X3T10/94-111r6

XOR Commands on SCSI Disk Drives

Gerry Houlder
Seagate Technology
8001 E. Bloomington Freeway
Bloomington, MN 55420-1094
TEL: (612) 844-5869
FAX: (612) 844-5708
Gerry_Houlder@notes.seagate.com

Jay Elrod
Seagate Technology
8001 E. Bloomington Freeway
Bloomington, MN 55420-1094
TEL: (612) 844-5978
FAX: (612) 844-5708
Jay_Elrod@notes.seagate.com

Mike Miller
Seagate Technology
8001 E. Bloomington Freeway
Bloomington, MN 55420-1094
TEL: (612) 844-5924
FAX:
Mike_Miller@notes.seagate.com

1. Model for XOR commands

In storage arrays, a host controller organizes a group of storage devices into a redundancy group. Some areas within the address space of the storage array are used for check data. The check data is generated by performing a cumulative exclusive-or (xor) operation with the data from other areas within the address space of the storage array. This xor operation can be performed by the controller, but in many cases system efficiency can be increased by having the storage device perform the xor operation.

An advantage of doing the xor operation in the storage device is a reduced number of data transfers across the system bus and reduced demand for controller resources. For example, when the xor operation is done within the controller 4 data transfer operations are needed for a typical update write sequence: a read transfer from the data device, a write transfer to the data device, a read transfer from the check data device, and a write transfer to the check data device. The controller also does 2 internal xor operations in this sequence; these operations don't consume any system bus bandwidth but do use internal controller resources. Compare this with controller supervised xor operations (see clause x.x) where only 3 data transfer operations are needed: a write transfer to the data device, a read transfer from the data device, and a write transfer to the check data device. Note that the controller doesn't do any internal xor operations, so demand for internal memory bandwidth is reduced. Also compare this with third party xor operation (see subclause x.x) where only 2 data transfer operations are needed: a write transfer from controller to data device and a write transfer from data device to check data device. Note that the controller only issues one command and does no xor operations, resulting in even less demand on internal controller resources.

A further goal of doing the xor operation in the device is to eliminate the need for the array controller to perform any xor operations. This would eliminate the need for an xor engine in the controller, so that an ordinary host adaptor can perform as an array controller and deliver good performance. An array controller performs 3 basic operations that require xor operations. These are the update write, regenerate, and rebuild functions. A command sequence for each of these operations is defined for the following operating modes. The command sequences use the device to perform the xor operations needed for the major functions.

1.1 Host Supervised XOR Operations

Three XOR commands are needed to implement host supervised xor operations: XDWRITE(10), XPWRITE, and XDREAD. The controller will also use READ and WRITE commands for certain operations. This technique may be used when all of the devices are on the same physical bus or when all devices are on separate physical buses that can be accessed by the same controller.

1.1.1 The Update Write function

The update write function is used to write new data to a data device and update the parity information on the check data device.

1) An XDWRITE(10) command is sent to the data device. This transfers the new write data to the data device. The device reads the current data, performs an xor operation on the current data and the new data, retains the xor result in its buffer, and writes the new data to the medium.
2) An XDREAD command is sent to the data device. This transfers the retained xor data from the data device to the controller.
3) An XPWRITE command is sent to the check data device. This transfers the xor data (received in the previous XDREAD command) to the check data device. The device reads the current data, performs an xor operation on the current data and the new data, and writes the xor result to the medium.

1.1.2 The Regenerate function

The regenerate function is used to recreate a data block that cannot be read from a data device. This is done by reading the associated data block from each of the other devices within the redundancy group and performing an xor operation with each of these data blocks. The last xor result is the data that should have been present on the unreadable device. The number of steps is dependent on the number of devices in the redundancy group, but the sequence is as follows:

1) A READ command is sent to the first device. This transfers the current data from the device to the controller.
2) An XDWRITE(10) command with disable write bit set is sent to the next device. This transfers the data from the previous read operation to the device. The device reads its current data, performs an xor operation on the current data and the new data, and retains the xor result in its buffer.
3) An XDREAD command is sent to the same device as in step 2. This transfers the retained xor data from the device to the controller.
4) Steps 2 and 3 are repeated until all devices (except the failed device) in the redundancy group have been accessed. The xor data returned by the last XDREAD command is the regenerated data for the failed device.

