CONGRUENT SOFTWARE, INC.
98 Colorado Avenue
Berkeley, CA 94707
(510) 527-3926
(510) 527-3856 FAX

FROM: Peter Johansson

TO: T10 SBP-3 working group

DATE: July 24, 2003

RE: SBP-3 isochronous data transfer greater than 64 kiB

On February 21, 2003, Andy Green sent a message to the SBP-3 working group reflector in which he observed:

> I entirely agree with the proposal [T10/03-090r1] for using existing command sets within isoc ORBs; however the cited example (an isoc DVD player using MMC contained within isoc command ORBs) may be impracticable, as the *data_size* is limited to 64 kiB per ORB. How about we make data_size a fixed multiplier if the isoc bit is one, say of 512 bytes, e.g. a data_size value of 0x10 = transfer 16x512 bytes = 8192 bytes.

On February 24, I replied:

> Andy, I'm not sure it would be a good idea to impose such a limitation on the transport protocol. For asynchronous transfers, we dodged this bullet by using a page table (even if no page table was, strictly speaking, necessary) to increase the overall transfer length past 64 KiB. For isochronous ORBs, two other solutions come to mind:
>
> a) When the *isochronous* bit is one, redefine the *spd*, *max_payload*, *page_table_present* and *page_size* fields so as to allow a larger *data_size* field—up to 27 bits; or
>
> b) Permit the use of a page table when the *isochronous* bit is one—but the only valid information in the page table entries would be the segment lengths.

Additional discussion on reflector converged on solution a) above and it was the intent of the Editor to modify the SBP-3 working draft accordingly. Regrettably, the Editor overlooked the changes and prepared SBP-3 Revision 4 for initial public review without them. Andy Green noted this oversight in a message to the SBP-3 working group reflector on May 12.

The changes that follow are suggested for SBP-3 Revision 4a with the intention that a second public review be conducted on a revised draft. Since the SBP-3 working group is no longer meeting face-to-face, this matter will be the subject of a T10 plenary vote in September. SBP-3 working group members and other interested parties should comment *via* the reflector so that a consensus position (or lack thereof) may be reported to the T10 plenary.

5.2.3 Command block ORBs

Command block ORBs are used to encapsulate data transfer or device control commands for transport to the target. A target's command set and device type determine the length of these ORBs, which shall be fixed for a particular command set and device type. A target reports this size in its configuration ROM (see 7.8.10).

NOTE – Although device designers may select arbitrary ORB lengths, system considerations may favor some ORB sizes over others, *e.g.*, 32 bytes. An ORB size of 32 bytes limits the command set-dependent information in a command block ORB to twelve bytes. This is adequate for many command descriptor blocks defined in command sets such as SCSI, but device designers should not hesitate to utilize larger ORBs if 16-byte or larger commands are required. Operating systems designers should take care not to preclude the use of arbitrarily large ORBs.

Command block ORBs may have either one or two buffer descriptors that may specify isochronous or non-isochronous methods for data transfer. The format of a command block ORB with a single, non-isochronous buffer descriptor is illustrated by the figure below.

