Revision History
Revision 0 (25 June 2003) first revision

Related Documents
sas-r04 - Serial Attached SCSI revision 4
03-186r1 - SAS-1.1 Transport Layer Retries

Overview
This presents ladder diagrams for all the error recovery cases in 03-186.

Conventions:
® DATA means resend the DATA frame
For certain frame types, the RETRANSMIT bit is set to 1 when retransmitting.
COMMAND Frame NAK Received

Same as in SAS-1

<table>
<thead>
<tr>
<th>INITIATOR</th>
<th>TARGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND</td>
<td>NAK</td>
</tr>
<tr>
<td>NAK received for COMMAND Frame - resend COMMAND Frame</td>
<td></td>
</tr>
<tr>
<td>@ COMMAND</td>
<td>ACK</td>
</tr>
</tbody>
</table>
COMMAND Frame ACK lost (command running, but target hasn’t sent a frame yet)

Same as in SAS-1
Command Frame ACK Lost, target processes command returning RESPONSE, XFER_RDY, or DATA

**INITIATOR**  **TARGET**

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>ACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command completed successfully</td>
<td>RESPONSE</td>
</tr>
<tr>
<td>Tag is not ready for reuse (this ACK could still be lost)</td>
<td></td>
</tr>
</tbody>
</table>

**INITIATOR**  **TARGET**

<table>
<thead>
<tr>
<th>Write COMMAND</th>
<th>ACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command active at target</td>
<td>XFER_RDY</td>
</tr>
</tbody>
</table>

**INITIATOR**  **TARGET**

<table>
<thead>
<tr>
<th>Read COMMAND</th>
<th>ACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command active at target</td>
<td>DATA</td>
</tr>
</tbody>
</table>
COMMAND Frame NAK Lost (command not running)

- **INITIATOR**
  - COMMAND
  - ACK/NAK timeout
  - DONE (ACK/NAK TIMEOUT)
  - CLOSE
  - OPEN
  - QUERY TASK
  - @ COMMAND

- **TARGET**
  - × NAK
  - DONE
  - CLOSE
  - OPEN_ACCEPT
  - RESPONSE of FUNCTION COMPLETE
  - ACK

Command not active at target - initiator can resend the command.
COMMAND Frame Not Delivered

- COMMAND
- ACK/NAK timeout
- DONE (ACK/NAK TIMEOUT)
- CLOSE
- OPEN
- QUERY TASK

@ COMMAND

TARGET
- DONE
- CLOSE
- OPEN_ACCEPT
- RESPONSE of FUNCTION COMPLETE
- ACK

Command not active at target - initiator can resend the command.
RESPONSE Frame NAK Received

INITIATOR | TARGET
------- | -------
NAK | RESPONSE
    | NAK received for RESPONSE Frame - resend RESPONSE Frame with RETRANSMIT=1
    | @ RESPONSE
ACK | The RETRANSMIT bit is not strictly necessary here; it doesn’t help the initiator. It helps debugging with logic analyzers.
    | Tag is not ready for reuse (this ACK could still be lost)
RESPONSE Frame ACK Lost

**INITIATOR**  
- ACK  
- DONE  
- CLOSE  
- OPEN_ACCEPT  
  - ACK but ignore since RETRANSMIT=1
  - Tag is not ready for reuse (this ACK could still be lost)

**TARGET**  
- RESPONSE  
- ACK/NAK timeout  
- DONE (ACK/NAK TIMEOUT)  
  - resend RESPONSE Frame with RETRANSMIT=1  
- CLOSE  
- OPEN  
- ® RESPONSE
RESPONSE Frame NAK Lost

INITIATOR | TARGET
---|---
NAK | RESPONSE
ACK/NAK timeout
DONE (ACK/NAK TIMEOUT)
resend RESPONSE Frame with RETRANSMIT=1
CLOSE
OPEN
OPEN_ACCEPT
OPEN
ACK
Tag is not ready for reuse (this ACK could still be lost)
RESPONSE Frame Not Delivered

INITIATOR | TARGET
---|---
RESPONSE
ACK/NAK timeout
DONE (ACK/NAK TIMEOUT)
resend RESPONSE Frame with RETRANSMIT=1
CLOSE
OPEN
© RESPONSE

Tag is not ready for reuse (this ACK could still be lost)
XFER_RDY Frame NAK Received
SAS-1.0: target aborts command

NAK received for XFER_RDY Frame - resend XFER_RDY Frame with RETRANSMIT=1 and new TPTT

© XFER_RDY
XFER_RDY Frame ACK Lost
SAS-1.0: target aborts command

Could still send more data for TPTT=123; ignored by the target
Can we require initiator stop sending and avoid this?

