High Performance Transport - HPT (Draft)

Revision 0.3e

September 9, 1997

Shigeru Ueda

Akihiro Shimura

Contact e-mail address: oid3-1394@pure.cpdc.canon.co.jp

CANON INC.

Contents

Tables

Figures

1. Overview

1.1 Scope

This document defines the command set protocol for transporting data between computer system and peripheral device on top of Serial Bus Protocol 2 (SBP-2). The SBP-2 is a transport protocol defined for IEEE Std 1394-1995, Standard for a High Performance Serial Bus.

This document defines the following attributes and features, required to interface devices via the standard IEEE Std 1394/SBP-2 mechanisms:

Command block and status block formats

Task management and behavior model

1.2 Purpose

The design of the HPT was intended to meet the following objectives:

a) *High Performance and low overhead*. The protocol should efficiently utilize the high bandwidth of IEEE 1394 and processing resources on devices.

b) *Flexible bi-directional data transport*. The protocol should enable full duplex communication between the devices.

c) *Multiple service channels*. The protocol should provide multiple communication channels between the devices.

d) *Application independence*. The protocol should be independent of application and should not depend on a particular device class or control set.

e) *Backward compatibility with bus environment support*. The protocol should cover the application that conventional point-to-point interface like a parallel port covers. The protocol should also provide additional support to take the advantage of the bus environment features like device sharing for such application.

2. References

This document shall be used in conjunction with the following publications. When they are superseded by an approved revision, the revision shall apply:

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

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

ANSI X3T10 1155D, Serial Bus Protocol 2 (SBP-2)

3. Definitions

3.1 Conformance

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

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

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

3.1.3 shall: A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products conforming to this document.

3.2 Glossary

The following terms are used in this document:
(Omitted at this point.)

4. Acronyms and abbreviations

The following are abbreviations that are used in this document.

HPT	High performance transport (this document itself)
ORB	Operation request block
SBP-2	Serial Bus Protocol 2

5. High Performance Transport(HPT) Model (informative)

High Performance Transport (HPT) defines a small command set and a behavior model on top of the Serial Bus Protocol 2 (SBP-2). The SBP-2 is a transport protocol defined for IEEE Std 1394-1995, Standard for a High Performance Serial Bus. The HPT provides full duplex communication capability between an initiator device and a target device.

This clause describes components of the HPT model. In addition to the information in this clause, users of this document should also be familiar with the SBP-2 and its normative references.

The HPT uses the data exchange mechanism provided by the SBP-2. The unsolicited status defined by the SBP-2 is used as an asynchronous data transfer request from the target to the initiator. The command block ORB (operation request block) works as a data transfer request from the initiator to the target as specified by the SBP-2. The HPT introduces a queuing model on both the initiator and the target to queue those requests. In order to achieve full duplex communication, the HPT controls the flow of those requests by using each queue. The flow control is similar to the credit flow control used in the IEEE 1284.4.

The HPT provides full duplex communication capability over single login with small additional resources by adopting the flow control based on the requests rather than on the transporting data itself. The benefits of the SBP-2 like high performance and low overhead are achieved via the features of SBP-2 like shared memory model and listed execution. The HPT also inherits these benefits by relaxing the synchronization between both ends with the queuing model.

5.1 Target Model

Figure 1 illustrates an example block diagram of a command block agent for the HPT target. The command block agent contains one command fetch agent, one **command pre-fetch queue** and two execution agents, called **QUEUED execution agent** and **IMMEDIATE execution agent** respectively.

When the command fetch agent fetches the normal command block ORB, the command fetch agent examines the parameter specified in the *command_block* field of the command block ORB. The command block agent dispatches the command block ORB to either of the **command pre-fetch queue** or **IMMEDIATE execution agent** according to the parameter. The **QUEUED execution agent** executes the commands queued in the **command pre-fetch queue** in order.

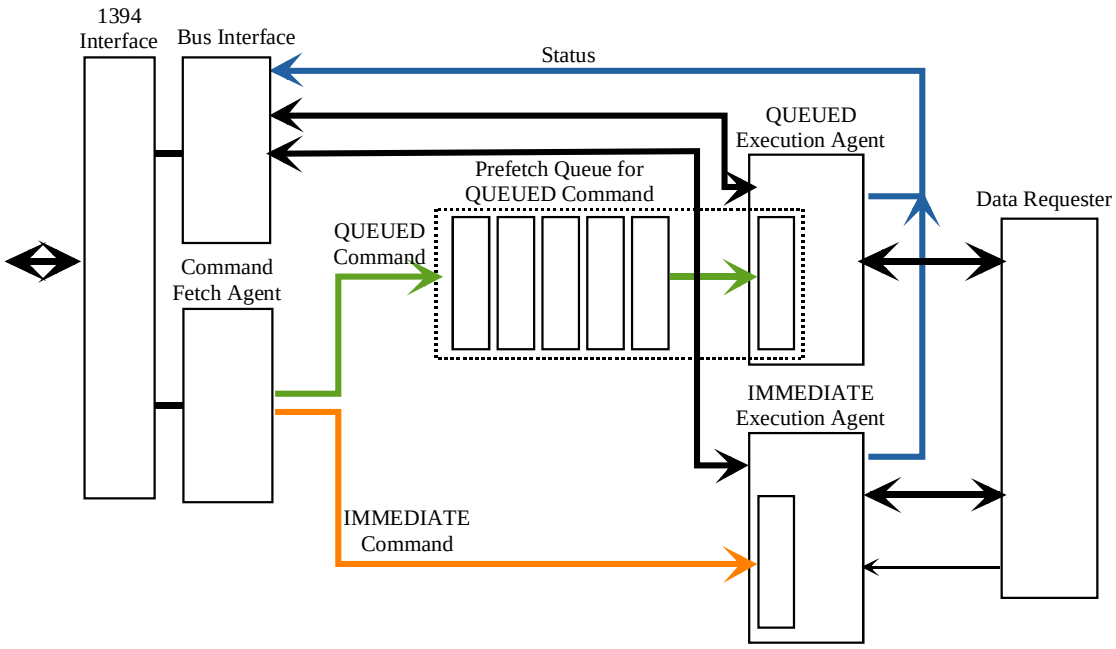

Figure 1 - Target Model

Each execution agent executes the dispatched command in order and independently of other agents. Each execution agent executes the data transfer associated with the command according to the parameters specified in the command.

The target stores a status block in initiator memory according to the value of the *notify* bit of the command block ORB after executing the command as specified by the SBP-2.

