adaptec

Adaptec Technical Memorandum

NCITS T10/97-198r0

To: Ultra-3 SCSI Protocol Ad-Hoc Group **Date**: June 17, 1997

From: Tak Asami (tasami@corp.adaptec.com) **File**: 97-198r0.doc

Subject: Status / Message Simplification for SCSI LFP

Background:

Newer generation of SCSI is proposed to enable data transfers of up to 160MB/sec or more across 16-bit bus. But the actual increase in throughput may be more limited due to the fixed protocol overhead still performed on 8-bit bus in asynchronous protocol. By making the data phase transfer speed twice, in effect, the "protocol overhead" is increased to twice. Without some improvement in protocols, Fast-80 and above SCSI bus may end up spending half the time in non-data transfer activities.

Of course, it is not my intention to demolish existing protocol structures as defined in SIP document. But I would like to propose a few simple modifications to cut down on what I perceive as unneeded overhead.

Proposal:

After the initiator and target negotiated to support LF Protocol (discussed in a separate proposal).

1. Upon target device disconnection within a data transfer, the device may send the newly defined "Save Data Pointer and Disconnect" instead of two separate messages "SDP" and "Disconnect".
2. Upon completion of a command, use a newly defined "Skip" bit in the STATUS byte to indicate the last Message In phase for "TASK COMPLETE" is skipped.

Justification

This proposal assumes a few things:

a) Disk Drives are the devices that benefits the most from Ultra-3 speed.
b) When Disk Drives disconnects and reconnects multiple times during a data transfer, it always asks for SDP before disconnecting.
c) The chance of a Disk Drive issuing "Good" status byte after the task completion is several orders of magnitude likely than otherwise.
d) Nowadays we can afford more hardware so that the possible complications due to the "phase skipping" can be processed hidden behind bus activity.
e) Most hardware implementation can accommodate this change without a major redesign.
f) Message and Status handling are typically not as efficient as data transfer engine in today's implementations, such simple change in protocol can save something in the order of microseconds, not nanoseconds.

The last point f) may be worth elaborating more.
Because all of the message exchange is performed in asynchronous information transfer mode, there is no definitive "minimum transfer cycle time" defined in the standard. It is a function of cable length and its characteristics as well as the protocol devices themselves.

For "SDP-Disconnect" sequence, the time it takes under current standard is:

(i) From Msg_In phase set to REQ assertion (NOT assuming data bus direction change):
(Bus Settle Delay) + (System Deskew Delay) + (Cable Skew Delay) = 455 nsec

(ii) From REQ assertion to ACK assertion to REQ negation to ACK negation, to next REQ assertion for the first message (Save Data Pointer)

There is no defined timing for this in SCSI-3, but it can take as much as Bus Settle Delay (400nsec), depending on the device position on the cable/backplane, without considering the state machine latencies. Here, I will use 200nsec (5MB/sec) as a typical number.

(iii) Repeat (ii) again for the second message (Disconnect).

Add another 200nsec.

Therefore, under the current protocol, it typically takes 855 nsec for SDP-Disc sequence.
In the proposed scheme, it will take 655 nsec. A modest saving of 200 nsec.
But in reality, the HBA who received the message may actually respond to SDP to save the Data Pointer before responding for the disconnect message, may take a little extra time, depending on where it is saving the pointer. In some cases, it was observed that it takes 2µsec to execute the transfer of two messages.
But if we already know that it is going to disconnect while storing the data pointer, there will be less decisions to make, less states to go through, thus simplifying the flow therefore the quicker turn-around.
So it has the potential of saving as much as 1µsec.
Considering the "legal minimum" protocol overhead is in the order of 32µsec (with two disconnects), the potential improvement (3%) is not negligible in this case.

Similar saving can be expected from STATUS phase skipping, in addition to saving of Bus Settle Delay for not going into Message In phase at all. In the current standard,

(i) STATUS Phase (NOT assuming data bus direction change)

(Bus Settle Delay) + (System Deskew Delay) + (Cable Skew Delay) + (REQ/ACK cycle time)
= 655 nsec

(ii) Message In Phase

(Bus Settle Delay) + (System Deskew Delay) + (Cable Skew Delay) + (REQ/ACK cycle time)
= 655 nsec

Therefore, by skipping the step (I), one can potentially save 655 nsec. Again, the other hardware overhead behind the bus protocol may become significant; today, the time it takes to send a message byte is as much as 1.2µsec, so the saving can be that much.

Both of these combined, the potential improvement in the protocol overhead is as much as:
Saving = *n**(REQ/ACK cycle time) + { (Bus Settle Delay) + (System Deskew Delay) + (Cable Skew Delay) + (REQ/ACK cycle time) }
where *n* is the number of times the target role agents breaks the data transfer.

For a 8K Byte transfer, assuming a drive will transmit 4K block at a time, i.e., there are two reconnects after CDB transfer, the total protocol overhead is typically about 32µsec. This proposal will cut this overhead by at least 1µsec (3%) or more, or as much as 2.2µsec (7%) depending on the implementation of the protocol devices.

