X3T9.2/92-026

Serial Storage Architecture
SSA-0 (Transport layer)
Version 1.0

12th December 1991

IBM Corporation

This document is preliminary and the contents are subject to change without notice. Enquiries, suggestions and requests for additional copies may be directed to the following:

Primary contact:
Colin G. Butterworth,
IBM Corporation,
5600 Cottle Road,
San Jose,
California 95193,
U.S.A.
Tel: (408)-256-3671

Secondary contact:
William G. Verdoorn,
IBM Corporation,
Hwy. 52 and 37th. St. NW,
Rochester,
Minnesota 55901,
U.S.A.
Tel: (507)-253-2197

IBM may use any information that you supply without incurring any obligation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program or service is not intended to state or imply that only IBM's product, program or service may be used. Any functionally equivalent product, program or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

This publication may refer to products that are announced but not currently available in your country. This publication may also refer to products that have not been announced in your country. IBM makes no commitment to make available any unannounced products referred to herein, or any product employing methods or devices described herein. The final decision to announce any product is based on IBM's business and technical judgment.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Director of Commercial Relations,
IBM Corporation,
Purchase,
New York 10577,
U.S.A.

The following terms in this publication are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM
MicroChannel

Contents

Introduction

This document describes the transport layer of the Serial Storage Architecture (SSA). SSA defines a hierarchy of serial interfaces for use within future storage sub-systems. It is divided into three layers, each described in a separate document:

SSA-0 A common transport layer used by SSA-1 and SSA-2.

SSA-1 A low-level order set for hard disk drives

SSA-2 A high-level (SCSI) command set for hard disk drives.

The SSA-0 transport layer is quite general and it may be used for other purposes within its distance and performance capabilities. SSA-1 and SSA-2 are alternative 'upper-level protocols'.

Characteristics

The main characteristics of the serial link defined by SSA-0 are summarized below:

- The link supports point-to-point communication only over a copper cable up to 10 Meters long. This distance is sufficient for connections between rack-mounted systems.
- The link provides full-duplex communication. This makes optimum use of the physical medium and it avoids performance overheads for arbitration and turn-around. Each node simply transmits on its outbound line subject to responses received on its inbound line.
- Depending on the implementation, the link can operate at 10 or 20 Mbytes/s in each direction. To allow compatibility between different nodes the operating speed is negotiated automatically. The architecture is open-ended so that higher speeds may be defined in the future.
- The unit of data transfer is a frame that contains a control field, an address field, an optional variable-length data field and a CRC field. Since the framing over-head is only 5 characters efficient communication is possible with a data field of, say, 64 bytes.
- The address field allows frames to/from different applications to be multiplexed freely on the same link. It also allows networks to be implemented by using one or more non-blocking frame switches.
- An interrupt facility is included to support circuit switching. For example, a low-cost cross-point switch could be used to share a number of controller paths amongst a larger number of devices. This allows the controller fan-out to be increased in a similar way to a multi-drop interface like the SCSI bus.
- The link provides excellent Reliability, Availability and Serviceability (RAS).

  All transmissions are checked for coding, protocol and CRC violations. An architected error recovery procedure provides transparent frame recovery after a transmission error. This enhances compatibility and it simplifies the design of the upper-level protocols.

  Power-On Self-Tests (POST's) within a node can perform a local wrap test. (A remote wrap test can also be provided by the upper-level protocol.)

  Voltage comparators in the line driver and receiver detect cable faults. The point-to-point connections facilitate fault isolation and concurrent maintenance.
- The transport layer is sufficiently general to support a variety of upper-level protocols, eg. the SSA-1 orders or the SSA-2 commands.

The transport layer is symmetrical when viewed from either node. However the upper-level protocol may impose a peer-to-peer or a master-slave relationship between the nodes.

- The protocol has been kept simple. This minimizes the implementation cost without sacrificing function or performance in the intended applications. A typical implementation including 4 128-byte frame buffers requires about 10K equivalent gates. Using 1 micron CMOS technology, several links can readily be integrated into a single LSI chip.
- The link is physically compatible with small form-factor devices. The cables and connectors are compact and a typical CMOS implementation will dissipate about 0.3 watts per node at 10 Mbytes/s.

Division of function

The SSA-0 transport layer defines the following functions:

- The protocol, ie. the frame format and flow control.
- Link management, ie. buffering, node states, resets, switching and error recovery.
- The physical medium, ie. encoding, modulation, clocking, line drivers/receivers, connectors and cables.

Each implementation of the link is responsible for defining:

- The data rate that is supported, ie. 10 MB/s or 20 MB/s or both.
- The amount of buffering that is provided in each node.
- The maximum length of the data field in each frame.

The following functions are defined by the relevant upper-level protocol (ie. SSA-1 or SSA-2):

- The interpretation of the undefined bits in the control field of each frame.
- The interpretation of the address field in each frame.
- The content of the data field in each frame, eg. commands, status and data.
- The conditions under which interrupts are signalled, if any.

Conventions

- The bits in an uncoded data byte are numbered 0 to 7 from right to left.
- The bits in an encoded character are designated a, b, c, d, e, i, f, g, h, j. Bit 'a' is transmitted on the line first and the other bits follow in the order shown.