When the target has data to be sent to the initiator asynchronously, the target stores a **read request status** or a **direct status** in the initiator memory. The target stores one of these two status blocks depending on the size of the data to be sent. These two status blocks are the unsolicited status defined by the SBP-2.

The target stores a **read request status** in the initiator memory in case that the block length of the status including the data to be sent asynchronously exceeds the size of maximum status block length. The initiator creates a new **requested read command** ORB that requests the data transfer of the data to be sent by the target.

The target stores a **direct status** in the initiator memory in case that the block length of the status including the data to be sent asynchronously fits in the size of maximum status block length. The **direct status** contains the data to be sent in the *command set-dependent* field of the status block.

An asynchronous data transfer from the target to the initiator using the **read request status** requires actions both on the initiator and on the target. The initiator creates a **requested read command**

corresponding to the **read request status** from the target, and the target stores the data in the initiator memory according to the **requested read command.**

An asynchronous data transfer from the target to the initiator using the **direct status** does not require an action that the target stores the data in the initiator memory according to the status. A small data transfer to the initiator, like a simple status data response, is performed with one serial bus transaction to store a **direct status** block by the target and one **direct status response command** ORB from the initiator.

The target manages a constraint on newly storing a **read request status** or a **direct status** in the initiator memory.

The **status queue** in the initiator queues **read request status** blocks and **direct status** blocks stored by the target. The target restricts to store these status blocks newly in the manner that the number of the status blocks in process does not exceed the available depth of the **status queue** in the initiator.

In order to manage this constraint, the target retrieves the depth of the **status queue** from the initiator before starting a full duplex communication.

The target becomes aware that the initiator has consumed the content of the **status queue** by receiving the **requested read command** or **direct status response command** corresponding to the **read request status** or **direct status**.

5.2 Initiator Model

The initiator of the HPT has status queue as illustrated below.

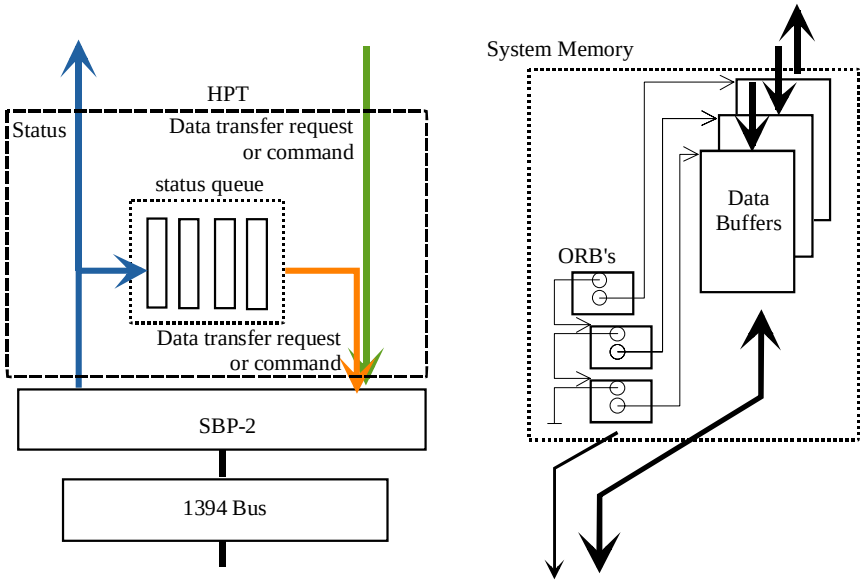

Figure 2 - Initiator Model

The initiator manages a constraint on appending a new **task** to a current **task set** by using the target model described in Section 5.1 in order to achieve a full duplex communication.

The **command pre-fetch queue** in the target queues the command ORB's destined to the **QUEUED execution agent**. The initiator restricts to append a new **task** destined to the **QUEUED execution agent** in the manner that the number of the commands destined to the **QUEUED execution agent** in the **task set** does not exceed the available depth of the **command pre-fetch queue** in the target.

In order to manage this constraint, the initiator retrieves the depth of the **command pre-fetch queue** from the target before starting a full duplex communication.

The initiator creates a command that specifies the **command pre-fetch queue** as a destination in case of the data transfer from the initiator to the target. The data transfer is initiated by the r**ead request status** or **direct status** from the target in case of the data transfer from the target to the initiator.

The initiator queues the **read request status** or **direct status** stored by the target in the **status queue**. Then, the initiator performs a write transaction to the UNSOLICITED_STATUS_ENABLE register in the target to response the reception of the status.

The initiator takes a status out from top of the **status queue**, and appends the **requested read command** ORB or **direct status response command** ORB, that specifies the **IMMEDIATE execution agent** as a destination, to the current **task set** according to the status.

The initiator becomes aware that the target has consumed the content of the **command pre-fetch queue** by receiving the status block specified by the SBP-2 corresponding to the command destined to the **command pre-fetch queue**.

Note 1: Storing the **read request status** and **direct status** requires the handshake using UNSOLICITED_STATUS_ENABLE register as specified in the SBP-2 because these status blocks are unsolicited. The asynchronous data transfer method described in this clause may not be suitable for the high frequency usage of these status blocks because this handshake brings with it a response overhead. In such a case, the initiator should use the **data transfer request command** ORB instead that specifies the **command pre-fetch queue** as a destination and requests the data transfer from the target to the initiator (This is the synchronous way originally provided by the SBP-2).

Note 2: It is possible for the initiator to append a command ORB that specifies the **IMMEDIATE execution agent** as a destination other than the **requested read command** or **direct status response command**. In such a case, the initiator shall be responsible for the constraint that the command destined to the **IMMEDIATE execution agent** should be consumed by the target in an appropriately finite period.

5.3 Multiple Logical Channel

This document defines two commands that open and close logical channels on single login of the SBP-2. The channel identifier specified in the command block and status block is used to multiplex/de-multiplex each channel data. Each channel is controlled by the model described in section 5.1 and section 5.2.

5.4 Device Control and Status (tentative)

This document also defines the two commands to acquire and release target device resource. Device like a printer may only have limited printing resources to print a document. In order target device to manage its resource contention, the initiator shall acquire the target device resource before requesting the execution that requires such resources. The initiator shall release the target device resource after the execution completed. These commands enable to handle acceptance or denial for the resource request at this layer, and the denial does not require an interaction with upper layers.

For example, the initiator acquires printing resources before submitting a printing job to a printer. After the printing job was finished, the initiator releases the printing resources. If another printing job is in progress on the printer, the initiator receives denial and may wait for acceptance.