1.1.3 The Rebuild function

The rebuild function is similar to the regenerate function, except that the last xor result is written to the formerly unreadable device. This is used when a failed device is replaced and the controller is rewriting the proper data to that replacement device. This sequence is as follows:

1) A READ command is sent to the first device. This transfers the current data from the device to the controller.
2) An XDWRITE(10) command with disable write bit set is sent to the next device. This transfers the data from the previous read operation to the device. The device reads its current data, performs an xor operation on the current data and the new data, and retains the xor result in its buffer.
3) An XDREAD command is sent to the same device as in step 2. This transfers the retained xor data from the device to the controller.
4) Steps 2 and 3 are repeated until all devices (except the replacement device) in the redundancy group have been accessed. The xor data returned by the last XDREAD command is the regenerated data for the replacement device.
5) A WRITE command is sent to the replacement device. This transfers the final xor result data from step 4 to the replacement device. The device writes this data to the medium.

1.2 Third Party XOR Operations

Five XOR commands are needed to implement the third party xor operations: XDWRITE(16), XPWRITE, XDREAD, REGENERATE, and REBUILD. The controller will also use READ and WRITE for certain operations. Third party xor operations are restricted to systems where all devices involved in the commands are located on the same physical bus or loop, or are capable of peer to peer communication via another path (e.g. where the devices have multiple ports).

1.2.1 The Update Write function

The update write function is used to write new data to a data device and update the parity information on the check data device.

1) An XDWRITE(16) command is sent to the data device. This transfers the new write data to the data device. The device reads its current data, performs an xor operation on the current data and the new data, temporarily stores the xor result in its buffer, and writes the new data to the medium.
2) The data device becomes a temporary initiator and sends an XPWRITE command to the check data device. This transfers the resulting xor data from the data device to the check data device.
3) After the data device receives status for its XPWRITE command, it returns ending status for the XDWRITE(16) command to the controller. This indicates that the operations on both the data device and the check data device have completed.

1.2.2 The Regenerate function

The regenerate function is used to recreate a data block that cannot be read from a data device. This is done by reading the associated data block from each of the other devices within the redundancy group and performing an xor operation with each of these data blocks. The last xor result is the data that should have been present on the unreadable device. The number of steps is dependent on the number of devices in the redundancy group, but the sequence is as follows:

1) A REGENERATE command is sent to a valid device in the redundancy group (a valid device is any device other than the failed device). This transfers the regenerate parameter list from the controller to the device.
2) The device reads the requested data from its own medium. The device retains this data for a future xor operation.
3) The device becomes a temporary initiator and sends a READ command to another device included in the regenerate parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the read data from the previous step and the read data received in this step. The resulting xor data is retained for the next step.
4) Step 3 is repeated until all devices listed in the regenerate parameter list have been accessed. The last xor data result is retained in the temporary initiator's buffer. The temporary initiator returns completion status for the REGENERATE command to the controller when this data is available.
5) An XDREAD command is sent from the controller to the temporary initiator device. This transfers the regenerated data from the device to the controller.

1.2.3 The Rebuild function

The rebuild function is similar to the regenerate function, except that the last xor result is written to the replacement device. This is used when a failed device is replaced and the controller is rewriting the data to that replacement device. This sequence is as follows:

1) A REBUILD command is sent to the replacement device in the redundancy group (the device that replaces a failed device). This transfers the rebuild parameter list from the controller to the device.
2) The device becomes a temporary initiator and sends a READ command to a device included in the rebuild parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator retains this data for a future xor operation.
3) The temporary initiator sends a READ command to another device included in the rebuild parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the read data from the previous step and the read data received in this step. The resulting xor data is retained for the next step.
4) Step 3 is repeated until all devices listed in the rebuild parameter list have been accessed. The last xor data result is written to the replacement device's medium, then the device returns completion status for the REBUILD command to the controller.