Protocol

Frame format

Data bytes and protocol functions are encoded into 10-bit characters for transmission on the physical medium. The valid characters may be classified as follows:

- 256 'data' characters that represent a byte.
- 5 'protocol' characters that are used by SSA-0 to delimit frames, to provide flow control and to indicate when a node is in the disabled state.
- 7 'user-defined' characters that are available to the upper-level protocol.

See "8B/10B code" on page 29 for the bit patterns that are used to encode each valid character.

The two nodes communicate over the link in units called frames. A frame consists of a sequence of at least 4 data characters delimited by FLAG characters at each end.

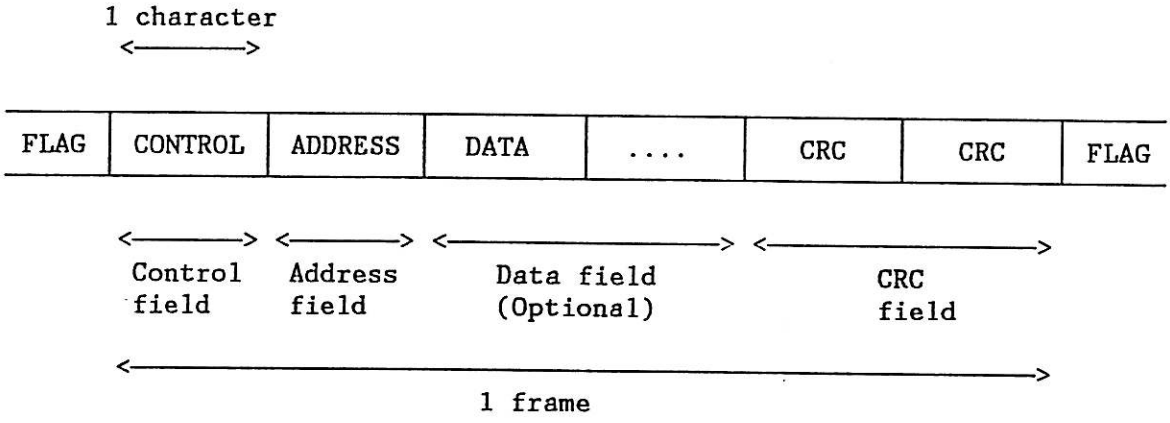


Figure 1. Frame format

A frame is divided into a sequence of 3 or 4 fields as shown in Figure 1:

Control field 1 data character, always present.

Address field 1 data character, always present.

Data field A variable number of data characters, optionally present.

CRC field 2 data characters, always present.

The shortest possible frame, with no data field, contains 4 data characters. If a node receives a frame containing less than 4 characters then it will indicate a 'protocol error'.

FLAG characters

Frames are delimited at each end by one or more FLAG protocol characters. The transition from a FLAG character to a data character marks the start of frame, and the transition from a data character to a FLAG marks the end of a frame. These will be referred to as the leading FLAG and the trailing FLAG respectively.

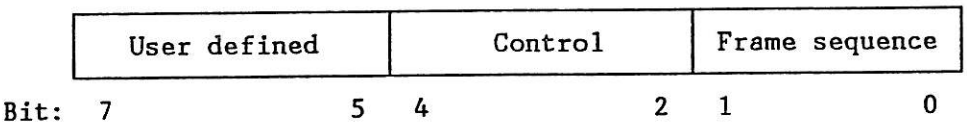
To minimize overheads a trailing FLAG can also be the leading FLAG of the next frame. Thus consecutive frames are separated by a minimum of one FLAG.

The bit pattern for the FLAG character has been chosen such that it does not occur at any bit position in an arbitrary sequence of any other valid characters. Therefore it serves the additional purpose of providing character synchronization.

FLAG characters are also sent when the link is idle in order to maintain synchronization at the receiver.

Control field

The control field is the first data character following a FLAG character. When it is decoded the resulting byte is interpreted as follows:



User defined These 3 bits are available for use by the upper-level protocol.

Control These 3 bits are binary coded to indicate various control functions:

- **0 0 0** No control function. This is the normal value in application frames.
- **0 0 1** Total reset. A Total Reset frame forces the destination node to re-initialize its internal state. It is typically used to recover from a catastrophic error. See "Total Reset" on page 18 for details.
- **0 1 0** Link reset. The Link Reset frame is used to report transmission errors and invoke the Link ERP. See "Link Reset" on page 19 for its effect.
- **0 1 1** Reserved.
- **1 0 0** Abort. This clears any inbound frames that may be buffered at the destination node. See "Abort" on page 19 for details.
- **1 0 1** Interrupt. This frame indicates that the source node is requesting service for an interrupt. See "Circuit switching" on page 17 for details.
- **1 1 0** Reserved.
- **1 1 1** Reserved.

Frame sequence These 2 bits are used to protect against lost or duplicate frames. They are incremented modulo 4 by the transmitter in each successive frame and checked by the receiver. See "Frame sequence numbers" on page 9 for details.

Address field

This is a single data character that immediately follows the control field.

The interpretation of the address field in 'application' frames, (see "Types of frame" on page 6), depends on the upper-level protocol. It normally indicates the destination of the frame within the receiving node. For example, the address field may indicate a software process, a DMA channel or another serial link. A portion of the address field may optionally be used to indicate the source of the frame within the transmitting node.

The address field in 'control' frames, (see "Types of frame" on page 6), is defined by the transport layer. Generally it does not contain addressing information.