This document defines one status related to a device class. The basic device status provides primary status information that is commonly used within the device class. This status is useful in case that the initiator does not have a capability to fully handle control set dependent device status, or the control set

implemented on the target does not define such device status.

6. Data Structure (tentative)

There are two classes of data structures defined by HPT:

- operation request blocks (ORB's);
- status blocks.

Note: The data structures described in this clause may be neither concrete nor consistent at this point. The description intends to help understanding how HPT works and what HPT appears.

6.1 Normal command block operation request blocks (ORB's) (tentative)

The format of the normal command block ORB is illustrated by the figure below.

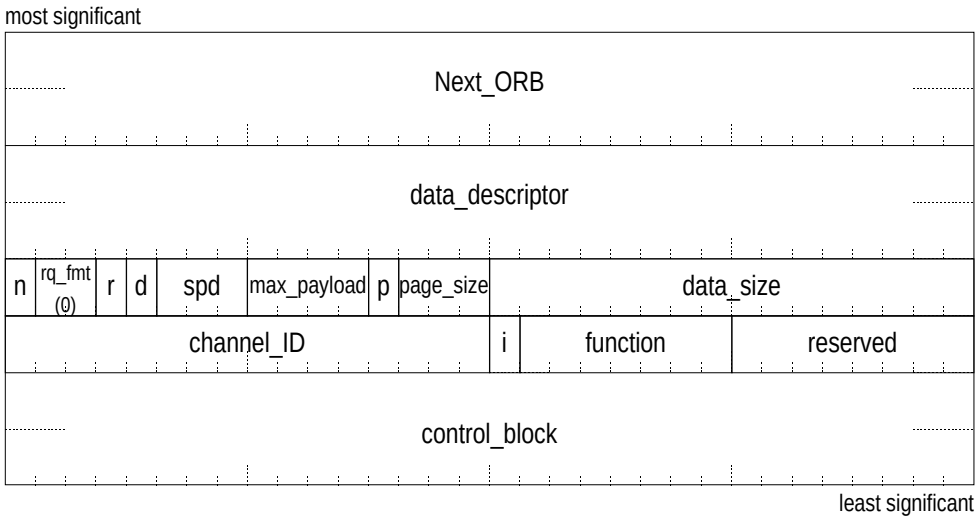

Figure 3 - Normal command block ORB

The *channel_ID* field specifies the communication channel identifier.

The *immediate* bit (abbreviated as *i* in the figure above) specifies destination of the command. If the *immediate* bit is zero, the command shall be destined for the command pre-fetch queue of the target. Otherwise, when the *immediate* bit is one, the command shall be destined for the immediate execution agent of the target.

The *function* field specifies the HPT command requested or responded, as defined by the tables below.

HPT commands defined in this document are divided into three categories.

The first category includes the commands those are used to achieve full duplex communication, and defined by the table below. The commands in this category shall be originated by the initiator.

Value	HPT command
0	reserved for future standardization
1	DATA TRANSFER
2	REQUESTED READ
3	DIRECT STATUS RESPONSE
4-7	reserved for future standardization

Table 1 - HPT commands (first category)

The second category includes the commands that are used to control device, and defined by the table below. The commands in this category shall be originated by the initiator.

Value	HPT command
8	ACQUIRE DEVICE RESOURCE
9	RELEASE DEVICE RESOURCE
A_{16}	ABDICATE DEVICE RESOURCE RESPONSE
B_{16}	BASIC DEVICE STATUS
C_{16}-F_{16}	Reserved for future standardization
10_{16}-$6F_{16}$	Control set dependent

Table 2 - HPT commands (second category)

The third category includes the commands those are used to control multiple channel, and defined by the table below. The commands in this category may be originated by the initiator or the target.

Value	HPT command
70_{16}/71_{16}	OPEN CHANNEL request/response
72_{16}/73_{16}	CLOSE CHANNEL request/response
74_{16}-$7F_{16}$	reserved for future standardization

Table 3 - HPT commands (third category)

The format of the *control_block* field is uniquely determined by a combination of *function* field, the control set implemented by the target. This document specifies those parts of the ORB that are invariant across target control sets.

All other fields are described by the SBP-2 standard.

6.1.1 Data transfer command ORB (tentative)

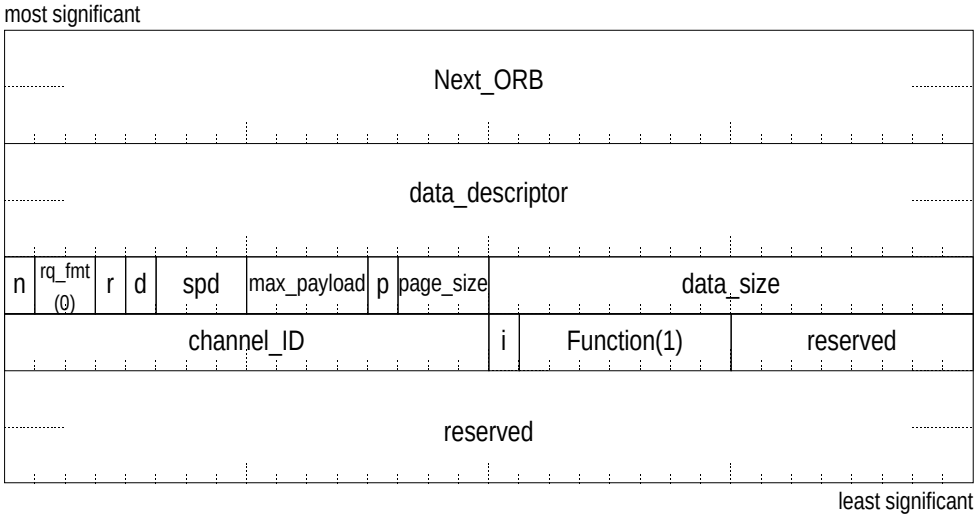

Figure 4 - Data transfer command ORB

The *data_descriptor* field, *page_table_present* bit (abbreviated as *p* in the figure above), *page_size* field and *data_size* shall specify the data buffer as specified in SBP-2.

The *direction* bit (abbreviated as *d* in the figure above) specifies direction of data transfer for the buffer as specified in SBP-2.

The initiator shall use REQUESTED READ command instead if the read request corresponds to a READ REQUEST status.