1.3 Hybrid Subsystem XOR Operations

A hybrid subsystem is one that has 2 or more physical buses and each physical bus has 2 or more devices that are part of the same redundancy group. Such a system could do its xor operations as described in clause x.x (Host Supervised xor operations) but it may choose to use third party xor commands for parts of the xor operation where all the involved devices are on the same physical bus. This clause describes use of the third party xor operations on a hybrid system with 2 physical buses. The redundancy group has 6 devices, with 3 devices on each physical bus.

1.3.1 The Update Write function

When the update write function involves 2 devices that are on different buses, the controller must use the technique described in host supervised xor operations. When the update write function involves 2 devices that are on the same physical bus the controller may use either the host supervised or the third party technique.

1.3.2 The Regenerate function

The regenerate function will always involve 5 devices (all but the failed device) where 3 devices are on one bus (referred to as bus A) and 2 devices on the other bus (referred to as bus B). The sequence is as follows:

1) A REGENERATE command is sent to a device on bus A. This transfers the regenerate parameter list (containing the 2 valid devices and data extents) from the controller to the device.
2) The device reads the requested data from its own medium. The device retains this data for a future xor operation.
3) The device becomes a temporary initiator and sends a READ command to another device included in the regenerate parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the read data from the previous step and the read data received in this step. The resulting xor data is retained for the next step.
4) Step 3 is repeated until all devices listed in the regenerate parameter list have been accessed. The last xor data result is retained in the temporary initiator's buffer. The temporary initiator returns completion status for the REGENERATE command to the controller when this data is available.
5) An XDREAD command is sent from the controller to the temporary initiator device. This transfers the partially regenerated data from the device to the controller.
6) A REGENERATE command with the intermediate data bit set is sent to a device on bus B. This transfers the regenerate parameter list (containing the other valid device and data extent) from the controller to the device. The xor data received in step 5 is sent as the intermediate data.
7) The device reads the requested data from its own medium. The device performs an xor operation between the intermediate data and its own data. The resulting xor data is retained for the next step.
8) The device becomes a temporary initiator and sends a READ command to the other device included in the regenerate parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the xor data from step 7 and the read data received in this step. The last xor data result is retained in the temporary initiator's buffer. The temporary initiator returns completion status for the REGENERATE command to the controller when this data is available.
9) An XDREAD command is sent from the controller to the temporary initiator device. This transfers the regenerated data from the device to the controller.

1.3.3 The Rebuild function

The rebuild function will always involve 5 valid devices and the replacement device where 3 valid devices are on one bus (referred to as bus A) and 2 valid devices and the replacement device are on the other bus (referred to as bus B). The sequence is as follows:

1) A REGENERATE command is sent to a device on bus A. This transfers the regenerate parameter list (containing the 2 valid devices and data extents) from the controller to the device.
2) The device reads the requested data from its own medium. The device retains this data for a future xor operation.
3) The device becomes a temporary initiator and sends a READ command to another device included in the regenerate parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the read data from the previous step and the read data received in this step. The resulting xor data is retained for the next step.
4) Step 3 is repeated until all devices listed in the regenerate parameter list have been accessed. The last xor data result is retained in the temporary initiator's buffer. The temporary initiator returns completion status for the REGENERATE command to the controller when this data is available.
5) An XDREAD command is sent from the controller to the temporary initiator device. This transfers the partially regenerated data from the device to the controller.
6) A REBUILD command with the intermediate data bit set is sent to the replacement device on bus B. This transfers the rebuild parameter list (containing the 2 valid devices and data extents) from the controller to the device. The xor data received in step 5 is sent as the intermediate data.
7) The device becomes a temporary initiator and sends a READ command to a device included in the rebuild parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the intermediate data and the read data received in this step. The resulting xor data is retained for the next step.
8) The temporary initiator sends a READ command to the other device included in the rebuild parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an xor operation between the read data from the previous step and the read data received in this step. The last xor data result is written to the replacement device's medium, then the device returns completion status for the REBUILD command to the controller.