Data field

The data field is optional and it is present only in application frames. If present it consists of a variable number of data characters that follow the address field. The content of the data field is controlled entirely by the upper-level protocol and it is of no relevance to the transport layer.

The maximum length of the data field depends on the implementation and the upper-level protocol. It is limited by the size of the receive buffer. Some implementations may have further restrictions, eg. the length must be an even number of characters. The maximum length to be used may be negotiated by the upper-level protocol or it may be pre-defined.

Due to the constant framing overhead the sustained data rate of the link depends on the data field length. See "Data Rate" on page 32.

If a node receives a frame with an incorrect length for the data field then it indicates a 'frame reject' error.

CRC field

The CRC field consists of 2 data characters that immediately precede the trailing FLAG. It is used to check the control, address and data fields. The destination must not regard any of the fields as valid until the CRC field has been received and checked. This will normally require each node to buffer at least 1 frame in the receiver.

The CRC field is calculated using the following polynomial:

$X^{16} + X^{15} + X^2 + 1$

The CRC registers are preset to all-ones at the start of each frame.

The transmitter can calculate the CRC on 8-bits in parallel using the algorithm illustrated in Figure 2 on page 6.

R(n) and S(n) represent the contents of the two 8-bit CRC registers. They are preset to all '1's, so R(0)=S(0)='FF'X. D is the input data byte over which CRC is being accumulated, 'AA'X in this example.

- The value of R(1) is calculated by XOR'ing the input data, D, with the contents of S(0) and then AND'ing the resultant byte (K') with each of the constant bytes shown in the 'P' column. This produces 8 bytes as shown in the T' column. Then, the bits contained within each of the bytes in the T' column are summed modulo 2, to give the bit contents of the required byte, R(1).

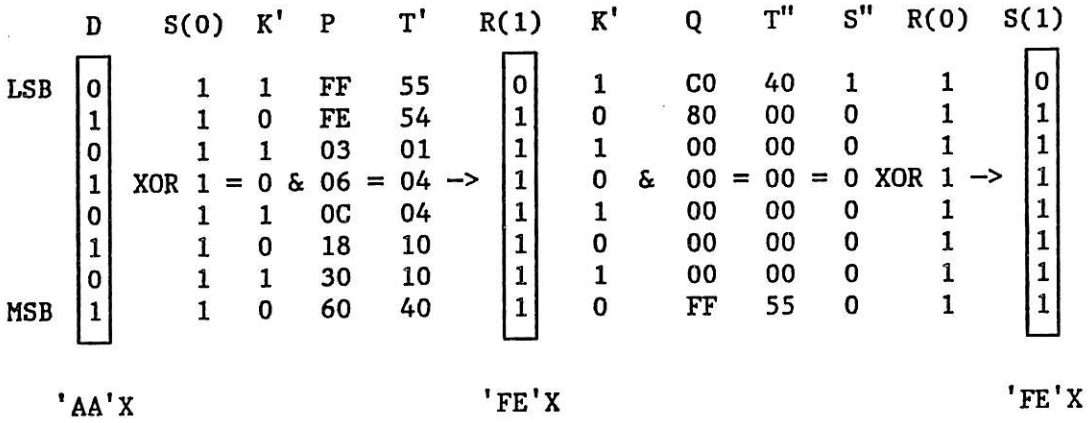


Figure 2. Parallel CRC calculation

- The value of S(1) is calculated by XOR'ing the input data with the contents of S(0) and then AND'ing the resultant byte (K') with the constant bytes in the 'Q' column. This produces 8 bytes as shown in the T'' column. The bits contained in each of the bytes in the T'' column are summed modulo 2, to produce an intermediate term S''. This is XOR'ed with R(0), the result being the required byte, S(1).

The constants shown in the P and Q columns depend on the polynomial being implemented. They are calculated by considering 8 individual shifts through a shift register implementation of division by the defined polynomial.

For hardware implementation, the AND operation is achieved by the presence/absence of connections and the modulo 2 addition is performed by XOR gates.

The CRC registers are encoded into 10-bit characters and transmitted in the order S(n) followed by R(n).

The receiver decodes the CRC field and checks it using the same algorithm as described above. The CRC registers are preset to all-ones at the start of each frame and accumulated over the control, address, data and CRC fields. The value of the CRC field for a frame does **not** include any interleaved protocol characters, (ie. responses or NUL characters), or user-defined characters. At the end of accumulation, provided that the incoming frame was received without error, the CRC registers should both contain zeros.

Types of frame

The transport layer distinguishes two types of frame:

Control frames These have a non-zero value in bits 7:2 of the control field. The contents of the address field are defined by the transport layer. The length of the data field must be zero. Otherwise the receiving node will indicate a 'frame reject' error.

Control frames are actioned immediately by the destination node and they are not subject to the normal pacing rules.

Application frames These have all zeros in bits 7:2 of the control field. The contents of the address field are defined by the upper-level protocol. The length of the data field can range from zero up to a maximum that depends on the implementation and the upper-level protocol.

Protocol

The link protocol defines the method of transmitting frames from a source node to a destination node. To implement the necessary flow control, the destination sends the source two responses for each received frame:

Acknowledgement A pair of consecutive ACK protocol characters

Receiver ready A pair of consecutive RR protocol characters

These protocol characters are used in pairs to protect the responses from being manufactured by transmission errors. A node only acts on a response when it has received both characters of the pair without any other intervening characters.