6.1.2 Requested read command ORB (tentative)

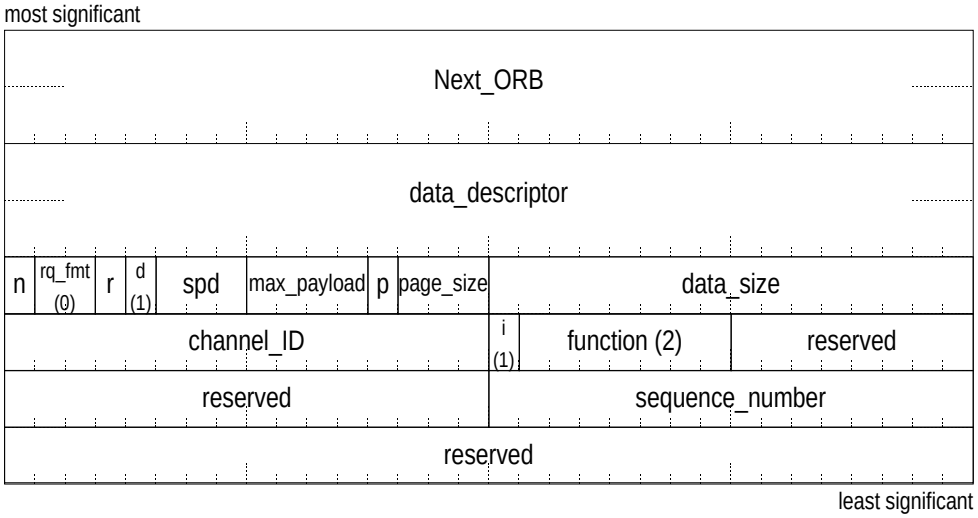

Figure 5 - Requested read command ORB

The *data_descriptor* field, *page_table_present* bit (abbreviated as *p* in the figure above), *page_size* field and *data_size* shall specify the data buffer as specified in SBP-2.

The *direction* bit (abbreviated as *d* in the figure above) shall be one.

The *immediate* bit (abbreviated as *i* in the figure above) shall be one.

The *sequence_number* field shall specify a value obtained from the field in the corresponding READ REQUEST status.

The initiator may specify a data buffer size smaller than the size requested by the target in the corresponding READ REQUEST status. The target shall manage the size of data that already transferred to the initiator. The initiator shall manage the size of data buffers that requests to the target to transfer.

The initiator shall use DIRECT STATUS RESPONSE command instead if the read request should be discarded by the target.

6.1.3 Direct status response command ORB (tentative)

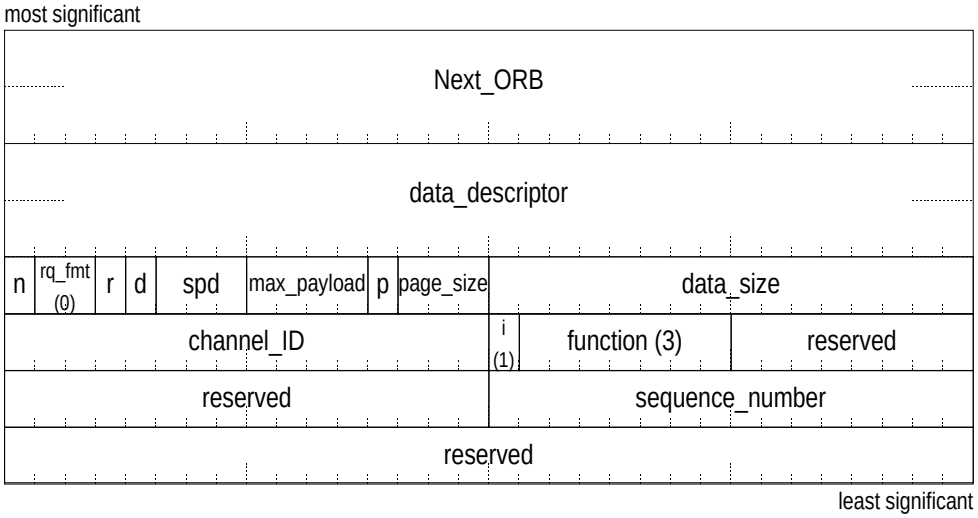

Figure 6 - Direct status response command ORB

The *immediate* bit (abbreviated as *i* in the figure above) shall be one.

The *sequence_number* field shall specify a value obtained from the field in the corresponding READ REQUEST status or DIRECT status.

If the READ REQUEST status has the same *sequence_number* value (i.e., this command ORB corresponds to a READ REQUEST status), the target shall discard the request and its associated data.

6.1.4 Acquire device resource request/response (tentative)

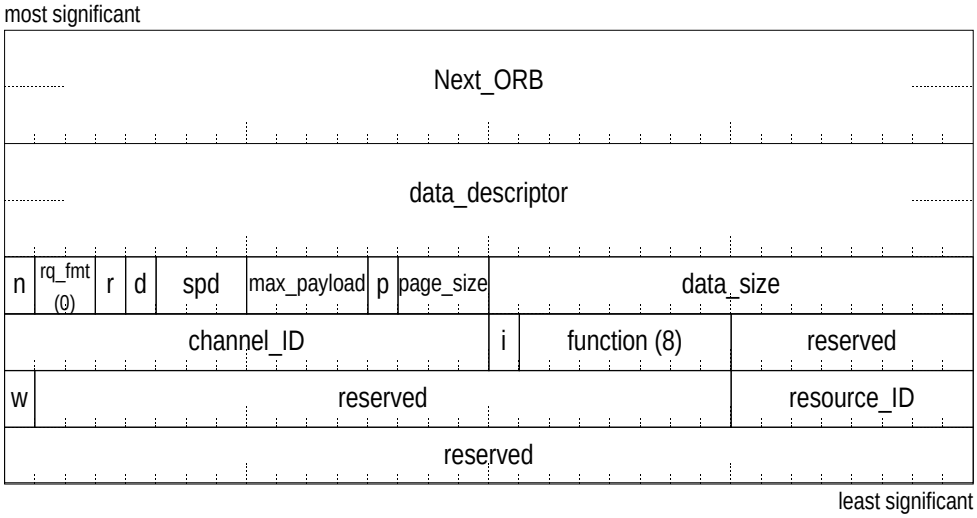

Figure 7 - Acquire device resource command ORB

The *wait* bit (abbreviated as *w* in the figure above) specifies whether the initiator waits successful acquisition or not. If the *wait* bit is zero, the target completes this command regardless of the result. Otherwise, if the *wait* bit is one, the target completes this command when the acquisition is succeeded.

The *resource_ID* field specifies resource identifier that identifies the target's resource to be acquired, as defined by the table below.

Value	Device resource
0	device class and service dependent
1-F_{16}	Reserved for future standardization
10_{16}-$7F_{16}$	Control set dependent

Table 4 - Device resource

The target shall manage contention of the resource specified in the *resource_ID* field.

