T10/00-377r1

To: T10 Technical Committee
From: Rob Elliott, Compaq Computer Corporation (Robert.Elliott@compaq.com)
Carl Zeitler, Compaq Computer Corporation (Carl.Zeitler@compaq.com)
Date: 17 October 2000
Subject: Bidirectional data transfers in FCP-2

Revision 0: Initial revision.

Revision 1: Applied suggestions received at Joint T10/T11.3 meeting on 4 October 2000.

Section 6.3.7 word 3 bit 10. Reverse meaning so 0 is the common case. This was apparently a misunderstanding – the bit is "bidirectional XFER_RDY disable" not "bidirectional command disable." FCP-2 allows XFER_RDY to be disabled "If the system has mechanisms outside the scope of this standard for controlling the data transfer length," Revision 0 proposed a separate bit so XFER_RDY could be disabled separately for bidirectional and unidirectional commands, in case a device wanted to support disabling them for old unidirectional commands but was not capable of doing so for a bidirectional command like XDWRITEREAD. Since this feature is practically unusable (relying on "mechanisms outside the scope of this standard") revision 1 just overloads bidirectional commands onto the current bit.

Table 24. Fix byte 11 so additional length uses the whole field. Done. This proposal is not changing that byte, so it should just reflect the latest FCP-2 definition.

Section 9.3. Last paragraph does apply to write data too, so fix the wording. Done.

Section A.2. Add comma where parallel bars were removed. Done.

Section 12.1.2. Add statement: "sequence level recovery should not be used for bidirectional commands." Add a note to leave out if exchange level recovery is adopted. Done.

Overview

T10/00-309r1 by George Penokie, accepted into SAM-2 revision 14, modified SAM to allow protocols and commands to support bidirectional data transfers – commands which transfer both read and write data. This proposal modifies FCP-2 (revision 4a) to support such transfers.

Although Fibre Channel is a full duplex interconnect, data transfers within an exchange (command) are half-duplex due to the sequence initiative concept. This means only a write or read for the command can be active at one time, not both. This is an FC-FS issue, not an FCP-2 issue. This proposal does not try to eliminate this restriction and limits transfers to half-duplex.

The target has full control over when read or write data is transferred. It can run them in any order, and switch as many times as it wants. If it wants to switch from a write to a read, it just sends a FCP_DATA IU. If it wants to switch from a read to a write, it sends an FCP_XFER_RDY IU with Sequence Initiative transfer indicated. The initiator follows by sending an FCP_DATA IUs, with Sequence Initiative transfer indicated. No additional IUs are needed.

The Command IU changes to allow both READDATA and WRITEDATA bits to be enabled at once. An additional Data Length (DL) field is added when both READDATA and WRITEDATA are set since the read data and write data sizes may be different. The data structure is already variable length thanks to the ADDITIONAL FCP CDB LENGTH field, so 4 more bytes should be tolerable.

The ability to disable the initial FCP_XFER_RDY via process login creates a problem for a bidirectional command which needs to transfer read data first. If the initiator is allowed to immediately send write data after sending the command, the target needs to buffer it. This buffering is slightly different from that needed for a unidirectional write command. For the XDWRITEREAD command, for example, the temporary XOR buffer may not be large enough to hold all the write data for the command. An additional write buffer is needed that can hold the largest amount of data that can be written (via a "mechanism outside the scope of this standard"). To avoid this issue, devices with such concerns must not support disabling XFER_RDY.

Overrun and underrun of the Data Length (FCP_DL) complicates the RSP IU. The read and write could each result in an error. The existing bits cover the write direction; extra bits are added to indicate these errors in the read direction. An extra field is added for the bidirectional read residual count if either of the bidirectional read error bits is set. This makes the field size vary based on presence of an error. Alternatively, it could be fixed at a larger size for bidirectional commands. Alternatively, this type of error reporting could simply not be supported for bidirectional commands (is this feature used by real devices?)