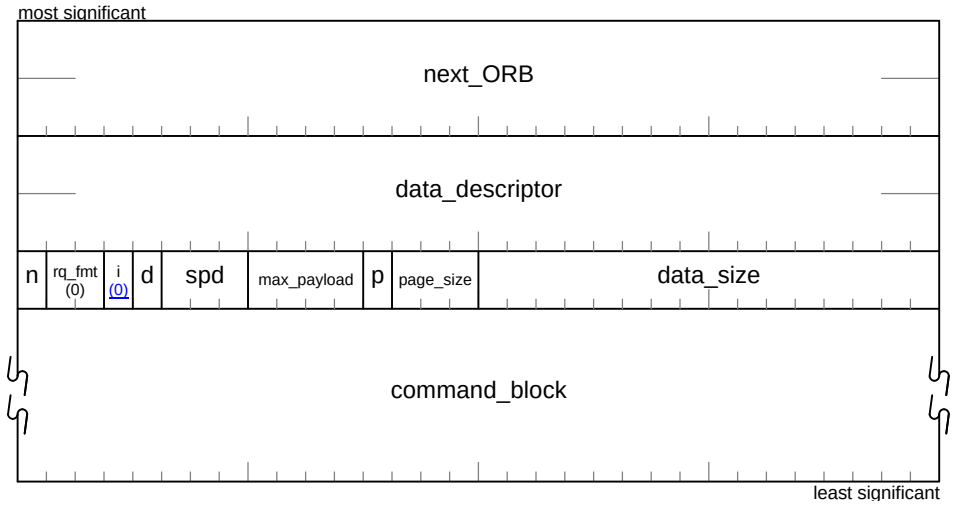

Figure 15 – Command block ORB (single, non-isochronous buffer descriptor)

The *next_ORB* field shall contain a null pointer or the address of a dummy ORB or a command block ORB and shall conform to the address pointer format illustrated by Figure 11.

The ~~value of the~~ *data_descriptor* field is ~~valid~~ defined only when the *isochronous* bit is zero and its value is valid only when *data_size* is nonzero, in which case it shall contain either the address of the data buffer or the address of a page table that describes the memory segments that make up the data buffer, dependent upon the value of *page_table_present* bit. The format of the *data_descriptor* field, when it directly addresses a data buffer, shall be a 64-bit Serial Bus address or, when it addresses a page table, shall be as specified by Figure 10. When *data_descriptor* specifies the address of a page table, the format of the page table shall conform to that described in 5.3.

The *notify* bit and *rq_fmt* field are as previously defined for all ORB formats. The *rq_fmt* field shall be zero for an ORB ~~which~~ that contains a single buffer descriptor.

The value of the *isochronous* bit (abbreviated as *i* in the figure above) is valid only when *data_size* is nonzero, in which case this it specifies the transfer method used for data associated with the ORB. When this bit is zero, Serial Bus read or write transactions shall be used to move data to or from the buffer

described by the *data_descriptor*, *spd*, *max_payload*, *page_table_present*, *page_size* and *data size* fields. Otherwise, when *isochronous* is one (see Figure 15a), data transfer shall be effected by Serial Bus isochronous streams, *i.e.*, packets with a transaction code of A_{16}.

NOTE – Command set-dependent methods may be used to specify isochronous data transfer even if the *isochronous* bit is zero. ~~See Annex D for an example.~~

The value of the *direction* bit (abbreviated as *d* in the figure above) is valid only when *data_size* is nonzero, in which case it specifies direction of data transfer. The meaning of the *direction* bit shall be interpreted in conjunction with the *isochronous* bit. If both the *isochronous* and the *direction* bits are zero, the target shall use Serial Bus read transactions to fetch data from system memory. When the *isochronous* bit is zero and the *direction* bit is one, the target shall use Serial Bus write transactions to store data in system memory. Otherwise, when the *isochronous* bit is one, a *direction* bit of zero specifies that the target shall receive isochronous data while a *direction* bit of one specifies that it shall transmit isochronous data.

The ~~value of the~~ *spd* field is ~~valid~~ defined only when the *isochronous* bit is zero and its value is valid only when *data_size* is nonzero, in which case it specifies the speed that the target shall use for data transfer transactions addressed to the data buffer or page table, as encoded by Table 1.

Table 1 – Data transfer speeds

Value	Speed
0	S100
1	S200
2	S400
3	S800
4	S1600
5	S3200
6 – 7	Reserved for future standardization

The ~~value of the~~ *max_payload* field is ~~valid~~ defined only when the *isochronous* bit is zero and its value is valid only when *data_size* is nonzero, in which case the maximum data transfer length is specified as $2^{max\_payload + 2}$ bytes, which is the largest data transfer length that may be requested by the target in a single Serial Bus read or write transaction addressed to the data buffer. The *max_payload* field shall specify a length less than or equal to the maximum asynchronous data payload specified by IEEE 1394 for the data transfer rate indicated by *spd*.