1.4 Additional Array Subsystem Considerations

This clause lists considerations that apply to any array subsystem, but describes how use of the XOR commands may affect handling of those situations.

1.4.1 Buffer Full status handling

When the controller sends an XDWRITE(10) or REGENERATE command to a device, the device has an obligation to retain the resulting xor data until the controller issues a matching XDREAD command to retrieve the data. This locks up part or all (depending on the size of the device's buffer and the size of the xor data block) of the device's buffer space. When all of the device's buffer is allocated for xor data, it may not be able to accept new media access commands other than valid XDREAD commands and it may not even be able to begin execution of commands that are already queued by the device.

When the device cannot accept a new command because there is not enough space in the buffer, the device shall terminate the command with Buffer Full status. When a controller receives this status, it should issue any matching XDREAD commands needed to satisfy any previous XDWRITE(10) or REGENRATE commands. This will result in buffer space being freed for other commands. If it is a multi-initiator system and the controller has no XDREAD commands to send, the controller should assume the buffer space has been allocated to another initiator. The controller should retry the command in the same manner that a command ending with BUSY status would be retried.

The controller can use command linking to avoid a Buffer Full condition. For example, a host supervised update write function would consist of an XDWRITE(10) command linked to an XDREAD command. This prevents the device from working on any other commands until after the xor data has been transferred to the controller.

1.4.2 Access to an inconsistent stripe

When the controller begins an update write to a device, the data in that device has been updated when the status is returned for the command. Until the check data device has been updated, however, the associated stripe in the redundancy group is not consistent. The controller must keep track of this window of inconsistency and make sure that a regenerate or rebuild function for any data extent within the stripe is not attempted until after the check data device has been updated (making the stripe consistent again). For multi-controller systems, this problem is more severe because each controller must make sure that the other controller is not writing to a stripe that it is regenerating or rebuilding. This coordination between controllers is system specific and is not addressed by this standard. The following list identifies cases where a controller must be careful to prevent data corruption due to a temporarily inconsistent stripe.

a) When an XDWRITE(10) command has been issued and completed, the data device has been updated but the check device has not. The stripe will be inconsistent until the XPWRITE command to the check device returns completion status.
b) When an XDWRITE(16) command has been issued, the data device and the check data device will be updated at different times during the course of the command. The stripe should be treated as inconsistent between the time the controller issues the XDWRITE(16) command and the completion status for the command is received.
c) Any time a regenerate or rebuild function is in progress, the stripe is known to be inconsistent until the function is completed. Update writes should not be performed on the same stripe whenever a regenerate or rebuild function is in process.

1.4.3 Error handling considerations

If any of the xor commands end with CHECK CONDITION status and an unrecovered error is indicated, an inconsistent stripe will result. The controller must identify the failing device and extent of the failure, then limit access to the inconsistent stripe accordingly. In the case of third party xor operations the failing device may be the data device or the check data device. The controller should be able to identify the failing device from the resulting sense data, but it may have to access the devices directly to determine the health of the affected devices. The recovery procedures that the controller must implement are not addressed by this standard. The use of xor commands in the storage array do not make this process any more difficult than if the array controller did its own xor operations and only sent READ and WRITE commands to the data devices.

New XOR Operation SCSI Commands and Mode Pages

2. XDWRITE(16) command

The XDWRITE(16) command (see table 1) requests that the target xor the data transferred with the data on the medium. The disposition of the data transferred from the initiator is controlled by the disable write bit. The resulting xor data is sent to a secondary device using an XPWRITE command.