Response interleaving

In full-duplex operation a node may wish to send a response for a received frame whilst it is in the middle of transmitting another frame. In this case the transmitter gives priority to the response and interleaves it within the frame. This scheme reduces latency and the amount of buffering required by each node to sustain the full data rate of the link.

Since responses consist of protocol characters the receiver can easily separate them from the data characters that make up a frame.

Acknowledgements

The link protocol requires a node to acknowledge every valid received frame, apart from those specifying Total Reset or Interrupt. A frame is valid if it does not contain any of the errors listed in "Receiver errors" on page 21. The destination transmits an ACK response when it receives a valid frame. When the source receives the ACK response, it can discard the transmitted data.

Each node has two associated conditions, 'waiting for ACK' and 'ACK pending', that control acknowledgements as follows:

1. When a node enters the Ready state it clears 'waiting for ACK' and 'ACK pending'.
2. A node sets 'waiting for ACK' 2 character periods after it finishes transmitting the trailing FLAG of any frame except Total Reset or Interrupt. A node resets 'waiting for ACK' when it receives an ACK response. The corresponding transmit data may then be discarded.

   If a node does not receive the second ACK of the response within 10 us after setting 'waiting for ACK' then it recognizes an ACK time-out.

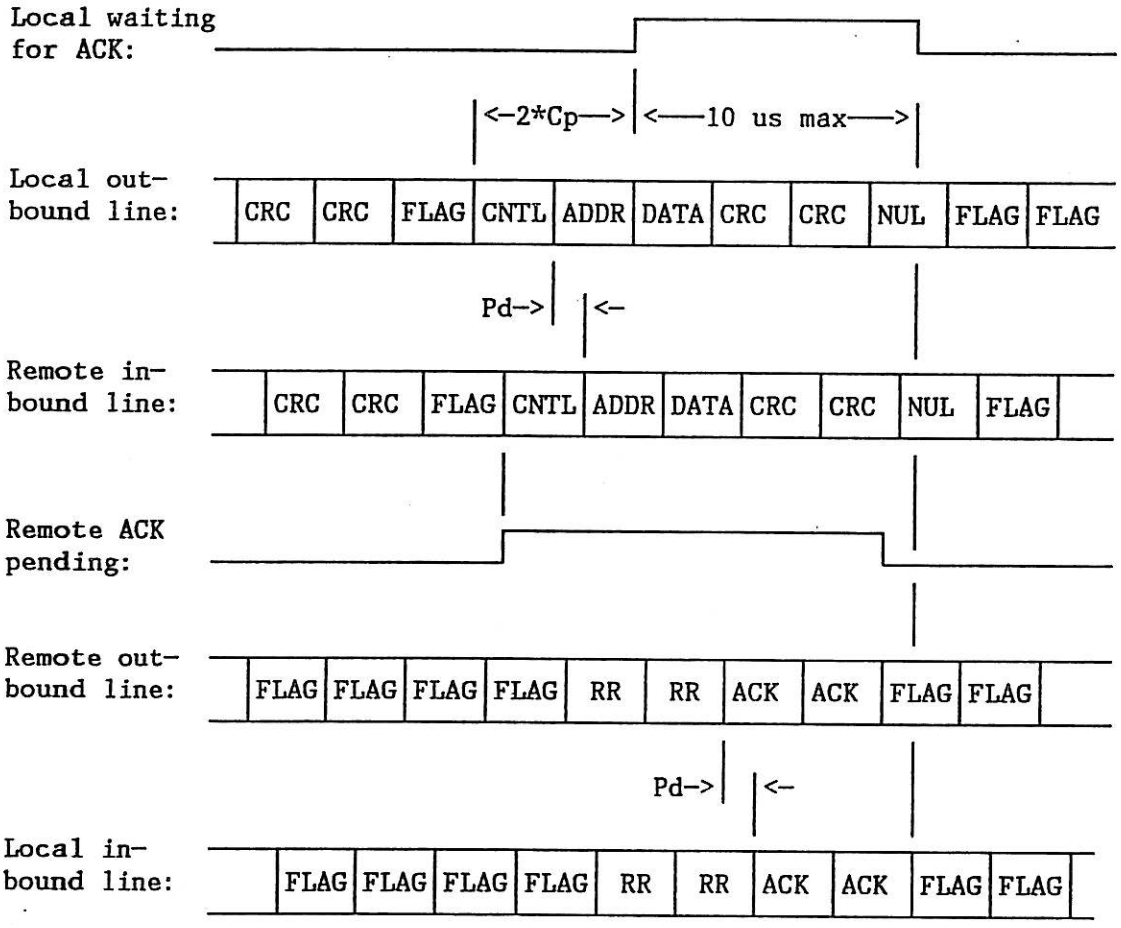
   If a node is still 'waiting for ACK' when it finishes transmitting the CRC field of the next frame then it must not transmit the trailing FLAG. Instead it sends NUL characters until either the ACK response is received or an ACK time-out occurs. If an ACK time-out does occur in this state then the node must abort the frame, as described in "Aborted frames" on page 11.

   This protocol guarantees that the transmitter can always associate each ACK response unambiguously with the corresponding frame independently of propagation delays, the transmission speed and the frame length.

If a node receives an ACK response when it is not 'waiting for ACK', or it receives only a single ACK character, then it recognizes a protocol error.