The value "0" specifies a printing resource in case that the device class is printer and the service is printing service.

Note: Device discovery or service discovery will define the device class or the service. These definitions are TBD.

6.1.5 Release device resource command ORB (tentative)

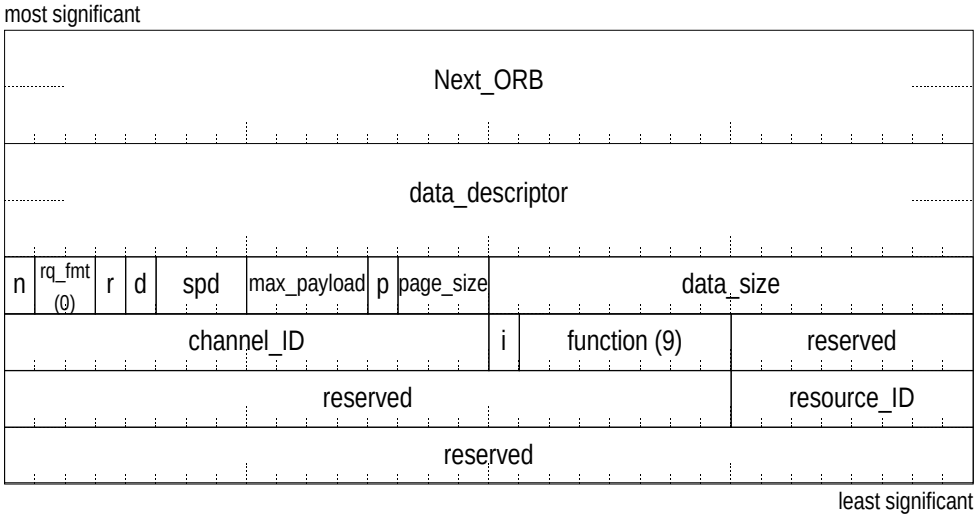

Figure 8 - Release device resource command ORB

The *resource_ID* field shall specify a value specified in the previous ACQUIRE DEVICE RESOURCE command ORB.

6.1.6 Abdicate device resource response command ORB (tentative)

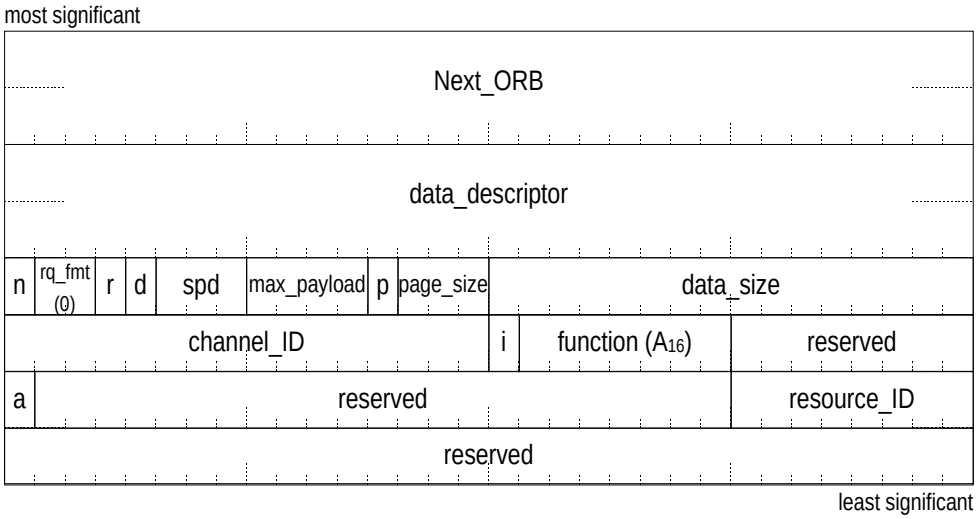

Figure 9 - Abdicate device resource response command ORB

The *accept* bit (abbreviated as *a* in the figure above) specifies whether the initiator accepts the

abdication request or not. If the *accept* bit is zero, the initiator denies the request. Otherwise, if the *accept* bit is one, the initiator accepts the request.

The *resource_ID* field shall specify a value specified in the corresponding ABDICATED DEVICE RESOURCE status.

6.1.7 Basic device status command ORB (tentative)

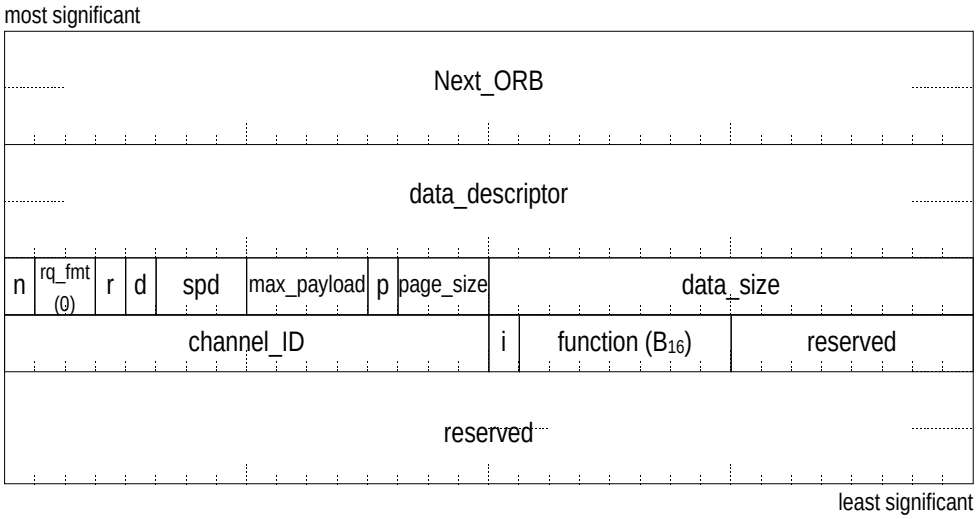

Figure 10 - Basic device status command ORB

This ORB requests a BASIC DEVICE status block to the target.

6.1.8 Open channel request/response (tentative)

The request format of the *control_block* field is illustrated by the figure below.

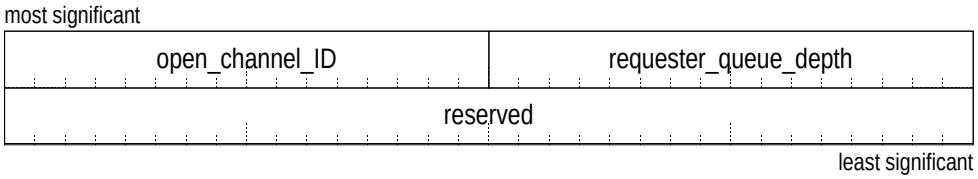

Figure 11 - Open channel request

The response format of the *control_block* field is illustrated by the figure below.