"BIDIRECTIONAL_READ" was added to the names of existing fields when creating new ones. A shorter acronym may be preferred (maybe just "BIDI").

Section 6.3.7.15 Word 3, Bit 0: WRITE XFER_RDY DISABLED

[after dropping the BIDIRECTIONAL WRITE XFER RDY DISABLED bit in revision 0, the only change left here is to fix a typo]

When this bit is set to 0, FCP_XFER_RDY IUs shall be used for SCSI write operations. When this bit is set to 1, FCP_XFER_RDY IUs may ~~be not~~ *not be* used before the first FCP_DATA IU to be transferred in the write operation. If both the originator and responder choose to disable write FCP_XFER_RDY IUs, then all FCP I/O operations performing SCSI writes between the FCP_Ports shall operate without using the FCP_XFER_RDY IU before the first FCP_DATA IU. The FCP_XFER_RDY IU shall be transmitted to request each additional FCP_DATA IU, if any, after the first one. If either the originator or the responder requires the use of FCP_XFER_RDY IUs during writes, then the exchange responder shall transmit an FCP_XFER_RDY IU requesting each FCP_DATA IU, including the first, from the exchange originator.

Section 9.1 FCP_CMND IU

Add a FCP BIDIRECTIONAL READ DL field to the table:

Table 24 – FCP_CMND payload

[Editor's note: formatting isn't exactly like FCP-2; just add the field at the end rather than replace the table]

0	(MSB) FCP_LUN		
7	(LSB)		
8	COMMAND REFERENCE NUMBER		
9	RESERVED 7:3	TASK ATTRIBUTE 2:0	
10	TASK MANAGEMENT FLAGS		
11	ADDITIONAL FCP CDB LENGTH 7:3	RDDATA	WRDATA
12	(MSB) FCP CDB		
27	(LSB)		
28	(MSB) ADDITIONAL FCP CDB		
	(LSB)		
n+1	(MSB) FCP DL		
	(LSB)		

n+4	
n+5	*(MSB)*
	FCP BIDIRECTIONAL READ DL
n+8	*(LSB)*

Section 9.1.1.4 Task management flags, Byte 10

Task management function shall be transmitted by the initiator (Exchange Originator) using a new Exchange. If any task management flag is set to 1, the FCP_CDB, FCP_DL, *FCP_BIDIRECTIONAL_READ_DL,* TASK ATTRIBUTES, RDDATA, and WRDATA fields and bits are not valid and are ignored. No more than one task management flag shall be set to *1* in any FCP_CMND IU.

Section 9.1.1.6 Read Data, Byte 11

[typo note: 9.1.1.x headers use a mix of small caps and mixed case]

READ DATA, when set to 1, specifies that the initiator expects FCP_DATA IUs for the task ~~to be~~ in the direction opposite to the direction of the FCP_CMND IU. This is a SCSI read ~~type~~ operation*, used for SCSI read commands and SCSI bidirectional commands*.

Section 9.1.1.7 WRITE DATA, BYTE 11

WRITE DATA, when set to 1, specifies that the initiator expects FCP_DATA IUs for the task ~~to be~~ in the same direction as the FCP_CMND IU. This is a SCSI write operation*, used for SCSI write commands and SCSI bidirectional commands*. *[end paragraph]*

If both READ DATA and WRITE DATA are set to 0, there shall be no FCP_DATA IUs and FCP_DL shall be 0. ~~The initiator shall not set both the READ DATA and the WRITE DATA bits to 1.~~ *If both READ DATA and WRITE DATA are set to 1, the command is a bidirectional command and the FCP_BIDIRECTIONAL_READ_DL field shall be included in the FCP_CMND payload.*

Section 9.1.1.10 FCP_DL

For a bidirectional command, the FCP_DL field contains a count of the greatest number of data bytes expected to be transferred from the application client data buffer by the SCSI command. The parameter is the command byte count defined by SAM-2.