3. A node sets 'ACK pending' immediately when it receives the trailing FLAG of a valid frame in either of the following cases:
   - The node is in the Ready state and the frame does not specify Total Reset or Interrupt.
   - The node is in the Check state and the frame specifies Link Reset.
4. When 'ACK pending' is set the node must transmit an ACK response as soon as possible. However if an RR response is in progress then it must be completed first. 'ACK pending' is reset when the ACK response has been transmitted.

The timing requirements for the ACK response are shown in Figure 3.



'Pd' - Line propagation delay
'Cp' - Character period

Figure 3. Timing of ACK responses.

Pacing

Pacing ensures that the transmitter does not overrun the available buffer space in the receiver. The unit of pacing is a frame. The protocol requires a receiver to have buffer space for only 1 frame, although additional buffer space is generally required to sustain the full data rate of the link.

Each node has two conditions that control pacing, 'waiting for RR' and 'RR pending':

1. When a node enters the Ready state it resets 'waiting for RR' and 'RR pending' and transmits at least 10 FLAG characters. Then it is normally allowed to send one frame. (However see "Circuit switching" on page 17.)
2. A node may only start to send a frame when either of the following conditions is satisfied:
   - The node is in the Ready state and it is not 'waiting for RR'.
   - The frame is a control frame.

   'Waiting for RR' is set when a node transmits the control field of an application frame and reset when it receives an RR response.
3. When all of the following conditions are satisfied a node transmits an RR response immediately after the current character:
   - The node is in the Ready state and 'RR pending' is set.
   - Buffer space is available to receive at least one more application frame after the frame currently being received, if any.
   - The node is not currently transmitting an ACK response and 'ACK pending' is not set.

   'RR pending' is set when a node receives the control field of an application frame. It is reset when the node transmits an RR response.

While a node is 'waiting for RR' communication over its outbound line is effectively blocked for all application frames. Care is necessary if a node needs to send multiplexed frames to two or more processes that have limited buffering. If any process cannot accept further frames because its buffers are full then the destination node cannot safely return an RR response. Thus all communication is blocked until each process has buffer space available for at least one frame. The destination node can then send an RR response.

The problem above can be avoided by using the following pacing technique in the upper-level protocol. The source node does not send any data which has not been solicited. When a process in the destination node is ready to receive data it sends a message to the source. The message requests only as much data as the destination process currently has buffer space available.

Frame sequence numbers

These protect against frames being lost or duplicated by a transmission error. For example, if a FLAG is corrupted then two frames may be merged into one. A frame could be duplicated if a transmission error corrupts an ACK response. To guard against this the Link ERP needs to know whether the corresponding frame has actually been received by the destination.

The control field of each frame contains a 2-bit Frame Sequence Number (FSN). In normal operation the FSN increments modulo 4 in each successive frame.

Each node maintains a 2-bit Transmit Sequence Number (TSN) and it copies this into the FSN of each frame sent. The TSN is reset to '00'B in the Disabled state. It is incremented modulo 4 for each frame transmitted, regardless of any response received.

Each node also maintains a 2-bit Receive Sequence Number (RSN). The RSN is reset to '00'B in the Disabled state. It is incremented modulo 4 only when the receiver accepts a frame, ie. when it returns an ACK response. However the RSN must not be incremented by Link Resets. When a frame is received the hardware checks the FSN against the RSN as follows:

- If FSN = RSN then the sequence is correct. Providing that there is no other error then the frame is accepted and an ACK response is returned.
- If FSN ¬ = RSN and the frame does not specify Link Reset, Total Reset or Interrupt then one or more frames have been lost. The current frame is not acknowledged and providing that there is no other error the node recognizes a 'sequence error'.

The receiver ignores the FSN in Link Resets and Total Resets. For Interrupt frames the receiver copies the FSN into its RSN.

NUL characters

The transmitter is permitted to insert NUL protocol characters within a frame anywhere after the control field.

These are the only occasions when a NUL character would be invalid:

- A NUL must not be sent immediately following a FLAG. This guarantees that character synchronization will be maintained when the link is idle. It also means that a NUL must not precede the control field of a frame.
- A NUL must not be inserted between the two ACK or RR characters in a response.
- NUL characters must not be sent in the Disabled state.

For example, NUL characters are permitted between the control and address fields of a frame, between the two CRC characters and within control frames.

Except for the following cases, the receiver must ignore NUL characters by discarding them without changing its state:

- If the receiver decodes a NUL character and it has not received a data character since it received the last FLAG then it indicates a 'protocol error'.
- If the receiver decodes a NUL character when it is expecting the second ACK or RR of a response then it indicates a 'protocol error'.

NUL characters are useful in the following cases:

- If the transmitter has started to send a frame but the data needed to complete the frame is temporarily unavailable.
- If the transmitter is still waiting for an ACK response when it is ready to send the trailing FLAG of the next frame. See "Acknowledgements" on page 7.

User-defined characters

SSA-0 provides 7 user-defined characters for the upper-level protocol. The transmitter is permitted to insert these characters anywhere subject only to the following restrictions:

- They must not be sent in the Disabled, Enabled or Check states.
- They must not be sent in the first 10 characters after a node enters the Ready state.
- They must not be inserted between the two ACK or RR characters in a response.

- The 10 us ACK time-out is **not** extended by any user-defined characters.