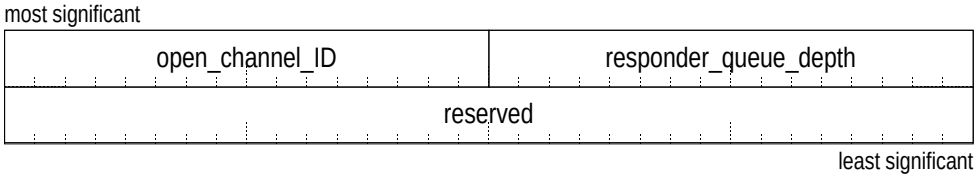

Figure 12 - Open channel response

The *open_channel_ID* field shall specify the identifier of the new channel.

The *requester_queue_depth* and *responder_queue_depth* field shall specify the number of queue entries sender has.

Before a requester may signal any other request to a responder on the new channel, the requester shall complete a queue depth negotiation that uses this *control_block.*

If the responder fails to allocate queue, the responder shall respond with *responder_queue_depth* equal to zero.

6.1.9 Close channel request/response (tentative)

The request format of the *control_block* field is illustrated by the figure below.

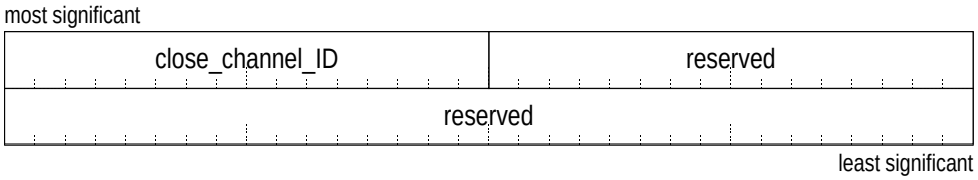

Figure 13 - Close channel request

The response format of the *control_block* field is illustrated by the figure below.

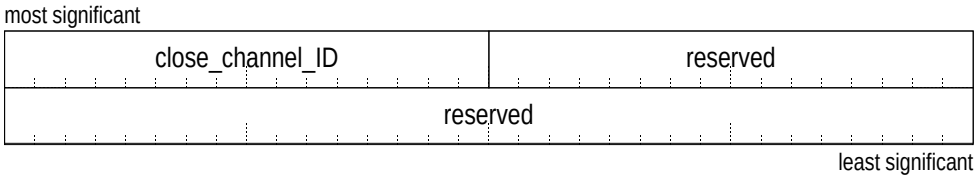

Figure 14 - Close channel response

The *close_channel_ID* field shall specify the identifier of the closing channel.

6.2 Status block (tentative)

The format of the status block is illustrated by the figure below.

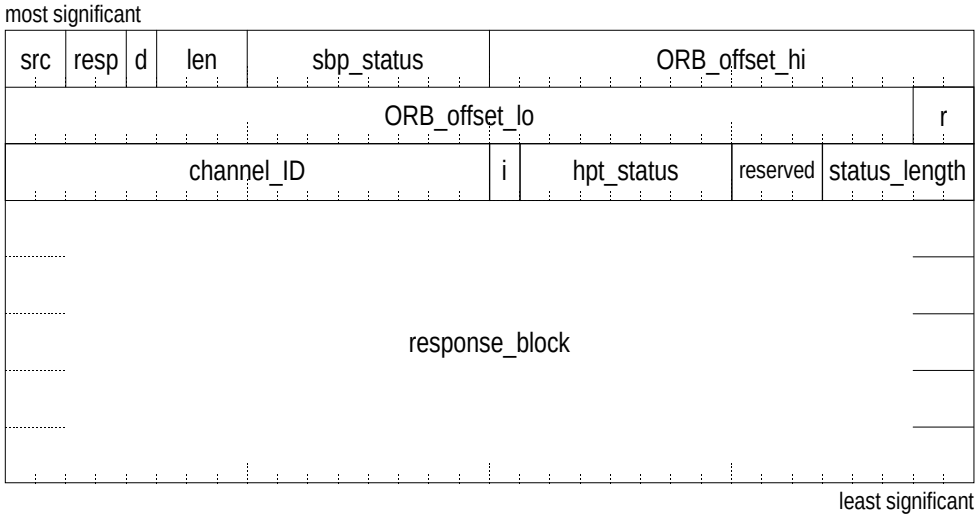

Figure 15 - Status block

The *channel_ID* field specifies the communication channel identifier.

The *immediate* bit (abbreviated as *i* in the figure above) shall indicate the source of the status. If the *immediate* bit is zero, the status has been generated by QUEUED execution agent. Otherwise, when the *immediate* bit is one, the status has been generated by IMMEDIATE execution agent.

The *hpt_status* field specifies the HPT status or request, as defined by the tables below.

HPT status defined in this document are divided into three categories.

The first category includes the status those are used to achieve full duplex communication, and defined by the table below. The status in this category shall be originated by the target.

Value	HPT status
0	reserved for future standardization
1	reserved for future standardization
2	READ REQUEST
3	DIRECT STATUS
4-7	reserved for future standardization

Table 5 - HPT status (first category)

The second category includes the status that is used to control device, and defined by the table below. The status in this category shall be originated by the target.

Value	HPT command
8	ACQUIRE DEVICE RESOURCE response
9	reserved for future standardization
A_{16}	ABDICATE DEVICE RESOURCE request
B_{16}	BASIC DEVICE STATUS
C_{16}-F_{16}	Reserved for future standardization
10_{16}-$6F_{16}$	Control set dependent

Table 6 - HPT status (second category)

The third category includes the status those are used to control multiple channel, and defined by the table 3. The status in this category shall be originated by the initiator or the target.

The *status_length* field shall specify the size, in bytes, of *command set-dependent* field in the status block.

The format of the *response_block* field is uniquely determined by a combination of *hpt_status* field, the control set implemented by the target. This document specifies those parts of the status block that are invariant across target control sets.

All other fields are described by the SBP-2 standard.

