To: SBP-2 working group

From: Pete McLean
Maxtor Corp

Date: 2 May 1997

Subj: Error senarios

I believe that there are a number of errors that can cause problems in SBP-2 that we need to address. In some cases it
may be as simple as warning of possible behavior. In other cases we may have to fix the standard.

Imagine the following:
1) A node sends a packet on the bus.
2)The destination receives the packet just fine and returns the ack.
3)The ack the source gets has a parity error when it arrives .
4)The 1394 spec says that this is reported as AKNOWLEDGE MISSING to a higher layer. In other words, the
originator has no idea if the packet got there or not, the only choice is
to retry.

Now let+s apply this to transactions with SBP-2.

Case 1

1 ) The host writes the MANAGEMENT_AGENT register to login.
2) The device gets the login OK but the ack returned has bad parity.
3) The device goes +busy+ processing the login.
4) A retry by the host will get a resp_conflict_error back.
5) The device sends the login response.
6) The host gets back a login response for a request that the host thought
failed.
7) In the meantime, if the host has sent a retry of the request that arrives after the device has ceased being +busy+ the
request is rejected by the device since the device believes the host is already logged in.

This probably all works out, but I think it would be good to warn host implementors that this behavior is possible.

Case 2

1) The device is in RESET or SUSPENDED state.
2) The host writes to the ORB_POINTER register.
3) The device gets the write OK but the ack returned has bad parity.
4) Having received the write, the device goes to ACTIVE state.
5) The host retries the write.

The spec currently states that when in ACTIVE state, a write to the

ORB_POINTER register may cause unpredictable
target behavior. This is unacceptable.

Case 3

A
1) The device sends a read request to get either an ORB or data to be written on the device.
2) The host gets the request OK but the ack returned has bad parity.
3) The device retires but also gets the data from the first try that the device thought failed.

B
1) The device sends a read request to get either an ORB or data to be written on the device.
2) The host gets the request OK and the ack returned is OK.
3) The host sends the data.
4) The device gets the data OK but the ack returned has bad parity.
5) The host sends the data a second time.

Both 3 Aand B can probably be managed but again I think that this possible
behavior should be pointed out.

Case 4

1) The device completes a command and writes status to the host.
2) The host receives the status write OK but the ack returned has bad parity.
3) The host having gotten the status, retires the ORB, and in fact may immediately create a new ORB in that same
memory location.
4) The device retries the status write.
5) In the best case, the host gets status for a request that has already been retired. In the worst case, the host gets a status
response that retires a command that the device hasn+t even seen yet.

I+m not sure how we handle this one.

I went through the possible register and task transactions a came up with
these. There may be more still hidden in there.

Regards, Pete