User-defined characters are not included in the frame CRC. The Link ERP will not recover user-defined characters that have been corrupted by a transmission error. Consequently a user-defined character should only be used for a time-critical function when delivery does not need to be guaranteed. (For example, spindle synchronization in a disk array.) Otherwise a control or application frame should be used.

Aborted frames

If a node wishes to abort a frame after it has started to transmit the frame then it can insert an illegal character anywhere before the trailing FLAG. The illegal character aborts the frame and ensures that it is rejected by the remote node. This will subsequently cause both nodes to invoke the Link ERP. The Link Status Byte from the source node will indicate the cause of the error, eg. a hardware error or an ACK time-out. See "Link ERP" on page 23 for the actions taken then.

Examples

The following examples illustrate the link protocol. For compactness, the diagrams show a data field of 4 characters only.

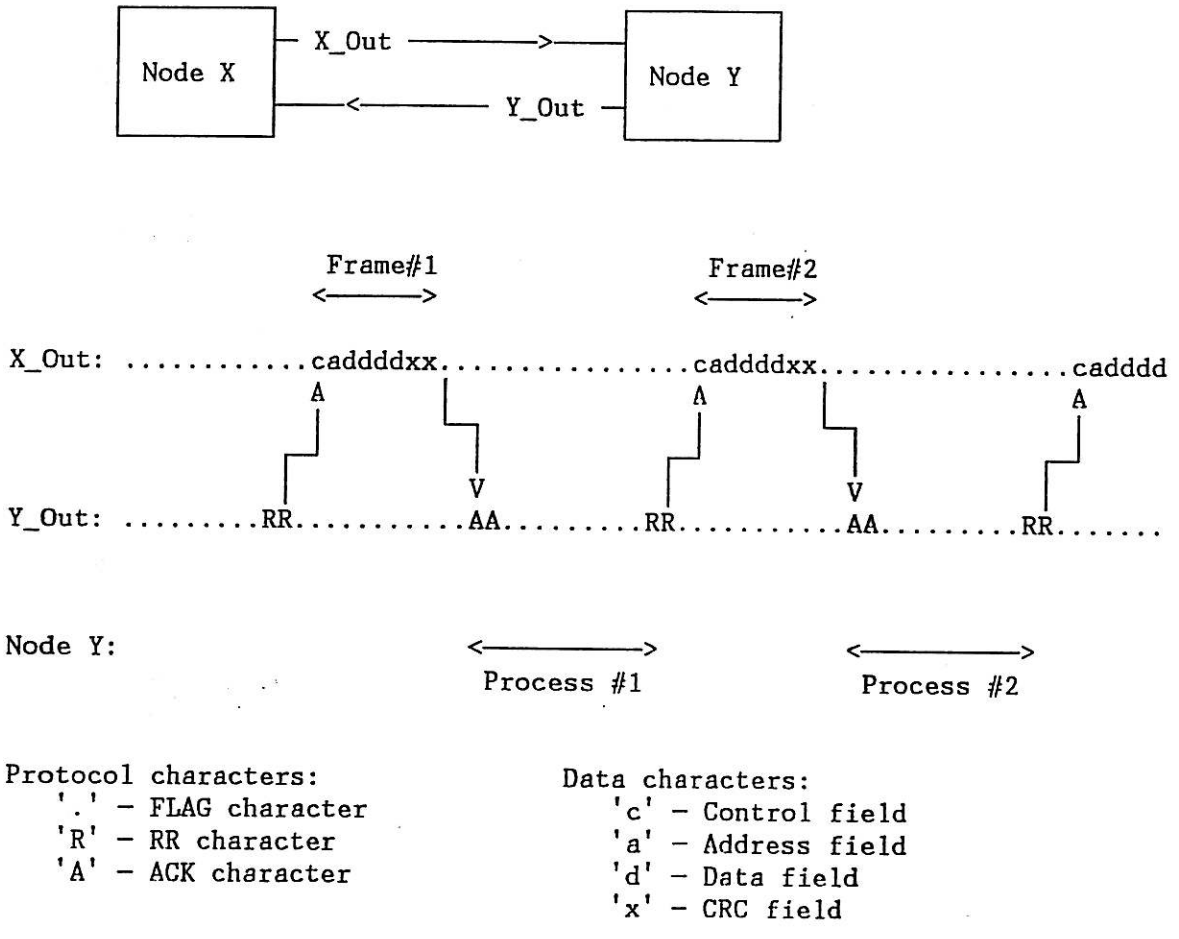


Figure 4. Half-duplex transfer with a single receive buffer

Figure 4 shows a half-duplex transfer with node X acting as a source and node Y acting as a destination. It is assumed that node Y has buffering for only a single frame in its receiver. Therefore when node Y receives a frame it delays sending the RR response until the destination process has emptied the buffer. In addition, node Y issues an ACK response immediately when it detects the trailing FLAG of each frame.

The next two examples assume that each node has a pair of A/B transmit frame buffers and a pair of A/B receive frame buffers. It is also assumed that the source/destination process is fast enough to fill/empty one buffer while the link empties/fills the other. Thus frames can be transmitted back-to-back, without any intervening delay.