When can target reuse TPTT=123?
Reuse it as soon as it gets the ACK for the RESPONSE for the command.

When does initiator stop sending TPTT=123 frames?
Must stop sending frames for a tag after it ACKs the RESPONSE frame
XFER_RDY Frame NAK Lost

INITIATOR | TARGET
----------|----------
NAK       | XFER_RDY
DONE      | DONE (ACK/NAK TIMEOUT)
CLOSE     | CLOSE
OPEN_ACCEPT | OPEN
ACK       | © XFER_RDY

ACK/NAK timeout

resend XFER_RDY Frame with RETRANSMIT=1 and new TPTT
XFER_RDY Frame Not Delivered

- INITIATOR
- TARGET

- XFER_RDY
- ACK/NAK timeout

- DONE
- DONE (ACK/NAK TIMEOUT)
- resend XFER_RDY Frame with RETRANSMIT=1 and new TPTT

- CLOSE
- CLOSE

- OPEN_ACCEPT
- OPEN

- ACK
- © XFER_RDY
Read DATA Frame NAK Received

Although DATA 400 is ACKed, the transport layer is not happy with the RO gap. For this logical unit, it does not abort the command, however; it should discard DATA 400 (but may store it where DATA 300 should have gone).

When DATA with Changing Data Pointer=1 arrives, the initiator resumes accepting read DATA at the correct destination.

Need to resend from RO=300. May also go back to last ACK/NAK balance point (RO=000) if that’s easier - the initiator won’t care. Should stop sending more data frames as soon as a NAK is received.
Read DATA Frame ACK Lost

Initiator honors Changing Data Pointer=1 from this logical unit

---

Got 4 ACKs, but target doesn’t know what was lost (an ACK or NAK, or which one since the last ACK/NAK balance point). Must restart from last ACK/NAK balance state (RO=000 here).
Read DATA Frame NAK Lost

Got 4 ACKs, but don’t know what was lost. Must restart from last ACK/NAK balance state (RO=000 here).
Read DATA Frame Not Delivered

Initiator shall not abort the command just because it detected an RO gap. It should discard all the data after the gap (but also might store it in the wrong location). No need to start NAKing.

Changing Data Pointer indicates the initiator can modify its data pointers and resume accepting data.

Got 4 ACKs, but don’t know what was lost. Must restart from last ACK/NAK balance state (RO=000 here).
Write DATA Frame NAK Received

INITIATOR | TARGET
---|---
ACK | XFER_RDY RO=000, TPTT=123, Retry DATA Frames=1
DATA RO=000, TPTT=123 |  
DATA RO=100, TPTT=123 |  
DATA RO=200, TPTT=123 |  
DATA RO=300, TPTT=123 |  
DATA RO=400, TPTT=123 |  

Go back to the frame that got NAKed, the last ACK/NAK balance point, or the beginning of XFER_RDY.
No need to close connection

© DATA RO=300, TPTT=123, Changing Data Pointer=1
© DATA RO=400, TPTT=123

ACK (000)
ACK (100)
ACK (200)
NAK (300)
ACK (400)
ACK (300)
ACK (400)
Write DATA Frame ACK Lost