6.2.1 Read request status (tentative)

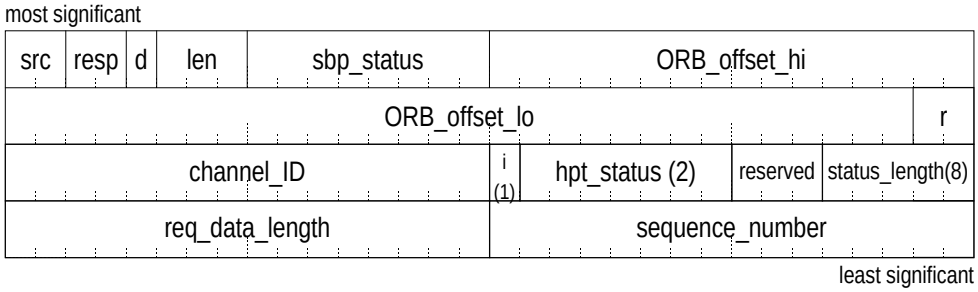

Figure 16 - Read request status

The *immediate* bit (abbreviated as *i* in the figure above) shall be one.

The *req_data_length* field shall specify the number of byte to be read by the initiator with REQUESTED READ command.

The *sequence_number* shall specify unique sequence number for each status. The *sequence_number* field shall be unique within every READ REQUEST and DIRECT status on the way in a channel.

This status is an unsolicited status.

6.2.2 Direct status (tentative)

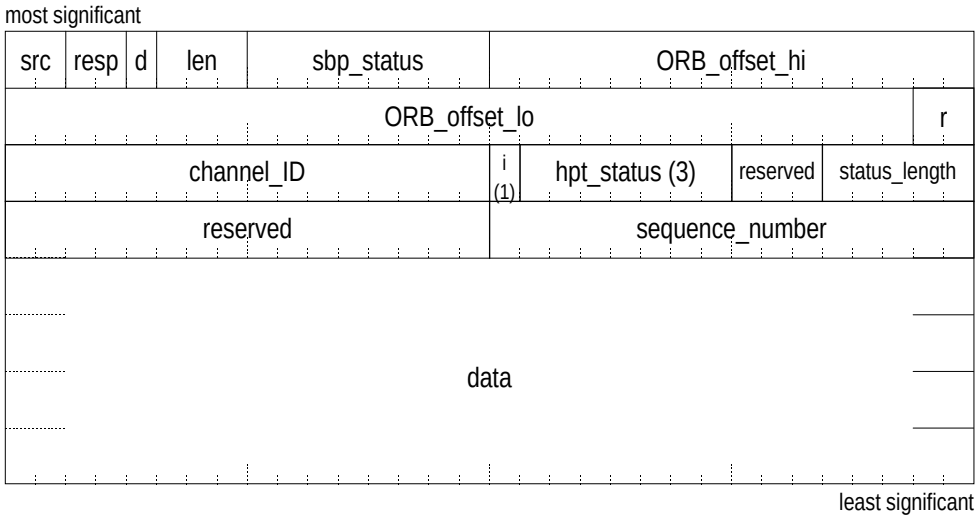

Figure 17 - Direct status

The *immediate* bit (abbreviated as *i* in the figure above) shall be one.

The *sequence_number* shall specify unique sequence number for each status. The *sequence_number* field shall be unique within every READ REQUEST and DIRECT status on the way in a channel.

The *data* field contains the data to be transferred to the initiator.

The *status_length* field shall specify the *data* length in byte. The value shall be (length of *data* field + 8).

This status is an unsolicited status.

6.2.3 Acquire device resource response status (tentative)

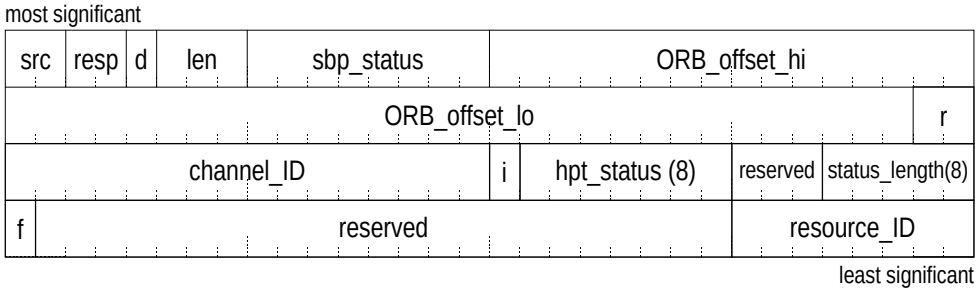

Figure 18 - Acquire device resource response status

The *fail* bit (abbreviated as *f* in the figure above) specifies the response to the ACQUIRE DEVICE

RESOURCE command from the initiator. If the *fail* bit is zero, the acquisition request is accepted. Otherwise, if the *fail* bit is one, the acquisition request is denied.

This status is unsolicited status if *wait* bit of corresponding command was zero and the acquisition failed.

The *resource_ID* field shall specify the value specified in corresponding ACQUIRE DEVICE RESOURCE command.

6.2.4 Abdicate device resource status (tentative)

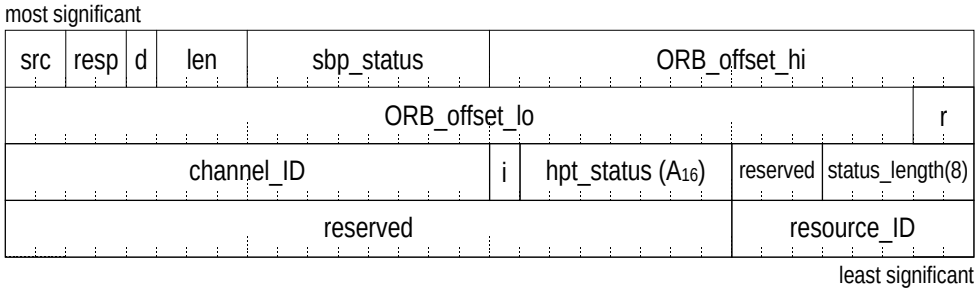

Figure 19 - Abdicate device resource status

The *resource_ID* field shall specify the *resource_ID* that acquired by previous ACQUIRE DEVICE RESOURCE command.

This status is an unsolicited status.