For a unidirectional write command, ~~The~~ *the* FCP_DL field contains a count of the greatest number of data bytes expected to be transferred ~~to or~~ from the application client data buffer by the SCSI ~~CDB~~*command*. *For a unidirectional read command, the FCP_DL field contains a count of the greatest number of data bytes expected to be transferred to the application client data buffer by the SCSI command.* The parameter is the command byte count defined by SAM-2. *[break paragraph here]*

An FCP_DL value of 0 indicates that no data transfer is expected regardless of the state of the READ DATA and WRITE DATA bits and that no FCP_XFER_RDY or FCP_DATA IUs shall be transferred.

[new] Section 9.1.1.11 FCP_BIDIRECTIONAL_READ_DL

If RDDATA and WRDATA are both set to 1, a FCP_BIDIRECTIONAL_READ_DL field follows the FCP DL field. The FCP_BIDIRECTIONAL_READ_DL field contains a count of the greatest number of data bytes expected to be transferred to the application client data buffer by the SCSI command. The parameter is the command byte count defined by SAM-2. An FCP_BIDIRECTIONAL_READ_DL value of 0 indicates that no read data transfer is expected regardless of the state of the READ DATA bit and that no FCP_DATA IUs shall be transferred for read data.

If either RDDATA or WRDATA is set to 0, the FCP_BIDIRECTIONAL_READ_DL field shall not be included in the FCP_CMND_IU data payload.

Section 9.3 FCP_DATA IU

…

During any *write* data transfer *(an operation that uses Data Out actions, IUs T6 or T7)*, the initiator shall always have available a buffer of length FCP_DL~~. The buffer contains~~ *containing* data to be transferred to the target ~~if the operation is a write operation (an operation that uses Data Out actions,.IUs T6 or T7).~~

During any read data transfer for a unidirectional read command (an operation that uses the Data In action, IU I3), the initiator shall always have available a buffer of length FCP_DL that~~The buffer~~ receives the data ~~if the operation is a read operation (an operation that uses the Data In action, IU I4~~.

[note: existing text above refers to I4 FCP_RSP when it should refer to I3 FCP_DATA]

During any read data transfer for a bidirectional command (an operation that uses the Data In action, IU I3), the initiator shall always have available a buffer of length FCP_BIDIRECTIONAL_READ_DL that receives the data.

The target shall never request or deliver data outside the buffer length defined by FCP_DL *or FCP_BIDIRECTIONAL_READ_DL*. If the command requested that data beyond FCP_DL be transferred, the FC_RSP IU shall contain the FCP_RESID_UNDER bit. *If the command requested that data beyond FCP_BIDIRECTIONAL_READ_DL be transferred, the FC_RSP IU shall contain the FCP_BIDIRECTIONAL_READ_RESID_UNDER bit.* The command is completed normally except for presentation of the overrun condition. See 9.4.~~23~~. *[this changed going from fcp2r4 to fcp2r4a – it should point to the FCP_RESID_UNDER bit]*

…

If the amount of data ~~returned~~ *transferred* does not match FCP_DL *for a unidirectional command, FCP_DL for the write data transfer of a bidirectional command, or FCP_BIDIRECTIONAL_READ_DL for the read data transfer of a bidirectional command*, the error detection and recovery procedure described in clause 12 may be invoked or the FCP I/O operation may be terminated with a recovery abort or other failure indication. The mechanism a SCSI Initiator uses to determine that the correct amount of data has been returned is outside the scope of this standard. Data that has been retransmitted and overlaid shall be counted only once.

