Document: T10/01-067r1
Date: 03/23/2001
Author: Lance Flake (lance_flake@maxtor.com)
Title: RBC Access For AV/C Data Interchange in SBP-3

New RBC Commands and Annex

This proposal includes two sets of new RBC commands for object-aware access. The first set would provide read/write access, and include Read Object Relative and Write Object Relative. The second set would provide for management capabilities for RBC objects, and include Read RBC Object List, Create RBC Object, and Change RBC Object Size. After the commands is a proposed informative annex explaining their usage.

Read Object Relative Command

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

Table 1- READ OBJECT RELATIVE Command Descriptor Block

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (xxh)							
1	Reserved				FUA	Reserved		
2	(MSB)							
:	Logical Block Address							
9								(LSB)
10	(MSB)							
:	Transfer Length							
13								(LSB)
14	(MSB)							
:	Object ID							
17								(LSB)
18	Control = 00h							

A FORCE UNIT ACCESS (FUA) bit of zero indicates that the target may satisfy the command by accessing the cache. A FUA bit of one indicates that the target shall access the media in performing the command prior to returning GOOD status.

The LOGICAL BLOCK ADDRESS field specifies the first logical block of the range of logical blocks within the object 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, and this condition shall not be considered an error.

The OBJECT ID field specifies the object containing the logical block(s) requested for transfer. For RBC types of objects the OBJECT ID refers to an object created through the CREATE RBC OBJECT command. For all other types of objects the OBJECT ID refers to an object managed through another command protocol.

Write Object Relative Command

The Write Object Relative command (see Table 2) requests that the target write data transferred from the initiator to the medium.

Table 2- WRITE OBJECT RELATIVE Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (xxh)							
1	Reserved				FUA	Reserved		
2	(MSB)							
:	Logical Block Address							
9								(LSB)
10	(MSB)							
:	Transfer Length							
13								(LSB)
14	(MSB)							
:	Object ID							
17								(LSB)
18	Control = 00h							

A FORCE UNIT ACCESS (FUA) bit of zero indicates that the target may satisfy the command by accessing the cache memory if the WCD bit in RBC mode page 06h is set to zero. Logical blocks may be transferred directly to the cache memory and 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 is a deferred error. A FUA bit of one indicates that the target shall access the media in performing the command prior to returning GOOD status. The 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 target 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 object 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, and this condition shall not be considered an error.

The OBJECT ID field specifies the object containing the logical block(s) requested for transfer. For RBC types of objects the OBJECT ID refers to an object created through the CREATE RBC OBJECT command. For all other types of objects the OBJECT ID refers to an object managed through another command protocol.

Read RBC Object List

The Read RBC Object List command (see Table 3) requests that the target transfer to the initiator a pointer into target memory, where the initiator can read a list of existing RBC objects' descriptions.

Table 3- READ RBC OBJECT LIST Command Descriptor Block

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

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

Table 4- READ RBC OBJECT LIST Data

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
:	Address of RBC Object List							
7								(LSB)
8	(MSB)							
:	Length of RBC Object List							
11								(LSB)

The ADDRESS OF RBC OBJECT LIST field specifies an address within the target memory. The LENGTH OF RBC OBJECT LIST field specifies the number of bytes available at the address in target memory. A list length of zero indicates no RBC objects exist on the medium, and all other list lengths shall be a multiple of 12.

The list format in target memory is shown in Table 5.

Table 5- OBJECT CAPACITY LIST Data

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
:				Object ID				
3								(LSB)
4	(MSB)							
:				Size in Logical Blocks				
11								(LSB)
12	(MSB)							
:				Object ID				
15								(LSB)
16	(MSB)							
:				Size in Logical Blocks				
23								(LSB)
:				:				
:				:				
:				:				

The OBJECT ID field indicates the reference value for the object, used in all other object-relative RBC commands. The SIZE IN LOGICAL BLOCKS field indicates the capacity of the object referred to by the immediately preceding object ID.

Create RBC Object Command

The Create RBC Object command (see Table 6) requests that the target create a new RBC-format object of a specified or unspecified number of logical blocks.