Considering the change requires minimal of hardware and design investment, it is well worth considering.

Document Changes:

In order to incorporate the proposed change, SCSI-3 Interlocked Protocol (SIP) rev 10 needs to be modified. For the revised definition of status byte, SCSI-3 Architecture Mode (SAM) rev 18 also need to be modified.
The proposed changes are shown in below; the deletions are marked with strike-out, and newly inserted texts are in italics.
The title and section numbers below are referencing the phrases within the SIP rev 10 and SAM rev 18 documents.

SCSI Interlocked Protocol Revision 10:

6.1 Valid service sequences

Figure 35 shows all the valid service request sequences allowed by the SCSI-3 Interlocked Protocol Standard for a target role agent. Any attempt to request a sequence not defined in figure 35 shall result in a protocol error. When a target role agent detects a protocol error it shall request a bus free service.

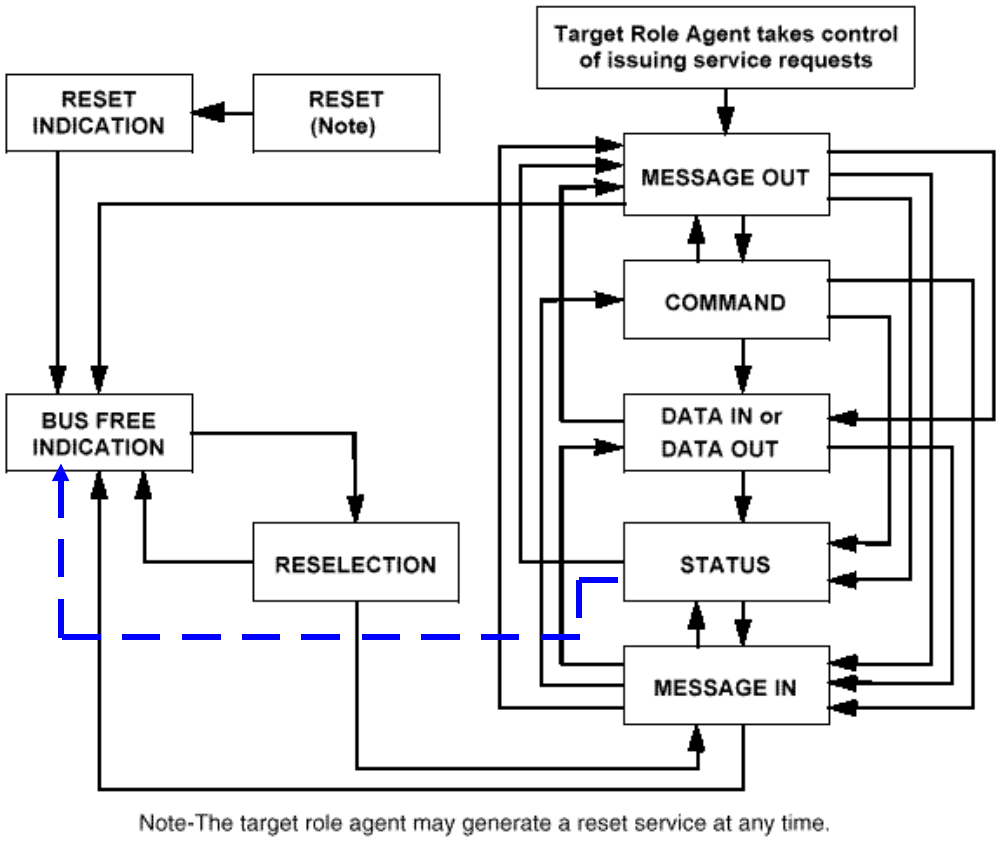

Figure 35 - Valid service request sequences for target role agents

6.3.7 Status service

The status service is a four step confirmed service that provides the means to transfer the status byte from the target role agent to the initiator role agent (see figure 51).

6.3.7.3 Status response

The status response contains the attention flag. The attention flag is set to zero to indicate the initiator role agent is not requesting a message out service. The attention flag is set to one to indicate the initiator role agent is requesting that a message out service be generated by the target role initiator, at its discretion.
If the attention flag is set to zero and status byte has its Disconnect bit set to one, then the next expected request shall be a bus free service.

6.3.7.4 Status confirmation

The status confirmation contains the attention flag. An attention flag set to zero indicates to the target role agent that no message out service is being requested. An attention flag set to one indicates to the target role agent that a message out service is being requested by the initiator role agent.
When the attention flag is set to one the next service request, after the status service, shall be a message out service (see figure 52).
If the attention flag is set to zero and status byte has its Disconnect bit set to one, then the next service request, after the status service shall be a bus free service.

7 SCSI pointers

The SCSI Interlocked Protocol 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 the data, 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 role agent's set of three active pointers. There is only one set of active pointers in each initiator role agent. The active pointers point to the next command, data, or status byte to be transferred between the initiator role agent and the target role agent. The saved and active pointers reside in the initiator role agent.

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 pointer points to the start of the data area until the target role agent sends a SAVE DATA POINTER message for the task.