Section 9.4.1 Overview and format of FCP_RSP IU

Add two bits and a field to Table 28 FCP_RSP payload:

Table 24 – FCP_RSP payload

0-9	Reserved							
10	*Rsvd*	*FCP BIDIRECTIONAL READ RESID UNDER*	*FCP BIDIRECTIONAL READ RESID OVER*	FCP CONF REQ	FCP RESID UNDER	FCP RESID OVER	FCP SNS_LEN VALID	FCP RSP_LEN VALID
11	SCSI status code							
12-15	FCP_RESID							
16-19	FCP_SNS_LEN (= n)							
20-23	FCP_RSP_LEN (= m)							
24	FCP_RSP_INFO (m bytes long)							
23+m								
24+m	FCP_SNS_INFO (n bytes long)							
23+m+n								
24+m+n	*FCP_BIDIRECTIONAL_READ_RESID*							
27+m+n								

[new] Section 9.4.x FCP_BIDIRECTIONAL_READ_RESID_UNDER

FCP_BIDIRECTIONAL_READ_RESID_UNDER, when 1, indicates that the FCP_BIDIRECTIONAL_READ_RESID field is present and valid and contains the count of bytes that were expected to be transferred, but were not transferred. The application client should examine the FCP_BIDIRECTIONAL_READ_RESID field in the context of the command to determine whether or not an error condition occurred.

[new] Section 9.4.x FCP_BIDIRECTIONAL_READ_RESID_OVER

FCP_BIDIRECTIONAL_READ_RESID_OVER, when 1, indicates that the FCP_BIDIRECTIONAL_READ_RESID field is present and valid and contains the count of bytes that could not be transferred because the FCP_BIDIRECTIONAL_READ_DL was not sufficient. The application client should examine the FCP_RESID field in the context of the command to determine whether or not an error condition occurred.

[no change] Section 9.4.3 FCP_RESID_UNDER

FCP_RESID_UNDER, when 1, indicates that the FCP_RESID field is valid and contains the count of bytes that were expected to be transferred, but were not transferred. The application client should examine the FCP_RESID field in the context of the command to determine whether or not an error condition occurred.

[no change] Section 9.4.4 FCP_RESID_OVER

FCP_RESID_OVER, when 1, indicates that the FCP_RESID field is valid and contains the count of bytes that could not be transferred because the FCP_DL was not sufficient. The application client should examine the FCP_RESID field in the context of the command to determine whether or not an error condition occurred.

[new] 9.4.x FCP_BIDIRECTIONAL_READ_RESID

If either the FCP_BIDIRECTIONAL_READ_RESID_UNDER bit or the FCP_BIDIRECTIONAL_READ_RESID_OVER bit is 1, the FCP_BIDIRECTIONAL_READ_RESID field shall be included in the FCP_RSP payload and shall contain a count of the number of residual data bytes that were not transferred in the FCP_DATA IUs for this bidirectional SCSI command. Upon successful completion of a FCP I/O operation, the residual value is normally 0 and the FCP_BIDIRECTIONAL_READ_RESID value is not valid. Devices having indeterminate data lengths may have a nonzero residual byte count after completing valid operations. Targets are not required to verify that the data length implied by the contents of the CDB cause an overrun or underrun before beginning execution of an SCSI command.

If the FCP_BIDIRECTIONAL_READ_RESID_UNDER bit is set, a transfer that did not fill the buffer to the expected displacement FCP_BIDIRECTIONAL_READ_DL was performed and the value of FCP_BIDIRECTIONAL_READ_RESID is a number equal to:

FCP_BIDIRECTIONAL_READ_DL - highest offset of any byte transmitted

A condition of FCP_BIDIRECTIONAL_READ_RESID_UNDER may not be an error for some devices and some commands.

If the FCP_BIDIRECTIONAL_READ_RESID_OVER bit is set, the transfer was truncated because the data transfer required by the SCSI command extended beyond the displacement value of FCP_BIDIRECTIONAL_READ_DL. Those bytes that could be transferred without violating the FCP_DL value may be transferred. The FCP_BIDIRECTIONAL_READ_RESID is a number equal to:

(Transfer length required by command) - FCP_BIDIRECTIONAL_READ_DL

If a condition of FCP_BIDIRECTIONAL_READ_RESID_OVER is detected, the termination state of the FCP I/O operation is not certain. Data may or may not have been transferred and the SCSI status byte may or may not provide correct command completion information.

