Document: T10/01-067r0
Date: 02/12/2001
Author: Lance Flake (lance_flake@maxtor.com)
Title: RBC Access For AV/C Data Interchange in SBP-3

Overview

Modern 1394 HDD storage devices require support for two access procotols currently: SBP-2 with RBC commands (hereafter referred to as RBC), and AV/C with HDD subunit commands (hereafter referred to as AV/C). A current HDD supporting both protocols must partition the total storage area into two distinct components: RBC and AV/C. Neither protocol has knowledge of or access to the other protocol's partition.

The RBC partition is managed through the attached computer file system. It reads and writes extents of LBA's only through asynchronous 1394 transfers: no isochronous support exists in SBP-2.

For the AV/C partition the HDD device must provide its own file system for track management. Management functions use asynchronous 1394 transfers, while data recording and playback are accomplished through isochronous 1394 transfers. No LBA-based asynchronous access to the recorded data is provided.

This multi-protocol architecture has one distinctly unpleasant requirement for HDD manufacturers – partitioning. The HDD device must be sold with the total storage capacity divided between the RBC and AV/C partitions. The RBC Read Capacity command will only return the RBC partition size, and the AV/C capacity will only reflect the AV/C partition. While the manufacturer will produce an HDD with a certain total capacity, the customer will see only part of this capacity at any one time.

More importantly, the partitioning must be done once and does not change (ignoring the use of a cumbersome custom repartitioning utility program). The particular fractions of the total capacity assigned to each partition would ideally be done for a particular customer base. Internal HDD devices intended for computers could be biased towards the RBC partition size, while internal devices intended for AV/C equipment could be biased towards the AV/C partition size. These two sets of customers do not appear to present a problem when setting the partition sizes.

The partitioning problem truly presents itself when producing an external 1394 HDD device. There is no way to control if the device will be used primarily for computer or A/V purposes. It may in fact be used for both (and in the future most certainly will). In this case the partitioning requirement becomes a burden. The customer will face unnecessary capacity constraints and the manufacturer will be faced with an artificial customer support problem.

In the mixed computer and A/V environment, partitioning presents another problem. Currently RBC-based asynchronous access to the data for editing programs running on computers is prevented. Data tracks are created on the HDD through isochronous AV/C record operations. Post-record editing through asynchronous Read and Write commands would facilitate editing programs running on computers to access these recorded tracks, and possibly allow for the creation of new tracks as well.

Proposal

To eliminate the need for a partition requires a method for RBC and AV/C applications to share data. Starting with the fact that AV/C access requires the drive the manage tracks, the obvious solution is to provide RBC access to tracks as well. Where before a single volume was created on the computer for the single RBC partition, now one or more volumes can be created from a collection of tracks dedicated to non-A/V data. The computer user would have access to both computer specific data in those volumes and to A/V data on the other tracks.

This overall approach requires nothing new of the AV/C protocol, as it already has provisions for describing the format of each track. The HDD device would present tracks dedicated to RBC access having a foreign format for AV/C inquiries. The management of tracks within the HDD device would also remain essentially the same, but would allow RBC access in addition to AV/C access.

The goal is to add RBC access to the underlying tracks, not to the AV/C management data structures or command set. For that a computer would have to be truly AV/C capable. The minimum interchange capability would be for the track numbering to match between the two protocols and the new RBC access to read/write tracks' storage locations. Track numbers are 32-bit values, starting with zero.

In this new approach computer file systems would need to address extents of LBA's as subsections of tracks. New read and write RBC commands would allow track-relative LBA access. A computer volume could be made per track, or could be assembled from several tracks. Some part of the file system stack within the computer would need slight modification to add the track-aware capabilities, and the RBC driver would need additions for the track-relative commands.

With this new approach HDD manufacturers would not have to face the partitioning problem at all. Internal HDD devices intended for computers could be initialized to contain a single large RBC track, while internal devices intended for AV/C equipment could be initialized to have no tracks. External HDD devices could be initialized to have no tracks, and if attached to a computer could easily be "formatted" with a command to create an RBC track.

HDD manufacturers would have the option of supporting old RBC commands (track unaware). They could initialize the disk to contain an old RBC partition in addition to supporting new RBC track-aware access. This proposal is neutral to this particular strategy and simply adds an interchange-capable access mechanism to RBC.

New RBC Commands

This proposal would require two sets of new RBC commands for track-aware access. The first set would provide read/write access, and include Read Track and Write Track. The second set would provide a minimum of track management, and include List Tracks, Create Track, and Delete Track.

Read Track Command

The Read Track command (see Table 1) requests that the device transfer data to the initiator. The most recent data value written in the addressed logical block shall be returned.

