T10/00-314r1

To: T10 Technical Committee
From: Rob Elliott, Compaq Computer Corporation (Robert.Elliott@compaq.com)
Date: 23 August 2000
Subject: Bidirectional data transfers in SPI-4

T10/00-309 by George Penokie proposes SAM modifications to allow protocols and commands to support bidirectional data transfers – commands which transfer both read and write data. This proposal modifies SPI-4 to support such transfers.

History

Revision 0, 11 August 2000: first revision
Revision 1, 23 August 2000: Added type codes in SPI L_Q indicating direction (section 14.3.2). Changed “primary/secondary” terminology to “<nothing>/bidirectional read.”

Overview

The required changes mainly deal with the task data pointers. The initiator and target must maintain separate data pointers for the read and write streams. The MODIFY DATA POINTER message needs to be split into two, as any pointer arithmetic will have only one direction in mind. There are no reserved bits in the message to specify which pointer to use, so a new message was created. The SAVE DATA POINTER message can remain as is.

The SPI L_Q needs to indicate direction so the initiator can assume the correct state (pick the correct data-in or data-out pointer and start the appropriate DMA engine) before data starts flowing.

Changes are presented in order of SPI-4. The key changes are in sections 14.3.2, 15 and 16.3.8.

Global change

Change “SAVE DATA POINTER” to SAVE DATA POINTERS”.

14.1 SPI information unit overview

When a data transfer agreement is in effect that enables information unit transfers there is no option equivalent to the "physical disconnect without sending a SAVE DATA POINTER*S* message". The initiator shall save the data pointers as soon as the last byte of the last iuCRC for a SPI information unit is transferred. The save shall occur even if the initiator detects an error in the SPI data information unit. If a target retries an operation it shall send a MODIFY DATA POINTER~~S~~ *or MODIFY BIDIRECTIONAL READ DATA POINTER* message*,* then request that the SPI data information unit be transferred again.

Section 14.3.2 Table 41 – Type

04h Data
Sent by a target to indicate a SPI data information unit shall follow this SPI L_Q information unit. The DATA LENGTH field shall not be set to zero. *For a bidirectional command, the SPI data information unit shall be sent from the initiator to the target.*

05h Data Stream
Sent by a target to indicate an unspecified number of SPI data stream information unit*s* shall follow this SPI L_Q information unit. The DATA LENGTH field shall not be set to zero. *For a bidirectional command, the SPI data stream information unit(s) shall be sent from the initiator to the target.*

[Add two more types:]
06h Bidirectional Read Data

Sent by a target to indicate a SPI data information unit shall follow this SPI L_Q information unit. The DATA LENGTH field shall not be set to zero. For a bidirectional command, the SPI data information unit shall be sent from the target to the initiator.

07h Bidirectional Read Data Stream
Sent by a target to indicate an unspecified number of SPI data stream information units shall follow this SPI L_Q information unit. The DATA LENGTH field shall not be set to zero. For a bidirectional command, the SPI data stream information unit(s) shall be sent from the target to the initiator.

15 SCSI pointers

The initiator provides for a set of ~~three~~ pointers for each task, called the saved pointers. The set of ~~three~~ pointers consist of one for the command, one *(for unidirectional commands) or two (for bidirectional commands)* for the data *(data-out and data-in)*, and one for the status. When a send command service is received from an application client, the task's ~~three~~ saved pointers are copied into the initiator's set of ~~three~~ active pointers. There is only one set of active pointers in each initiator. The active pointers point to the next command, data*-out, data-in*, or status byte to be transferred between the initiator and the target. The saved and active pointers reside in the initiator.

The saved command pointer always points to the start of the command descriptor block for the task. The saved status pointer always points to the start of the status area for the task. The saved data*-out* pointer points to the start of the data*-out* area until the target sends a SAVE DATA POINTER*S* message for the task or after the initiator successfully ~~receives or~~ transmits a SPI data information unit. *The saved data-in pointer points to the start of the data-in area until the target sends a SAVE DATA POINTERS message for the task or after the initiator successfully receives a SPI data information unit.*

In response to the SAVE DATA POINTER*S* message or successful receipt or transmission of a SPI data information unit, the initiator stores the value*s* of the current data*-out and data-in* pointer*s* into the saved data*-out and data-in* -pointer*s* for that task. The target may restore the active pointers to the saved pointer values for the current task by sending a RESTORE POINTERS message to the initiator. The initiator then copies the set of saved pointers into the set of active pointers. Whenever a target does a physical disconnect from the bus, only the set of saved pointers are retained. The set of active pointers is restored from the set of saved pointers upon a physical reconnection of the task or a successful receipt of a SPI L_Q information unit.

Since the data pointer value*s* may be modified by the target before the task ends, ~~it~~ *they* should not be used to test for actual transfer length because the value may no longer be valid.

Figures 63, 64, 65, Section 14.3.2, and Section 16.3.2

Change "data pointer" to "data pointers".