6.2.5 Basic device status (tentative)

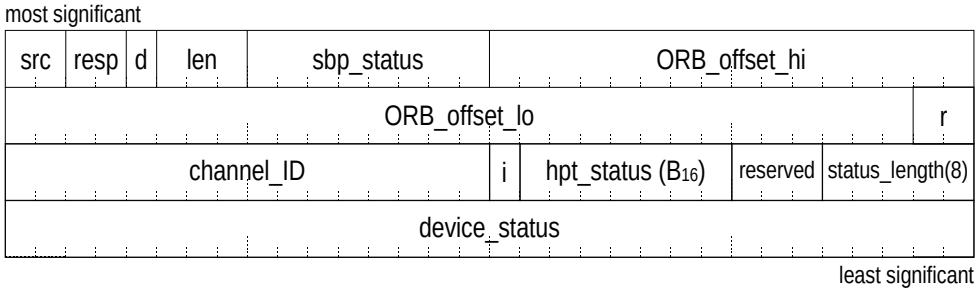

Figure 20 - Basic device status

The *device_status* field shall specify the current device status. The device status is device class dependent.

Note: the device status encoding for the printer device is TBD. The encoded device status will show the primary status like, warming up, ready, printing, out of paper, paper jam, out of toner (or ink),

door open, error, and so on.

This status is unsolicited status unless requested by the BASIC DEVICE STATUS command.

6.2.6 Open channel request/response (tentative)

The request format of the first two quadlets of *response_block* field is illustrated by the figure 11.

The response format of the first two quadlets of *response_block* field is illustrated by the figure 12.

The request status is unsolicited status.

6.2.7 Close channel request/response (tentative)

The request format of the first two quadlets of *response_block* field is illustrated by the figure 13.

The response format of the first two quadlets of *response_block* field is illustrated by the figure 14.

The request status is unsolicited status.

7. Control and Status Registers

The control and status registers (CSR's) implemented by a target shall conform to the requirements defined by the SBP-2 and its normative references. Isochronous capabilities are optional and not required for HPT targets.

8. Configuration ROM

The configuration ROM implemented by a target shall conform to the requirements defined by the SBP-2 and its normative references. A target shall implement entries defined by the SBP-2 as described in the clauses that follow.

8.1 Command_Set_Spec_ID entry

The Command_Set_Spec_ID entry is an immediate entry in the unit directory or the logical unit directory, that specifies the organization responsible for the command set definition for the target. Figure 21 shows the format of this entry.

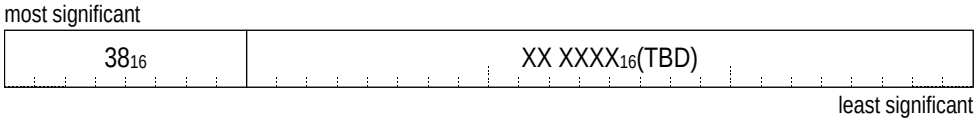

Figure 21 - Command_Set_Spec_ID entry format

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

XX $XXXX_{16}$(TBD) is the *command_set_spec_ID* value, an organizationally unique identifier, obtained from the IEEE/RAC by the organization responsible for the command set definition in this document.

8.2 Command_Set entry

The Command_Set entry is an immediate entry in the unit directory or the logical unit directory that, in combination with the *command_set_spec_ID*, specifies the command set implemented by the target. Figure 22 shows the format of this entry.

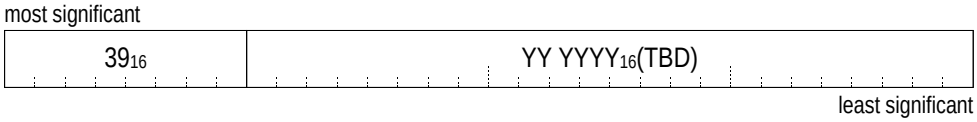

Figure 22 - Command_Set entry format

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

YY $YYYY_{16}$(TBD) is the *command_set* value that indicates that the target conforms to this document.

8.3 Command_Set_Revision entry

The Command_Set_Revision entry is an immediate entry in the unit directory or the logical unit directory that specifies the revision level of the command set implemented by the target. Figure 23 shows the format of this entry.

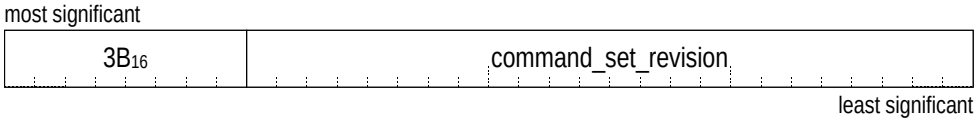

Figure 23 - Command_Set_Revision entry format

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

The meaning of *command_set_revision* is TBD.

8.4 Logical_Unit_Characteristics entry

The Logical_Unit_Characteristics entry is an immediate entry in the unit directory or the logical unit directory that specifies characteristics of the target implementation. Figure 24 shows the format of this entry.

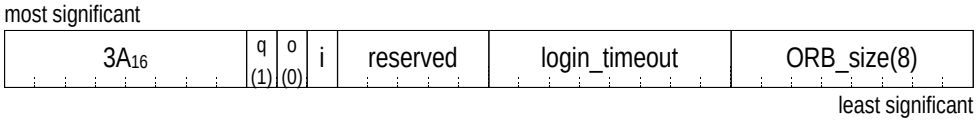

Figure 24 - Logical_Unit_Characteristics entry format

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

The target implements the task management (queuing) model described in this document. The target executes and reports completion status without any ordering constraints.

The *isochronous* bit (abbreviated as *i* in the figure above) specifies whether or not the target supports isochronous operations as specified by SBP-2.

The *login_timeout* field shall specify the maximum time an initiator allows for a target to store a status block in response to the initiator's login request. The setting of the *login_timeout* field is implementation specific.

The target uses the 8 quadlets (32 bytes) fetch size to obtain ORB's from initiator memory.

8.5 Logical-Unit-Number entry

The Logical_Unit_Number entry is an immediate entry in the unit directory or the logical unit directory that specifies the peripheral device type and number of logical unit implemented by the target. Figure 25

shows the format of this entry.

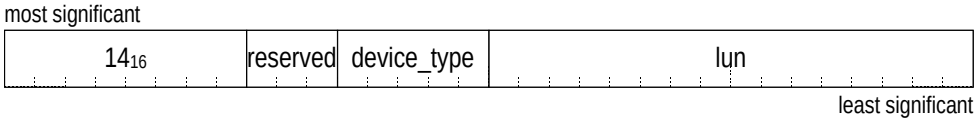

Figure 25 - Logical_Unit_Number entry format

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
0-1	Reserved for future standardization
2	Printer device
3-$1E_{16}$	Reserved for future standardization
$1F_{16}$	Unknown device type; function discovery mean will be used to determine the peripheral device type.

Table 7 - Peripheral device type

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

9. Function Discovery

Function Discovery implemented by a target shall conform to the requirements defined by the Function Discovery specification.

Note: the Function Discovery is under work at this point. The definitions required by the HPT are TBD.