Expert IO™

SAS-1.1 Letter Ballot Comments - Rev 1

April 18, 2005

Authors:
Alexander Amezquita (alex@expertio.com) of Expert I/O
Craig Stoops (craig@expertio.com) of Expert I/O
www.expertio.com

Introduction

This document describes issues/suggestions for modification to the current SAS 1.1r09 specification. The following list identifies the subject matter that is described in detail further in this document:

1. Expander – Backoff Reverse Path
2. Expander – Request Path Handling
3. Port Layer – Frame Transmitted Handshake
4. SSP Transport Layer – Balance Counter
5. SSP Transport Layer – Ack Transmitted Confirmation needs Tag Argument
6. Expander – 7.15.4.5 – Transition XL1:Request_Path to XL5:Forward_Open

Expander – Backoff Reverse Path

Problem

Expander Port 0 forwards an open to Expander Port 1, which causes Expander Port 0 to transition to XL3:Open_Confirm_Wait. Expander Port 1 receives an open address frame from the device it is connected. The received open address frame wins according to arbitration rules and thus causes Expander Port 1 to issue a Backoff Reverse Path message destined to Expander Port 0. The specification indicates that upon reception of the backoff reverse path message, Expander Port 0 should transition to XL5:Forward_Open. The specification goes on to say upon entry into XL5:Forward_Open, the expander port should transmit an open address frame based on the arguments of the oaf coincident with the state transition. However, in the case of the backoff reverse path message, there is no mechanism detailed to provide the Expander Port with the open address frame arguments along with the message.

Solution

This problem could be solved by having the expander port that received the backoff reverse path message transition to XL0:Idle rather than XL5:Forward Open. This will enable the expander port to be ready to accept the forward open message that will follow the backoff reverse path message and proceed as described in the specification.

Alternatively, the backoff reverse path message could include the arguments for the open address frame along with the message. However, this option seems more intrusive than the first option

Expander – Request Path Handling

Problem

The specification should clarify what should occur in the following condition: Expander Port 1 issues a request path message which will route to Expander Port 0 resources. Expander Port 1 wins arbitration and proceeds to issue a forward open message that will go to Expander Port 0. Coincidentally (or any time up to receiving the forward open message from Expander Port 1), an open address frame is received by Expander Port 0. This causes Expander Port 0 to transition to XL1:Request_Path. While Expander Port 0 is in XL1:Request_Path the forward open message is received. It is not clear what the expander link should do from this point forward. If the open address frame received by Expander Port 0 is greater, ultimately a backoff retry or backoff reverse path message should be issued to Expander Port 1. If the open address frame received by the forward open message is greater, the request path message to the ECM should be negated. Neither of these methods is explained.

Solution

A possible solution requires modification in both the XL1:Request_Path section of the specification and the request path handling in the ECM. In the XL1:Request_Path section, the contents of the received open address frame could be kept. Upon receiving a forward open message while in XL1:Request_Path, the expander link uses arbitration rules to determine if a backoff retry message or backoff reverse path message should be issued while remaining in the same state. If either of these messages is issued, then no further modifications are necessary since the request path message from Expander Port 0 will control the state operation of Expander Port 0. If the forward open message wins arbitration over the received open address frame by Expander Port 0, the state transition to XL5:Forward_Open would take place as described. The only modification in this case would be that the ECM would ignore the request path message already issued by Expander Port 0.

As a note, the cases are covered if the open address frame is received while Expander Port 0 is in XL5:Forward_Open and XL6:Open_Confirm_Wait. The only case missing in the specification is the case described above.

Port Layer – Frame Transmitted Handshake

Problem

Due to the architectural freedom of having multiple SSP Transports running concurrently on top of a single Port Layer, multiple frames with different tags may be queued to the port layer. The port layer section of the specification does not describe any restriction for issuing multiple transmit frame messages to the link layer as long as the protocol, connection rate, and destination address match. However, the SSP Link Layer state machine is specified such that it can only accept one transmit frame message at a time. This creates an environment where a frame could be implicitly dropped if the transmit frame message is issued by the Port Layer while the SSP Link Layer is not in a state that recognizes the message.

Solution

The description of a handshake should be added to the Port Layer section of the specification. Specifically in the PL_PM3:Connected state should specify that a new transmit frame message can only be issued if there are no outstanding frame transmitted confirmations from the SSP Link Layer.

SSP Transport Layer – Balance Counter

Problem

The specification is very detailed in the description of the ITS state transitions. A section particularly describes how the ITS cannot transition out of PREPARE_DATA_OUT until it has received as many ack received confirmations as data frames it has sent out. This wording implies a counter that is not explained.

Solution

The specification should describe a balance counter (similar to ones described in the Link Layer) that increments on every frame transmitted transmission status confirmation and decrements on every ack received transmission status confirmation

SSP Transport Layer – Ack Transmitted Confirmation Needs Tag Argument

Problem

When an ack transmitted confirmation is received by the SSP Transport layer, it is not known for which frame the ack transmitted confirmation is associated. For instance, in the case of a wide link where a single transport layer is servicing commands for multiple tags simultaneously, the ST layer needs to know which ack transmitted confirmation is associated with which received frame.

Solution

The port layer has access to the information regarding which tag is associated with which confirmation. The specification should detail that the transmission status and the ack transmitted message should include an argument of the tag associated with the confirmation.

Expander – 7.15.4.5 Transition XL1:Request_Path to XL5:Forward_Open

Problem

The specification indicates that if a forward open message is received after an arbitrating (NORMAL) message has been received, the forward open message is ignored. We believe there is a flaw in the statement or perhaps overall in the expander function handling. It is illustrated in the following case.

Expander Port 2 wins arbitration to open Expander Port 0. Expander Port 1 receives an Open Address Frame (OAF) to Open Expander Port 2 but has to hold off as it waits for the connection to try to open between 2 and 0. The expander function sends an arbitrating (NORMAL) message to expander Port 1 to acknowledge the receipt of the request path message. Expander Port 0 transmits an OAF at the same time an OAF resolving to port 1 is received from the device connected to Expander Port 0. The OAF received by expander Port 0 wins over the outgoing OAF by arbitration rules. Expander Port 0 sends a backoff retry message to Expander Port 2 and also a request path message to the expander function requesting a connection to port 1. The request path message from Expander Port 0 wins by arbitration rules, that is, it is more significant than the outstanding request message by Expander Port 1. As a result, Expander Port 0 issues a forward open message to Expander Port 1. However, since Expander Port 1 had already received an arbitrating (NORMAL) message while in the XL1:Request_Path state, the forward open message is ignored and a stall occurs.

<u>Solution</u>
Removing the restriction of transitioning to XL5:Forward_Open if an Arbitrating (Normal) message has been seen alleviates the problem. There are two cases to consider in determining that this is a valid solution. Following the example described above:

The first case is that the OAF that is forwarded to Expander Port 1 wins via arbitration rules over the OAF received by Expander Port 1. The second case being the OAF that is forwarded to Expander Port 1 loses via arbitration rules over the OAF received by Expander Port 1.

In the case that the forwarded OAF wins, with the restriction removed, the OAF will be received by the port connected and will discard the OAF it sent. This mechanism is already described in both the expander and link layer specification.

The second case can not occur because a forward open message will not be generated by Expander Port 0 destined for Expander Port 1 since Expander Port 0 did not win arbitration in the Expander Function.