Table 1 - XDWRITE(16) command

Byte/Bit	7	6	5	4	3	2	1	0
0	Operation Code (80h)							
1	Table address	Reserved	Reserved	DPO	FUA	Disable write	Port control	
2	MSB							
3	Logical Block Address							
4								
5								LSB
6	MSB							
7	Secondary logical block address							
8								
9								LSB
10	MSB							
11	Transfer length							
12								
13								LSB
14	Secondary address							
15	Control							

A table address bit of zero indicates that the secondary address field contains the SCSI ID or destination ID of the target to which the xor data is transferred.

> Note: This assumes that all the devices are on the same physical bus or loop. If the bus protocol requires more than one byte of address the secondary address is used as the least significant byte of the address. The upper bytes are set equal to the upper bytes of the target.

A table address bit of one indicates that the secondary address field contains a pointer to a look up table of SAM compliant addresses. This look up table is not specified in this standard.

A disable write bit of zero indicates that the data transferred from the initiator shall be written to the medium after the xor operation is complete. A disable write bit of one indicates that the data shall not be written to the medium. If the disable write bit is set to one the FUA bit is ignored.

The port control field is defined in table 7. If the port control field has a value of 01b and the target is not a multiple port device the command shall be terminated with a CHECK CONDITION status. The sense data shall be set to ILLEGAL REQUEST : INVALID FIELD IN CDB.

The xor data transfer to the secondary target is performed using an XPWRITE command. The XPWRITE command is sent to the device specified in the secondary address field. The secondary logical block address field value is placed in the starting logical block address field of the XPWRITE command. The transfer length field value is placed in the transfer length field of the XPWRITE command. The completion status of the XPWRITE command shall be received before the completion status of the XDWRITE is returned to the initiator.

The XDWRITE command has an implied exclusive access extent reservation for the specified logical blocks that remains in effect until completion status is returned.

If the XPWRITE command terminates with a CHECK CONDITION status and the sense key is not set to RECOVERED ERROR the XDWRITE command shall return a CHECK CONDITION status.

2.1 Errors During The XDWRITE Command

Two classes of exception conditions may occur during execution of an XDWRITE command. Either or both of these classes of exception may occur during command execution:

2.1.1 Errors within the managing SCSI device (primary errors)

The first class consists of those exception conditions which are detected by the device that received the XDWRITE command and are not due to the failure of a secondary command. These conditions include invalid parameters in the XDWRITE command, inability of the device to continue operating, and parity errors while transferring the XDWRITE command, data, or status byte. In the event of such an exception condition, the target of the XDWRITE command shall:

1) Terminate the XDWRITE command with CHECK CONDITION status.
2) Build sense data according to the exception condition.

2.1.2 Errors from the secondary command (secondary errors)

The second class of errors consists of exception conditions resulting from a failure of a secondary command (i.e., XPWRITE). The sense data for such errors shall be passed to the XDWRITE initiator in the additional sense bytes area of the sense data.

If the managing SCSI device detects the exception it shall:

1) Terminate the XDWRITE command with CHECK CONDITION status.
2) Set the sense key to ABORTED COMMAND if there are no primary errors to report. Otherwise, the sense key shall be set according to the primary error.
3) Set the first byte of the command specific information field of the sense data to the starting byte number, relative to the first byte of sense data, of an area that contains the managing SCSI device's sense data for the secondary error. A zero value in this byte indicates no secondary error has been detected by the managing SCSI device. The secondary sense data shall be built in the normal sense data format as defined for the REQUEST SENSE command.

If the secondary target detects the exception, the managing SCSI device receives CHECK CONDITION status from the secondary target. The managing SCSI device shall recover the sense data associated with the exception condition and shall:

1) terminate the XDWRITE command with CHECK CONDITION status;
2) set the sense key to ABORTED COMMAND if there are no primary errors to report. Otherwise, the sense key shall be set according to the primary error;
3) set the second byte of the command specific information field of the sense data to the starting byte number, relative to the first byte of sense data, of an area that contains (unchanged) the secondary target's status byte followed by its sense data. A zero value in this byte indicates no secondary error has been detected by the secondary target.