Table 6- CREATE RBC OBJECT Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0				Operation Code (xxh)				
1	(MSB)							
:				Object ID				
4								(LSB)
5	(MSB)							
:				Size in Logical Blocks				
12								(LSB)
13				Control = 00h				

If there is no available space on the medium the target shall return status of CHECK CONDITON, sense of ILLEGAL REQUEST, and additional sense of INSUFFICEINT RESOURCES.

The OBJECT ID field specifies the desired reference value for the new object. If this value refers to an existing object the target shall return status of CHECK CONDITION, sense of ILLEGAL REQUEST, and additional sense of INVALID FIELD IN CDB. If the value is 0xFFFFFFFF, the target shall assign a previously unused ID to the new object.

The SIZE IN LOGICAL BLOCKS field specifies the capacity of the new object. If the value is 0xFFFFFFFFFFFFFFFF, the target shall create the largest possible object and return GOOD STATUS. For any other value, if the available free space is less than the requested object size the target shall return status of CHECK CONDITON, sense of ILLEGAL REQUEST, and additional sense of INSUFFICEINT RESOURCES.

CREATE RBC OBJECT data (see Table 7) shall be returned to the initiator prior to sending GOOD status for the command.

Table 7- CREATE RBC OBJECT Data

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
:				Object ID				
3								(LSB)
4	(MSB)							
:				Size in Logical Blocks				
11								(LSB)

The OBJECT ID field indicates the reference value for the new object, used in all other object-relative RBC commands. The SIZE IN LOGICAL BLOCKS field indicates the capacity of the new object.

Change RBC Object Size Command

The Change RBC Object Size command (see Table 8) requests that the target change the size of the specified RBC-format object.

Table 8- CHANGE RBC OBJECT SIZE Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (xxh)							
1	(MSB)							
:	Object ID							
4								(LSB)
5								
:	Size in Logical Blocks							
12								
13	Control = 00h							

The OBJECT ID field specifies the object to be modified. If this value does not refer to an existing RBC object the target shall return status of CHECK CONDITION, sense of ILLEGAL REQUEST, and additional sense of INVALID FIELD IN CDB.

The SIZE IN LOGICAL BLOCKS field specifies the new capacity of the object. If the value is 0x0000000000000000, the target shall remove the ID from the list of RBC objects and de-allocate the associated logical blocks. If the value is less than the existing object size the target shall de-allocate the logical blocks beyond the new size. If the new size is larger than the existing size and the requested additional number of blocks are greater than the available free space, the target shall return status of CHECK CONDITON, sense of ILLEGAL REQUEST, and additional sense of INSUFFICEINT RESOURCES.

Annex X

(informative)

Object Relative Command Usage

Object relative commands provide access to logical blocks for two primary purposes: LBA-based access to foreign objects managed by other command protocols such as AV/C, and access of native RBC objects.

An object ID for a foreign object would be a handle returned through an action of another command protocol. The object-relative access commands (Read and Write Object Relative) would allow convenient access to blocks within the object but leave object management to the other protocol. The RBC object management commands are not used for this application: the application or computer system residing above the RBC layer would have to be fluent in the other protocol’s object management functions. Any ramifications of block modification through the Write Object Relative command must be dealt with through the other protocol’s functions as well.

In contrast, RBC objects can be wholly managed through the RBC commands Read RBC Object List, Create RBC Object, and Change RBC Object Size. Any object attributes required beyond existence and size are application dependent and outside the scope of this standard. The initiator can maintain object meta-data external to the target, or it can store meta-data within object(s).

In all cases of object relative access the target is assumed to contain some type of object management that maps each object's logical blocks to the physical medium. The RBC commands have no knowledge of this mapping, allowing it to be implementation specific. For each object the target presents a linear extent of LBA's for RBC access.

One benefit of having the target maintain object management for RBC access is that it already must maintain such a system for other protocols such as AV/C. Common object management enables multiple protocols to access the same data in different ways.

For a computer system to take advantage of object relative commands, some part of its file system/driver stack would need slight modification to add the object-aware capabilities. Where before a single volume was created on the computer for the single RBC partition, computer-specific volumes can now be created from one or more objects dedicated to RBC data. One approach would be to have an object represent a single computer file. Other implementations are possible as well.