If the FCP_BIDIRECTIONAL_READ_RESID_UNDER and the FCP_BIDIRECTIONAL_READ_RESID_OVER bits are 0, the FCP_BIDIRECTIONAL_READ_RESID field is not meaningful and may have any value.

The FCP_BIDIRECTIONAL_READ_RESID field is only included in the FCP_RSP IU if either FCP_BIDIRECTIONAL_READ_RESID_OVER or FCP_BIDIRECTIONAL_READ_RESID_UNDER is set.

Section 12.1.2 Sequence level error recovery

To recover from errors, FCP-2 compliant devices may optionally perform retransmission procedures that allow the commands to be completed without requiring higher level programs to perform command retries. Such recovery is desirable for SCSI logical units that depend critically on command ordering and maintaining records of internal device state. The SCSI initiator and the SCSI target shall agree to perform retransmission using the SRR ELS by setting the retry bit to 1 in PRLI before performing the retransmission of individual IUs. (See 6.3.7.9). An FCP-2 device that has agreed to perform retransmission shall use and accept the REC and SRR ELSs as defined by this standard to perform the retransmission. *Sequence level recovery should not be used for bidirectional commands. [Editor's Note: leave out previous sentence (possibly the whole section) if exchange level recovery is adopted.]*

Section 12.2.2 FCP Error Detection using protocol errors for all classes of service.

…

d) a *unidirectional* read command completed with the data count smaller than FCP_DL and FCP_RESID_UNDER ~~is~~ set to 0*, or a bidirectional read command completed with the data count smaller than FCP_BIDIRECTIONAL_READ_DL and FCP_BIDIRECTIONAL_READ_RESID_UNDER set to 0*;

e) a *unidirectional* read ~~type~~ command completed with the data count smaller than FCP_DL, FCP_RESID_UNDER ~~is~~ set to 1, and the data count plus FCP_RESID ~~is~~ not equal to FCP_DL; *or a bidirectional command completed with the read data count smaller than FCP_BIDIRECTIONAL_READ_DL, FCP_BIDIRECTIONAL_READ_RESID_UNDER set to 1, and the data count plus FCP_BIDIRECTIONAL_READ_RESID not equal to FCP_BIDIRECTIONAL_READ_DL;* and

Section 12.4.1.5 FCP_RSP IU Recovery

…

An Exchange carrying a command that was terminated by a CHECK CONDITION requesting FCP_CONF prior to transferring data may have the same REC values as an Exchange carrying a command having an FCP_XFER_RDY IU not received by the initiator. For a write *or bidirectional* command with a non-zero FCP_DL, the parameters for the SRR shall indicate that an FCP_XFER_RDY IU is expected from the target. The target is aware of the actual present state of the transfer and response and shall either retry the FCP_XFER_RDY IU or, if the actual data transfer length for the command was zero, retry the FCP_RSP.

Section A.2 Application client SCSI command services

The SCSI command services shall be requested by the application client using a procedure call defined as:

Service response = execute command (
fully qualified exchange identifier + logical unit number,
command descriptor block,
[task attribute],
[data-out buffer],
[command byte count] ~~||~~ ,
[data-in buffer],
status).

Section A.4.1 Overview of data buffer movement services

The SCSI data buffer movement services shall be requested from the device server using a procedure call defined as:

Service response = move data buffer (
fully qualified exchange identifier + logical unit number,
device server buffer,
application client buffer offset,
request byte count ||).

~~Only one type of data buffer movement procedure call shall be used while processing one command, either data-in delivery or data-out delivery.~~ *Either data-in delivery, data-out delivery, both data-in and data-out delivery, or no data delivery may be used while processing one command. If both are used (for a bidirectional command), the device server shall combine the data-in and data-out service responses into one service response.*

[new] Section C.1.x SCSI FCI bidirectional command with write before read

A typical SCSI FCI bidirectional command with a single data IU transferred in each direction is shown in table C.x. The example command accepts write data before returning read data.