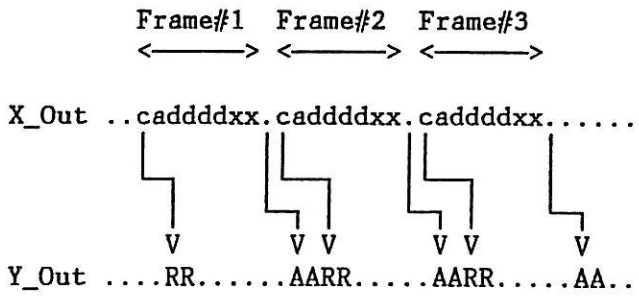


Figure 5. Half-duplex transfer with A/B buffering

Figure 5 shows a half-duplex transfer with node X acting as a source and node Y acting as a destination. Node Y sends an RR response as soon as it detects the start of a frame since its receiver has buffer space for another frame. This resets 'waiting for RR' in node X so that it can start sending another frame immediately after the trailing FLAG of the current frame.

Figure 6 shows a full-duplex transfer with node X acting as a source of 4 frames and node Y acting as a source of 3 frames. This scenario shows how responses can be interleaved within a frame.

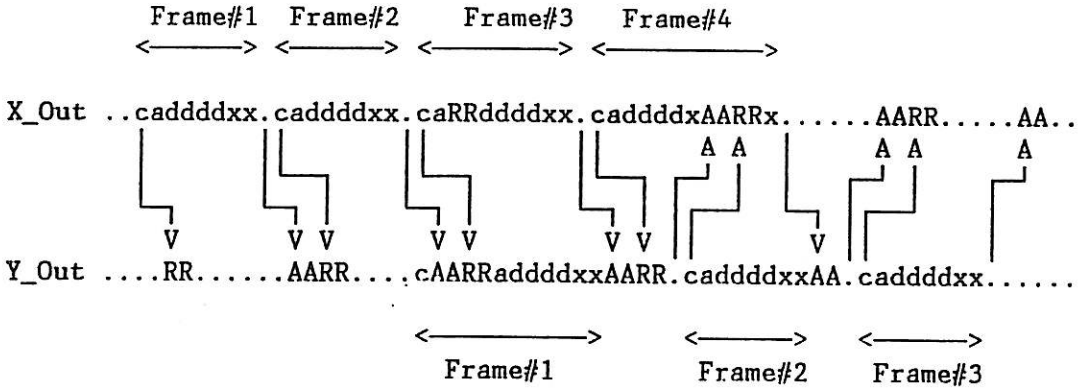


Figure 6. Full-duplex transfer with A/B buffering

Link management

Frame buffers

Each node must provide buffering to receive at least 1 application frame. The buffer must be large enough to accommodate the longest data field that is supported. The buffer is needed to allow the CRC field to be verified before the receiver transfers the data field to the application or acts on the control and address fields. Since the unit of pacing is a frame the buffer is also necessary to prevent overruns.

The source node must retain each frame until it receives the corresponding ACK response. If no ACK is received then the Link ERP may have to retransmit the last one or two frames.

To achieve continuous communication at the full bandwidth of the link it is generally necessary for each node to have additional buffering. For example, a node might have a pair of transmit buffers and a pair of receive buffers to provide 'A/B' buffering. One buffer of each pair is emptied/filled by the link while the other is filled/emptied by the application.

Buffer management

The transmit buffering must be carefully managed to allow correct recovery after an error. The Link ERP may need to retransmit or discard the last one or two frames that were transmitted just before an error. The link hardware must maintain sufficient status to identify the buffers containing these frames and the order in which they were transmitted.

For example, if there are N transmit buffers and the transmitter always accesses them in a strict cyclic sequence then the following two pointers provide sufficient information:

Transmit pointer This points to the buffer that is to be transmitted next. It is incremented modulo N each time a trailing FLAG is sent.

Retry pointer This points to the next buffer to be acknowledged. It is incremented modulo N each time an ACK response is received while 'waiting for ACK' is set. Normally it will follow the transmit pointer closely but when an error occurs it may lag by up to 2.

The above scheme is purely an example that will be used later in the description of the Link ERP. However if some buffers are reserved for a particular function (eg. commands) then it may be impossible to guarantee cyclic use. In this case an alternate solution is to keep a history log of up to 2 buffers that have been transmitted but not acknowledged.

Node states

Each node can be in one of four states, as illustrated in Figure 7 on page 14. The current state may be inspected by the node processor to determine the state of the link. The node processor may force state changes and the hardware may also change state automatically when certain events occur.