In response to the SAVE DATA POINTER message, the initiator role agent stores the value of the active data pointer into the saved data pointer for that task. The target role agent may restore the active pointers to the saved pointer values for the current task by sending a RESTORE POINTERS message to the initiator role agent. The initiator role agent then copies the set of saved pointers into the set of active pointers. Whenever a target role agent disconnects 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 reconnection of the task.

Alternatively, if LF protocol is activated, whenever a target role agent disconnects from the bus during the data transfer phases, a single SAVE DATA POINTER & DISONNECT message may be used in place of the "SDP" and "Disconnect" messages.

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

8.2 Link control messages

Table 8 - Link control message codes

Code	Support		Message Name	Direction		Negate ATN before last ACK
	Initiator	Target				
12h	O	O	CONTINUE TASK		Out	Yes
04h	O	O	DISCONNECT	In		n/a
04h	O	O	DISCONNECT		Out	Yes
80h+	M	O	IDENTIFY	In		n/a
80h+	M	M	IDENTIFY		Out	Not required
23h	O	O	IGNORE WIDE RESIDUE	In		n/a
05h	M	M	INITIATOR DETECTED ERROR		Out	Yes
09h	M	M	MESSAGE PARITY ERROR		Out	Yes
07h	M	M	MESSAGE REJECT	In	Out	Yes
***	O	O	MODIFY DATA POINTER	In		n/a
08h	M	M	NO OPERATION		Out	Yes
03h	O	O	RESTORE POINTERS	In		n/a
02h	O	O	SAVE DATA POINTER	In		n/a
TBD#	***O#***	***O#***	***SAVE DATA POINTER & DISCONNECT***	***In***		***n/a***
***	O	O	SYNCHRONOUS DATA	In	Out	Yes

			TRANSFER REQUEST			
13h	O	O	TARGET TRANSFER DISABLE		Out	Yes
00h	M	M	TASK COMPLETE	In		n/a
***	O	O	WIDE DATA TRANSFER REQUEST	In	Out	Yes

Key: M=Mandatory support, O=Optional support
In=Target role agent to initiator role agent, Out=Initiator role agent to target role agent
Yes=Initiator parallel interface agent shall negate ATN before last ACK of message (see SCSI-3 Parallel Interface Standard)
Not required=Initiator parallel interface agent may or may not negate ACK before last ACK of mes-sage (see SCSI-3 Parallel Interface Standard)
n/a=Not applicable
=Extended message ***# = LF protocol message
80h+=Codes 80h through FFh are used for IDENTIFY messages

8.2.2 DISCONNECT

The DISCONNECT message is sent from a target role agent to inform an initiator role agent that the present connection is going to be ended and that a later reconnect will be required in order to complete the task. The message shall not cause the initiator role agent to save the data pointer.
After successfully sending this message, the target role agent shall generate a bus free service. The target role agent shall consider the message transmission to be successful when it receives a message in confirmation with the attention flag cleared.

Target role agents that are requested to break data transfers into multiple connections shall end each successful connection (except possibly the last) with *either* a SAVE DATA POINTER - DISCONNECT message sequence*, or a single "SAVE DATA POINTER AND DISCONNECT" message if the LF Protocol is activated*.

8.2.14 TASK COMPLETE

The TASK COMPLETE message is sent from a target role agent to an initiator role agent to indicate that a task has completed and that valid status has been sent to the initiator role agent**.** After successfully sending this message, the target role agent generates a bus free service. The target role agent shall consider the message transmission to be successful on receipt of a message in confirmation with the attention flag cleared.
The task may have completed successfully or unsuccessfully as indicated in the status.
This message may be abbreviated if STATUS byte with Disconnect bit set is issued from the target role agent.

9.5 Unexpected bus free

An unexpected bus free occurs after a bus free service indication occurs and the initiator role agent does not expect an indication of a bus free service. Initiator role agents only expect an indication of a bus free service to occur after one of the following occurs:
a) reset service,
b) task management function service
c) the following link control messages:
    a) DISCONNECT message when sent from an initiator role agent,
    b) TASK COMPLETE,
d) unsuccessful selection or reselection,
e) Reception of STATUS with Disconnect bit set to one.

Note the Disconnect bit is defined to be the least significant bit (bit 0) of a status byte.

SCSI-3 Architecture Model Revision 18:

5.2 Status

Table 4 -- Status Codes

Status byte codes	Status
0h	GOOD
1h	*GOOD & DISCONNECT*
2h	CHECK CONDITION
4h	CONDITION MET
8h	BUSY
10h	INTERMEDIATE
14h	INTERMEDIATE-CONDITION MET
18h	RESERVATION CONFLICT
22h	COMMAND TERMINATED
28h	TASK SET FULL
30h	ACA ACTIVE
All other codes	Reserved

Definitions for each status byte code are given below.

GOOD & DISCONNECT. This status indicates that the Device Server has successfully completed the task and it is about to enter bus free phase. To be used by SCSI-3 Parallel Interface medium utilizing SCSI-3 Interlocked Protocol only.
In this case, the least significant bit of this byte may be referred to as “Disconnect” bit.