2.2 XDWRITE(10) command

The XDWRITE(10) command (see table 3) requests that the target xor the data transferred with the data on the medium. The resulting xor data is stored in the target's buffer. The disposition of the data transferred from the initiator is controlled by the disable write bit.

Table 3 - XDWRITE(10) command

Byte/Bit	7	6	5	4	3	2	1	0
0	Operation code (50h)							
1	Reserved	Reserved	Reserved	DPO	FUA	Disable write	Reserved	
2	MSB							
3	Logical block address							
4								
5								LSB
6	Reserved							
7	Transfer length							
8								
9	Control							

The resulting xor data is retained in the target's buffer until it is retrieved by an XDREAD command with starting logical block address and transfer length fields that match the starting logical block address and transfer length of this command.

The field descriptions are the same as in the XDWRITE(16) command.

2.3 XPWRITE command

The XPWRITE command (see table 4) requests that the target xor the data transferred with the data on the medium and then writes the xor data to the medium.

Table 4 - XPWRITE command

Byte/Bit	7	6	5	4	3	2	1	0
0	Operation code (51h)							
1	Reserved			DPO	FUA	Reserved		
2	MSB							
3	Logical block address							
4								
5								LSB
6	Reserved							
7	Transfer length							
8								
9	Control							

The logical block address field specifies the starting logical block address for the target to read data from its medium. It also specifies the starting logical block address to write the xor result data to its medium.

The transfer length field specifies the number of blocks to be read from the medium. It also specifies the number of blocks that are written to the medium.

2.4 XDREAD command

The XDREAD command (see table 5) requests that the target transfer to the initiator the xor data generated by an XDWRITE(10) or REGENERATE command.

Table 5 - XDREAD command

Byte/Bit	7	6	5	4	3	2	1	0
0	Operation code (52h)							
1	Reserved							
2	MSB							
3	Logical block address							
4								
5								LSB
6	Reserved							
7	Transfer length							
8								
9	Control							

The xor data transferred is identified by logical block address and transfer length that are the same as those specified in a prior XDWRITE(10) or REGENERATE command. If a match is not found the command is terminated with a CHECK CONDITION status. The sense data is set to ILLEGAL REQUEST : INVALID FIELD IN CDB.

2.5 REBUILD Command

The REBUILD command (see table 6) requests that the target write to the medium the xor data generated from the specified source devices. The target, acting as a temporary initiator, issues READ commands to retrieve the specified data.

Table 6 - REBUILD COMMAND

Byte/Bit	7	6	5	4	3	2	1	0
0	Operation code (81h)							
1	Reserved			DPO	FUA	IntData	Port control	
2	MSB							
3	Logical block address							
4								
5								LSB
6	MSB							
7	Rebuild length							
8								
9								LSB
10	MSB							
11	Parameter list length							
12								
13								LSB
14	Reserved							
15	Control							

Note: The target that receives the REBUILD command is not one of the source devices. If only one source is specified, then an xor operation does not occur. This case can occur in disk mirroring applications.

The rebuild operation shall be performed in ascending logical block order, beginning with the specified logical block address. If the command terminates with a CHECK CONDITION status the sense data contains the logical block address of the last successfully rebuilt block plus one (i.e., the address of the block in error).

IMPLEMENTOR'S NOTE: Since blocks are rebuilt in ascending order, all blocks with a logical block address less than that returned in the sense data should be rebuilt and written to the medium.

If the intermediate data (IntData) bit is set to zero, then intermediate data is not sent with the rebuild parameter list. If the bit is set to one, the rebuild parameter list includes intermediate data. The length of the intermediate data can be calculated by multiplying the rebuild length times the block size. This data shall be treated as an additional source, and an xor operation performed with it and the data from the specified sources.

The logical block address field specifies the starting logical block address for the target to write the xor result data to its own medium.

The rebuild length field specifies the number of blocks to be written to the medium. It also specifies the number of blocks that are read from each source.

The port control field is defined in table 7. If the port control field has a value of 01b and the target is not a multiple port device the command shall be terminated with a CHECK CONDITION status. The sense data shall be set to ILLEGAL REQUEST : INVALID FIELD IN CDB.