The ~~value of the~~ *page_table_present* bit (abbreviated as *p* in the figure above) is ~~valid~~ defined only when the *isochronous* bit is zero and its value is valid only when *data_size* is nonzero, in which case it shall be zero if *data_descriptor* directly addresses the data buffer, else one when *data_descriptor* addresses a page table.

The ~~value of the~~ *page_size* field is ~~valid~~ defined only when the *isochronous* bit is zero and its value is valid only when *data_size* is nonzero, in which case it shall specify the underlying page size of the data buffer memory (see 9.4 for an explanation of target responsibilities with respect to page boundaries). A *page_size* value of zero indicates that the underlying page size is not specified. Otherwise the page size is $2^{page\_size + 8}$ bytes. The page size applies to the data buffer whether or not a page table is present. When a page table is used to describe the data buffer, the *page_size* field also specifies the page table format. A *page_size* value of zero indicates an unrestricted page table (also known as a scatter/gather list) while a nonzero *page_size* indicates a normalized page table.

If the *isochronous* bit is zero and *page_table_present* is zero, the *data_size* field shall contain the size, in bytes, of the system memory addressed by the *data_descriptor* field. When the *isochronous* bit is zero and *page_table_present* is one, *data_size* shall contain the number of elements in the page table addressed by *data_descriptor*. Otherwise, the *isochronous* bit is one and *data_size* shall specify the maximum count, in bytes, of isochronous data to be transferred.

The *command_block* field contains information not specified by this standard.

As specified above, the *data descriptor*, *spd*, *max payload* and *page size* fields and the *page_table_present* bit are not present in a command block ORB whose *isochronous* bit is one. Some of these fields are reserved; the others are redefined to extend the *data_size* field to 24 bits. The format of a command block ORB with a single, isochronous buffer descriptor is illustrated by the figure below.

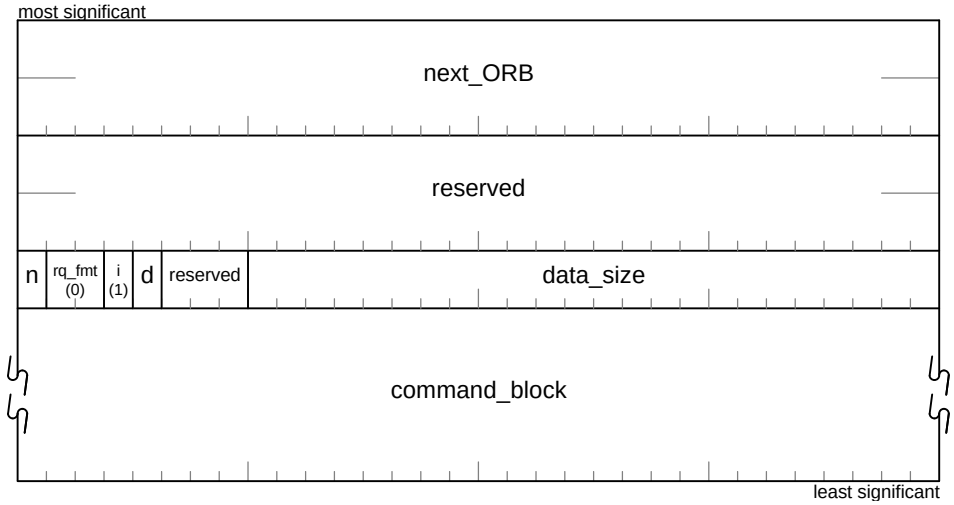

Figure 15a – Command block ORB (single, isochronous buffer descriptor)

When *rq_fmt* equals one, the ORB contains two buffer descriptors, as illustrated by Figure 16.