Section 16.3.4 IGNORE WIDE RESIDUE

The IGNORE WIDE RESIDUE message (see table 52) shall be sent from a target to indicate that the number of valid bytes sent in the last REQ/ACK handshake data of a DATA IN phase is less than the negotiated transfer width. When information unit transfers are disabled the IGNORE WIDE RESIDUE message shall be sent following that DATA IN phase and prior to any other messages.

[A bidirectional command may encounter IGNORE WIDE RESIDUE messages. I don't think any mention is necessary here. It only applies to the read data, so there should be no confusion which part of a bidirectional command is affected.]

Section 16.3.8 MODIFY DATA POINTER

The MODIFY DATA POINTER message (see table 54) is sent from the target to the initiator and requests that the signed ARGUMENT be added (two's complement) to the value of the current data pointer. *In a unidirectional command, the data pointer is whichever of the data-out or data-in pointers is being used by the command. In a bidirectional command, the data pointer is the data-out pointer.* The ~~E~~*e*nable ~~M~~*m*odify ~~D~~*d*ata ~~P~~*p*ointer (EMDP) bit in the disconnect-reconnect mode page (see 18.1.2) indicates whether or not the target is permitted to issue the MODIFY DATA POINTER message.

It is recommended that the target not attempt to move the data pointer outside the range addressed by the command. Initiators may or may not place further restrictions on the acceptable values. Should the target send an ARGUMENT value that is not supported by the initiator, the initiator may reject the value by responding with the MESSAGE REJECT message. In this case, the data pointer is not changed from its value prior to the rejected MODIFY DATA POINTER message.

Section 16.3.8 MODIFY BIDIRECTIONAL READ DATA POINTER

The MODIFY BIDIRECTIONAL DATA POINTER message (see table 54) is sent from the target to the initiator and requests that the signed ARGUMENT be added (two's complement) to the value of the current data-in pointer. The enable modify data pointer (EMDP) bit in the disconnect-reconnect mode page (see 18.1.2) indicates whether or not the target is permitted to issue the MODIFY BIDIRECTIONAL READ DATA POINTER message. The target shall only issue the MODIFY BIDIRECTIONAL READ DATA POINTER message during a bidirectional command.

It is recommended that the target not attempt to move the data-in pointer outside the range addressed by the command. Initiators may or may not place further restrictions on the acceptable values. Should the target send an ARGUMENT value that is not supported by the initiator, the initiator may reject the value by responding with the MESSAGE REJECT message. In this case, the data-in pointer is not changed from its value prior to the rejected MODIFY BIDIRECTIONAL READ DATA POINTER message.

[assign a new extended message code for this message, either 02h or 05h. This appears in tables 49 and 50.]
[duplicate the MODIFY DATA POINTER table here, with the new extended message code and the name renamed to MODIFY BIDIRECTIONAL READ DATA POINTER]

Section 16.3.12 RESTORE POINTERS

The RESTORE POINTERS message is sent from a target to direct the initiator to copy the most recently saved command, data, and status pointers for the task to the corresponding active pointers. The command and status pointers shall be restored to the beginning of the present command and status areas. The data pointer*s* shall be restored to *either* the value*s* at the beginning of the data area*s* in the absence of a SAVE DATA POINTER*S* message or to the value*s* at the point at which the last SAVE DATA POINTER*S* message occurred for that task.

When information unit transfers are enabled there are implied restore pointers. For more information on this see 14.1 and 14.3.3.

Section 16.3.13 SAVE DATA POINTER*S*

The SAVE DATA POINTER*S* message is sent from a target to direct the initiator to copy the current data pointer*s* to the saved data pointer*s* for the current task.

Section 18.1.2 Disconnect-reconnect mode page

The enable modify data pointer (EMDP) bit indicates whether or not the initiator allows the MODIFY DATA POINTER *and MODIFY BIDIRECTIONAL READ DATA POINTER* message*s* to be issued by the target. If the EMDP bit is a zero, the target shall not issue the MODIFY DATA POINTER *and MODIFY BIDIRECTIONAL READ DATA POINTER* message*s*. If the EMDP bit is a

one, the target is allowed to issue MODIFY DATA POINTER *and MODIFY BIDIRECTIONAL READ DATA POINTER* messages.

If the EMDP bit is a one and the initiator responds to a MODIFY DATA POINTER *or MODIFY BIDIRECTIONAL READ DATA POINTER* message with a MESSAGE REJECT, then the target shall return a CHECK CONDITION. The sense key shall be set to ABORTED COMMAND and the sense code shall be set to INVALID MESSAGE ERROR.

Section 19.3.1 Application client SCSI command services overview

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

Service response = execute command (
target SCSI ID+logical unit number[+tag],
command descriptor block,
[task attribute],
[link control function],
[data-out buffer],
[command byte count] ~~||| [data-in buffer]~~,
[data-in buffer],
status,
service response).

Section 19.4.1 Device server SCSI command services overview

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

Service response = move data buffer (
target SCSI ID+initator SCSI ID+logical unit number [+tag],
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.*