Table 7 - Port control field.

Value	Description
00	The target transfers the data using the same port that received the command.
01	The target transfers the data using a different port than that which received the command.
10	The target transfers the data using one port of the targets choice.

11	The target transfers the data using one or more ports of the targets choice.

The REBUILD parameter data is described in table 8.

Table 8 - REBUILD and REGENERATE parameter data

Byte/Bit	7	6	5	4	3	2	1	0
0	Number of source descriptors							
1	Reserved							
2	Reserved							
3	Reserved							
	Source descriptor							
0	MSB							
1								
2								
3	Source physical address							
4								
5								
6								
7								LSB
8	MSB							
9								
10	Source starting logical block address							
11								LSB
0	MSB							
1								
n	Intermediate data							LSB

The number of source descriptors field indicates the number of source descriptors in the rebuild parameter data. Each source descriptor identifies one device that is a data source for the rebuild operation and the source logical block address.

Note: Intermediate data is not considered a source device for purposes of this field. The intermediate data, if any, follows the source descriptors.

The source physical address field specifies a SAM compliant address of a device that is a data source.

The intermediate data field contains data that shall be used in the xor operation with the data from the specified source devices. The length of the data is equal to the rebuild length multiplied by the block size.

The source starting logical block address field indicates the starting logical block address to use when reading data from the source.

2.6 REGENERATE command

The REGENERATE command (see table 9) requests that the target write to the buffer the xor data generated from its own medium and the specified source devices. The target, acting as a temporary initiator, issues READ commands to retrieve the specified data.

Table 9 - REGENERATE command

Byte/Bit	7	6	5	4	3	2	1	0
0	Operation code (82h)							
1	Reserved					IntData	Port control	
2	MSB							
3	Logical block address							
4								
5								LSB
6	MSB							
7	Regenerate length							
8								
9								LSB
10	MSB							
11	Parameter list length							
12								
13								LSB
14	Reserved							
15	Control							

The resulting xor data is retained in the target's buffer until it is retrieved by an XDREAD command with a starting logical block address and transfer length that match the logical block address and regenerate length of this command.

See 2.5 for a definition of the IntData bit.

See table 7 for a definition of the port control field.

The logical block address field specifies the starting logical block address for the target to read data from its own medium. This data is a source for the regenerate operation.

The regenerate length field indicates the length in logical blocks of the resulting xor data. It also specifies the length in logical blocks that is transferred from each of the specified sources.

The parameter data for the REGENERATE command is defined in table 8. This parameter data describes the other devices that will be sources for the regenerate operation. The target receiving the REGENERATE command is implicitly a source.

2.7 XOR control mode page

The xor control mode page (see table 10) provides the initiator with the means to obtain or modify certain xor operating parameters of the target.

Table 10 - XOR control mode page

Byte/Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	Page code (10h)					
1	Page length (14h)							
2	Reserved						XorDis	Log mode
3	Reserved							
4	MSB							
5	Maximum xor write size							
6								
7								LSB
8	MSB							
9	Reserved							
10								
11								LSB
12	MSB							
13	Maximum regenerate size							
14								
15								LSB
16	MSB							
17	Maximum rebuild read size							
18								
19								LSB
20	Rebuild delay							
21								

An xor disable (XorDis) bit of zero enables the xor functions within a device. An XorDis bit of one disables the xor functions within a device.

A log mode bit of zero indicates that the target is not enabled as a logging device. A log mode bit of one specifies that the target is enabled as a logging device.

The maximum xor write size field specifies the maximum transfer length in blocks that the target accepts for the XDWRITE or XPWRITE commands.

The maximum regenerate size field specifies the maximum regenerate length in blocks that the target accepts for the REGENERATE command.

The maximum rebuild read size field specifies the maximum rebuild length in blocks that the target shall use for READ commands during a rebuild operation.

The rebuild delay field specifies the minimum time in milliseconds between successive READ commands during a rebuild operation.