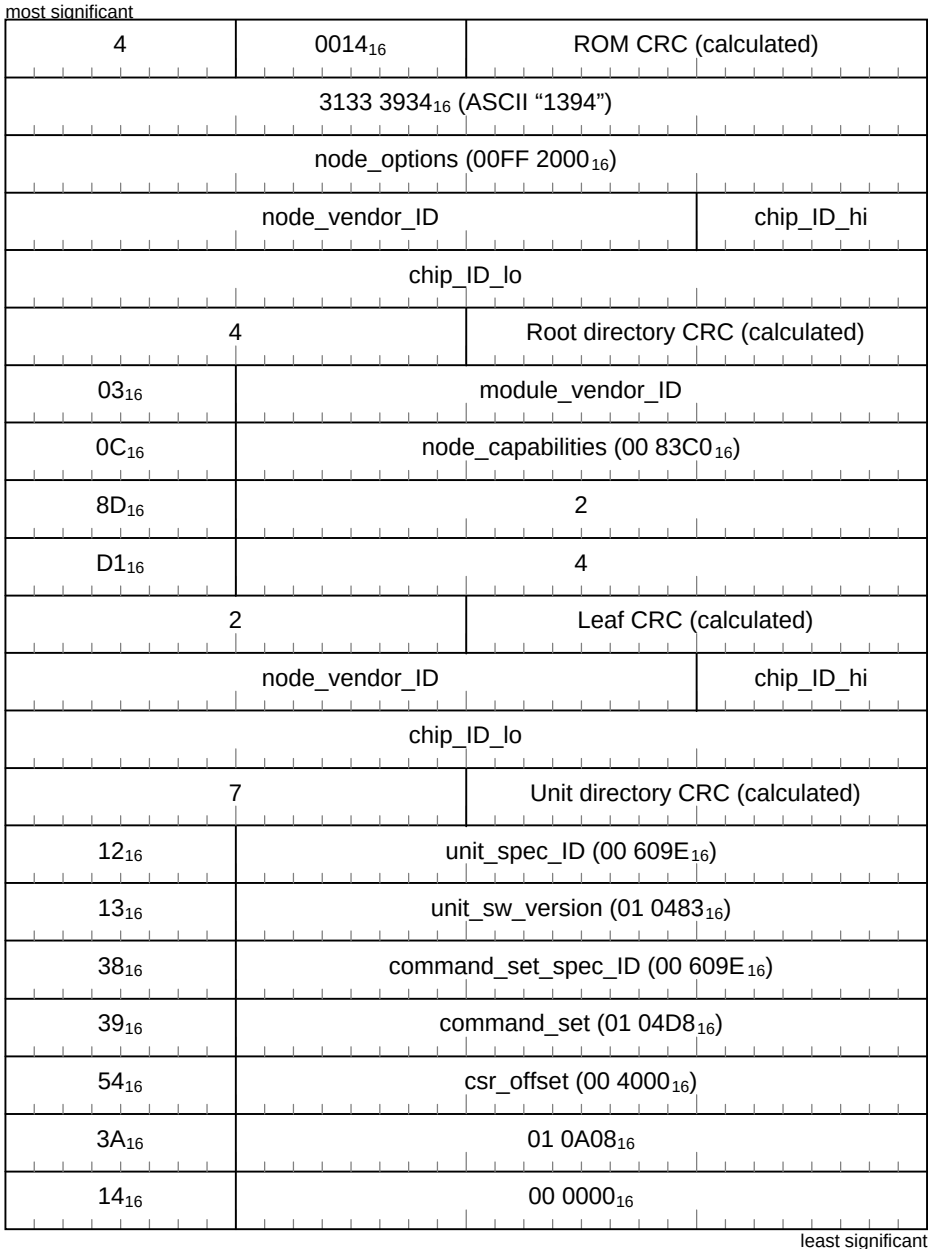

Figure Q.4 – Sample SCSI configuration ROM

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

Q.4.4.1 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.

Q.4.4.2 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.

Q.5 SCSI Architecture Model

This clause provides information useful to systems implementers: it relates the facilities provided by SBP-2 to the terminology used by the SCSI Architecture Model.

Q.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.

Q.5.2 Status

SCSI status is reported in the *status* field, which is part of the status block defined in Q.2.

Q.5.3 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.

Q.5.3.1 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.

Q.5.3.2 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.

Q.5.3.3 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 Q.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

Q.5.3.4 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 Q.1 – SAM-2 Service responses

.

Q.5.4 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.

Q.5.4.1 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.

Q.5.4.2 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.

Q.5.4.3 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.

Q.5.4.4 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.

Q.5.5 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.

Q.5.6 Asynchronous event reporting

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

Q.5.7 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.

Q.5.8 Hard reset

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

Q.5.9 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 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 Q.1 for the numeric values that encode the service responses.

Q.5.9.1 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.

Q.5.9.2 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 MANAGEMENT_AGENT register, the target may fetch the request from initiator memory.

Q.5.9.3 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.

Q.5.9.4 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*.