<table>
<thead>
<tr>
<th>INITIATOR</th>
<th>TARGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>XFER_RDY RO=000, TPTT=123, Retry DATA Frames=1</td>
</tr>
<tr>
<td>DATA RO=000, TPTT=123</td>
<td></td>
</tr>
<tr>
<td>DATA RO=100, TPTT=123</td>
<td></td>
</tr>
<tr>
<td>DATA RO=200, TPTT=123</td>
<td></td>
</tr>
<tr>
<td>DATA RO=300, TPTT=123</td>
<td></td>
</tr>
<tr>
<td>DATA RO=400, TPTT=123</td>
<td></td>
</tr>
<tr>
<td>ACK (000)</td>
<td></td>
</tr>
<tr>
<td>ACK (100)</td>
<td></td>
</tr>
<tr>
<td>ACK (200)</td>
<td></td>
</tr>
<tr>
<td>ACK (300)</td>
<td></td>
</tr>
<tr>
<td>ACK (400)</td>
<td></td>
</tr>
</tbody>
</table>

ACK/NAK timeout

DONE (ACK/NAK TIMEOUT)

CLOSE

OPEN

OPEN_ACCEPT

DATA RO=000, TPTT=123, Changing Data Pointer = 1

DATA RO=100, TPTT=123

DATA RO=200, TPTT=123

DATA RO=300, TPTT=123

DATA RO=400, TPTT=123

ACK

ACK

ACK

ACK

ACK

ACK

ACK

ACK

ACK

Go back to the last ACK/NAK balance point, or the beginning of XFER_RDY.
Write DATA Frame NAK Lost

DATA RO=000, TPTT=123
DATA RO=100, TPTT=123
DATA RO=200, TPTT=123
DATA RO=300, TPTT=123
DATA RO=400, TPTT=123

ACK
DATA RO=000, TPTT=123
DATA RO=100, TPTT=123
DATA RO=200, TPTT=123
DATA RO=300, TPTT=123
DATA RO=400, TPTT=123

ACK/NAK timeout
DONE (ACK/NAK TIMEOUT)

ACK (000)
ACK (100)
ACK (200)
NAK (300)
ACK (400)

XFER_RDY RO=000, TPTT=123, Retry DATA Frames=1

DATA RO=000, TPTT=123
DATA RO=100, TPTT=123
DATA RO=200, TPTT=123
DATA RO=300, TPTT=123
DATA RO=400, TPTT=123

ACK (000)
ACK (100)
ACK (200)
ACK (300)
ACK (400)

DATA 400 is unexpected due to the gap, and shall be discarded by the target transport layer even though the link layer ACKs it.

Changing Data Pointer=1 signals that the target can go back and start accepting data again. It might internally discard the new writes up to the gap, or rewrite.

Go back to the last ACK/NAK balance point, or the beginning of XFER_RDY.
Write DATA Frame Not Delivered

DATA  RO=000, TPTT=123
DATA  RO=100, TPTT=123
DATA  RO=200, TPTT=123
DATA  RO=300, TPTT=123
DATA  RO=400, TPTT=123

ACK (000)
ACK (100)
ACK (200)
ACK (300)
ACK (400)

DATA 400 is unexpected due to the gap, and shall be discarded by the target transport layer even though the link layer ACKs it.

Changing Data Pointer=1 signals that the target can go back and start accepting data again. It might internally discard the new writes up to the gap, or rewrite.

ACK/NAK timeout
DONE (ACK/NAK TIMEOUT)
CLOSE
OPEN
OPEN_ACCEPT

ACK (000)
ACK (100)
ACK (200)
ACK (300)
ACK (400)

Go back to the last ACK/NAK balance point, or the beginning of XFER_RDY.
Write DATA Frame ACK Lost crossing RESPONSE

Command is done, but an ACK was apparently missed

Want to require initiator to stop on receipt of RESPONSE so target can reuse TPTT. If not, have to let it go back to the last ACK/NAK balance point, or the beginning of XFER_RDY.

Target cannot reuse TPTT yet unless initiator is required to stop sending frames for the command when it accepted the RESPONSE.

If initiator cannot be required to stop, all these are discarded with an unknown (obsolete) TPTT and tag.
Write DATA Frame ACK Lost crossing XFER_RDY

The XFER_RDY burst is done, but an ACK was apparently missed because another XFER_RDY just arrived.

Target cannot reuse TPTT=123 yet

Target can now reuse TPTT=123

Initiator must stop servicing the old XFER_RDY burst.