Table 1- READ TRACK Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (xxh)							
1	Reserved							
2	(MSB)							
3	Logical Block Address							
4								
5								(LSB)
6	Reserved							
7	(MSB)			Transfer Length				
8								(LSB)
9	Control = 00h							
10	(MSB)							
11	Track Number							
12								
13								(LSB)
14	Reserved							
15	Reserved							

The LOGICAL BLOCK ADDRESS field specifies the first logical block of the range of logical blocks within the track that shall be read.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred. A TRANSFER LENGTH of zero indicates that no logical blocks shall be transferred. This condition shall not be considered an error. Any other value indicates that number of logical blocks that shall be transferred.

The TRACK NUMBER field specifies the track containing the logical block(s) requested for transfer.

Write Track Command

The Write Track command (see Table 2) requests that the device write data transferred from the initiator to the medium.

Table 2- WRITE TRACK Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (xxh)							
1	Reserved				FUA	Reserved		
2	(MSB)							
3	Logical Block Address							
4								
5								(LSB)
6	Reserved							
7	(MSB)							
8	Transfer Length							(LSB)
9	Control = 00h							
10	(MSB)							
11	Track Number							
12								
13								(LSB)
14	Reserved							
15	Reserved							

A FORCE UNIT ACCESS (FUA) bit of zero indicates that the device may satisfy the command by accessing the cache memory if the WCD bit in RBC mode page 06h is set to zero. For write operations, logical blocks may be transferred directly to the cache memory. GOOD status may be returned to the initiator prior to writing the logical blocks to the medium. Any error that occurs after GOOD status is returned as a deferred error.

A FUA bit of one indicates that the device shall access the media in performing the command prior to returning GOOD status. A WRITE command shall not return GOOD status until the logical blocks have actually been written on the media (i.e., the data is not write cached).

If the device supports write caching, FUA support shall be implemented. If write caching is NOT supported then the FUA bit may be ignored.

The LOGICAL BLOCK ADDRESS field specifies the first logical block of the range of logical blocks within the track that shall be written.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred. A TRANSFER LENGTH of zero indicates that no logical

blocks shall be transferred. This condition shall not be considered an error. Any other value indicates that number of logical blocks that shall be transferred.

The TRACK NUMBER field specifies the track containing the logical block(s) requested for transfer.

List Tracks Command

The List Tracks command (see Table 3) requests that the device transfer xx to the initiator.

Table 3- LIST TRACKS Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (xxh)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Control = 00h							

LIST TRACKS data (see Table 4) shall be returned to the initiator prior to sending GOOD status for the command.

Table 4- LIST TRACKS Data

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) Free Size							
1								
2								
3								(LSB)
4	(MSB) Number of Tracks							
5								
6								
7								(LSB)
8	Reserved							AV/C
9	Reserved							
10	(MSB) Track Size							
11								
12								
13								(LSB)
14	Other Tracks							
-								
N								

The FREE SIZE field returns the current number of logical blocks on the medium unallocated to any track. This number represents a dynamic quantity, as other initiators may be growing tracks or creating new tracks.

The NUMBER OF TRACKS field indicates the number of tracks stored on the medium and the number of 6-byte entries to follow in the list. The ordinal position of each 6-byte entry is the track number for the purposes of other commands.

For each track the list contains a 6-byte entry containing the AV/C and TRACK SIZE fields. The AV/C bit, when set, indicates that the track was created through the AV/C protocol. When cleared, the AV/C bit indicates that the track was created through the CREATE TRACK command described in this document. The TRACK SIZE field indicates the number of logical blocks currently owned by the track.

Create Track Command

The Create Track command (see Table 5) requests that the device create a new track object of a specific number of logical blocks.

Table 5- CREATE TRACK Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (xxh)							
1	(MSB)							
2	Minimum Size							
3								
4								(LSB)
5	(MSB)							
6	Maximum Size							
7								
8								(LSB)
9	Control = 00h							

The MINIMUM SIZE field instructs the device to not create a track with less logical blocks and not return GOOD status. The device must reject any CREATE TRACK command if the MINIMUM SIZE field is greater than the MAXIMUM SIZE field. This field allows the initiator to control the minimum granularity of tracks it supports.

The MAXIMUM SIZE field instructs the device to not create a track with more logical blocks. Any value greater than or equal to the current free space on the medium represents a request for the largest possible track size. This field allows the initiator to request a fixed track size when set to the MINIMUM SIZE value, or ask for all of the remaining space with an arbitrarily large value (e.g., 0xFFFFFFFF).

CREATE TRACK data (see Table 6) shall be returned to the initiator prior to sending GOOD status for the command.

Table 6- CREATE TRACK Data

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				Track Number				
2								
3								(LSB)
4	(MSB)							
5				Track Size				
6								
7								(LSB)

The TRACK NUMBER field returns the number of the track created.

The TRACK SIZE field returns the number of logical blocks of the track created.

Delete Track Command

The Delete Track command (see Table 7) requests that the device delete the track specified.

Table 7- DELETE TRACK Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (xxh)							
1	(MSB)							
2	Track Number							
3								
4								(LSB)
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Control = 00h							

The TRACK NUMBER field specifies the track to be deleted by the device. Deleting a track will renumber (subtracting one) all subsequent tracks on the medium.