Initiator function	IU	Target function
Command request	T1, FCP_CMND ->	
		[Prepare data out transfer buffer]
	<- I1, FCP_XFER_RDY	Data out delivery request
Data out action	T6, FCP_DATA ->	
		[Prepare data in transfer]
	<- I3, FCP_DATA	Data in action
		[Prepare response message]
	<- I4, FCP_RSP	Response
[Indicate command completion]		

[new] Section C.1.x SCSI FCI bidirectional command with read before write

A typical SCSI FCI bidirectional command with a single data IU transferred in each direction is shown in table C.x. The example command returns read data before accepting write data.

Initiator function	IU	Target function
Command request	T1, FCP_CMND ->	
		[Prepare data in transfer]
	<- I3, FCP_DATA	Data in action
		[Prepare data out transfer buffer]
	<- I1, FCP_XFER_RDY	Data out delivery request
Data out action	T6, FCP_DATA ->	
		[Prepare response message]
	<- I4, FCP_RSP	Response
[Indicate command completion]		

[new] Section C.1.x SCSI FCI bidirectional command with write before read and FCP_XFER_RDY disabled

A SCSI FCI bidirectional command with two write data IUs and one read data IU is shown in table C.x. The example command accepts write data before returning read data. The initial FCP_XFER_RDY IU has been disabled during process login.

Initiator function	IU	Target function
Command request	T1, FCP_CMND ->	
Data out action	FCP_DATA ->	First Data out
	<- I1, FCP_XFER_RDY	Second Data out delivery request
Data out action	T6, FCP_DATA ->	
	<- I1, FCP_XFER_RDY	Last Data out delivery request
Data out action	T6, FCP_DATA ->	
		[Prepare data in transfer]
	<- I3, FCP_DATA	Data in action
		[Prepare response message]
	<- I4, FCP_RSP	Response
[Indicate command completion]		

[new] Section C.1.x SCSI FCI bidirectional command with intermixed writes and reads

A SCSI FCI bidirectional command with three data IUs transferred in each direction is shown in table C.x. The example command accepts some write data before returning read data, but intermixes writes and reads thereafter.

Initiator function	IU	Target function
Command request	T1, FCP_CMND ->	
		[Prepare data out buffer]
	<- I1, FCP_XFER_RDY	First Data out delivery request
First Data out action	T6, FCP_DATA ->	
		[Prepare data in transfer]
	<- I3, FCP_DATA	First Data in action
	<- I1, FCP_XFER_RDY	Second Data out delivery request
Second Data out action	T6, FCP_DATA ->	
	<- I1, FCP_XFER_RDY	Last Data out delivery request
Third Data out action	T6, FCP_DATA ->	
	<- I3, FCP_DATA	Second Data in action
	<- I3, FCP_DATA	Last Data in action
		[Prepare response message]
	<- I4, FCP_RSP	Response
[Indicate command completion]		

Section E.1 Introduction

This annex diagrams several error detection and recovery procedures for SCSI devices that are executing a Read command with multiple fixed blocks. The examples that follow are shown in Class 3 with in-order delivery for simplicity.

In the following examples the method of error detection and recovery will depend on the Initiator's capabilities:

If the Initiator can only detect that lost read data has occurred by comparing its internal transferred byte count with the Target reported transfer byte count (FCP_DL - FCP_RESID *for unidirectional read commands or FCP_BIDIRECTIONAL_READ_DL – FCI_BIDIRECTIONAL_READ_RESID for bidirectional commands*) then the Initiator will use the After Status Recovery Method. See E.2.

Section E.2.1 Discovery [no change; this is just an example and need not consider bidir]

During the transfer of the read data a frame was lost or dropped. However this Initiator was not designed to detect a sequence error when it occurred. The Target reported in the FCP_RSP that the transferred byte count (FCP_DL - FCP_RESID) was 36000. The Initiator compared this to its internal byte transfer count and discovered that some data had been lost.