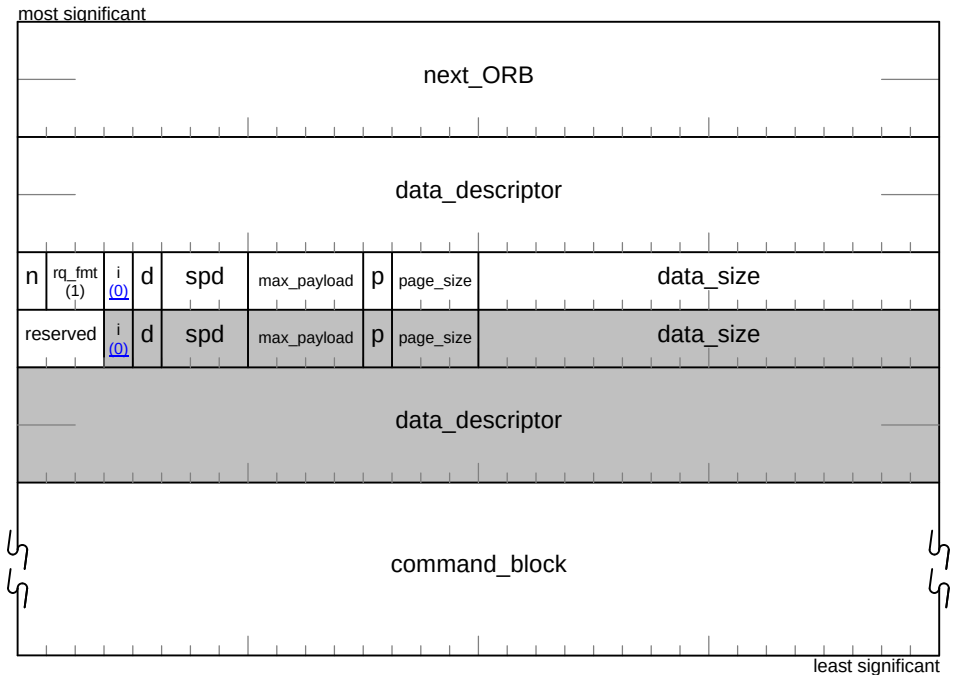

Figure 16 – Command block ORB (dual, non-isochronous buffer descriptors)

Each buffer associated with an ORB is described by a set of bits and fields: *data_descriptor*, *isochronous*, *direction*, *spd*, *max_payload*, *page_table_present*, *page_size* and *data_size*. A command block ORB whose *rq_fmt* is zero describes a single buffer or isochronous stream (referred to as *buffer*[0]) via the fields specified by Figure 15. When *rq_fmt* is one, the ORB includes two sets of these fields, capable of describing *buffer*[0] and *buffer*[1]. The fields that describe *buffer*[0] are in the same location as in a single buffer descriptor command block ORB; the additional fields (shown shaded in Figure 16) describe *buffer*[1]. The meaning of the individual buffer descriptor fields remains the same whether the field pertains to *buffer*[0] or *buffer*[1].

NOTE – Although not illustrated, either or both buffer descriptors may specify isochronous data transfer, in which case the pertinent fields are either reserved or redefined to extend the *data_size* field (see Figure 15a).

All of a buffer's characteristics are independent of the other buffer. Buffers need not reside in the same node nor be subject to the same speed or maximum payload characteristics. One buffer may be described by a page table and the other not. The matrix below illustrates how it is possible in all except two cases to determine buffer use from the information contained in the ORB. The two cases that require additional information (shown shaded in gray) occur when two buffers are described and the *direction* bit for both has the same value.

***data_size*[0]**	***direction*[0]**	***data_size*[1]** zero	***data_size*[1]** nonzero, ***direction*[1]** 0	***data_size*[1]** nonzero, ***direction*[1]** 1
zero		No buffers	*buffer*[1] outbound	*buffer*[1] inbound
nonzero	0	*buffer*[0] outbound	Both buffers outbound; Consult command set for details	*buffer*[0] outbound *buffer*[1] inbound
nonzero	1	*buffer*[0] inbound	*buffer*[0] inbound *buffer*[1] outbound	Both buffers inbound; Consult command set for details