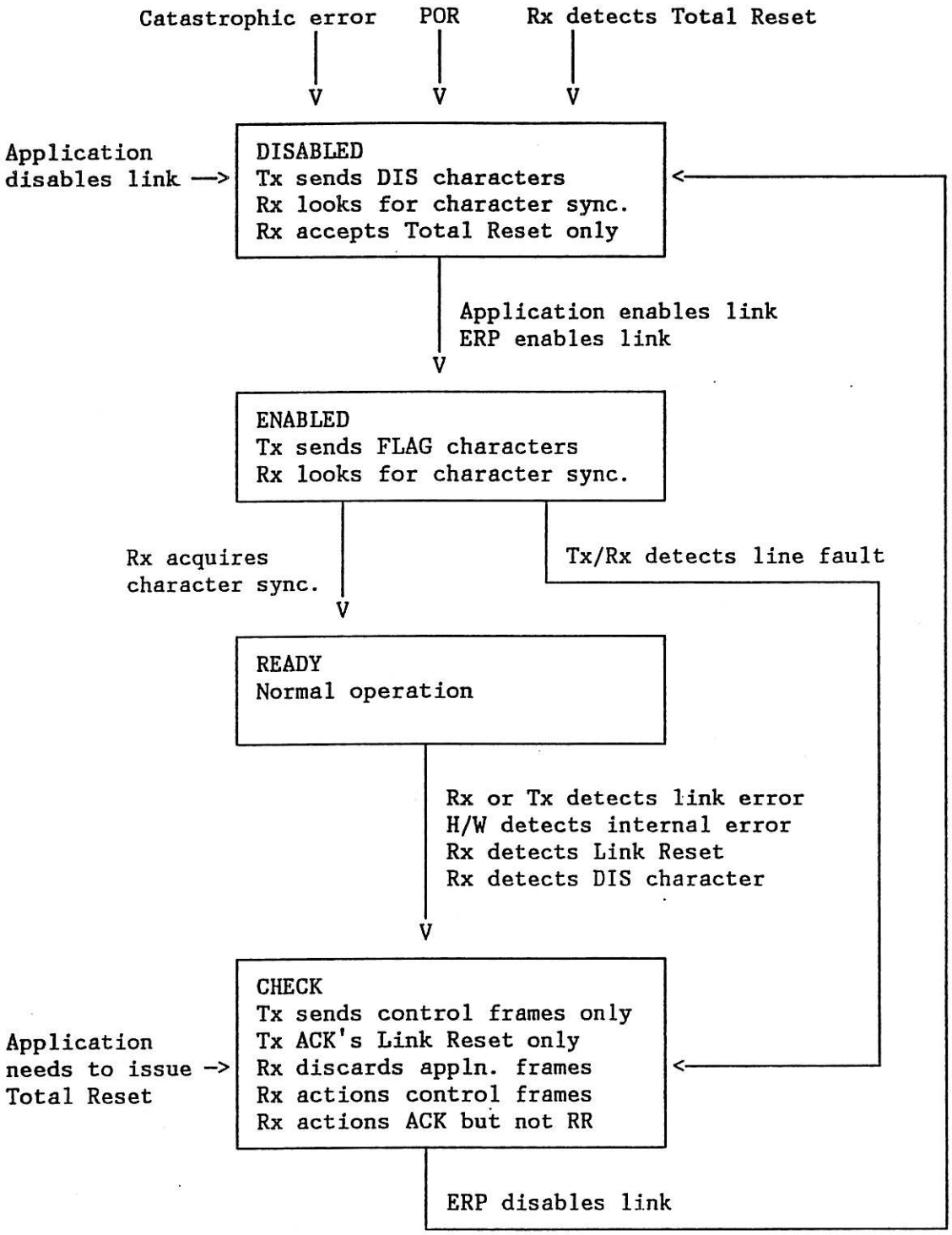


Figure 7. State transitions

Disabled

This is the power-on state before the link is made operational. The Disabled state is entered automatically after a Local Reset is performed or when a frame specifying Total Reset is received in any of the other states. It is also selected explicitly during the Link ERP.

In the Disabled state the transmitter sends DIS characters. (This maintains DC balance and it allows the remote receiver to maintain byte synchronization during the Link ERP.) The receiver attempts to acquire character synchronization, as described in "Data Rate" on page 32. If synchronization is achieved the receiver responds only to a Total Reset.

To guarantee recognition by the remote node, the minimum duration of the Disabled state is 5 character periods.

Enabled

This is a transient state on the way to making the link operational.

When a node is ready to begin communications the processor will change the hardware to the Enabled state. (See "Beginning communication" on page 16.) In the Enabled state the transmitter sends FLAG characters. If not achieved previously, the receiver continues to try to acquire character synchronization, as described in "Data Rate" on page 32. When character synchronization has been achieved the node hardware automatically enters the Ready state. The node processor may need to poll to detect this transition or the hardware may provide an interrupt.

Ready

This is the state for normal transmission and reception of frames.

In order to allow the remote node sufficient time to acquire character synchronization, when a node first becomes Ready it must transmit at least 10 FLAG characters before sending any frames or user-defined characters.

Check

This state is entered automatically when the hardware detects an error or it receives a Link Reset control frame. The transition to the Check state invokes the Link ERP. The link is then inoperable until the Link ERP successfully returns the hardware to the Ready state.

When the hardware enters the Check state the transmitter stops sending application frames after completing the current frame, if any. The transmitter then sends FLAG characters continuously, except in the following cases:

- If the receiver instructs it to acknowledge a Link Reset.
- If the node wishes to send a control frame.
- If the node wishes to send a user-defined character.

When in the Check state the receiver discards any incoming frames, unless they are control frames. The receiver also discards RR responses but ACK responses are accepted and actioned.

Wrap mode

Independently of the above states a node may be able to operate its link hardware in a Wrap mode. This is useful to perform a power-on self-test (POST) of the local hardware. In Wrap mode the transmitter output is internally connected to the receiver input. This allows half-duplex communication using the normal protocol. However in Wrap mode frames and their responses share the same line. Except for the line driver and receiver the link hardware can be fully tested without needing a remote node.

During the Wrap mode the outbound line is held at logic zero and the inbound line is totally ignored.

The Wrap mode should be selected with care at any time after the POST's since in some configurations if the node processor hangs it may then be impossible to reset it.

Beginning communication

The link hardware is in the Disabled state at power-on or after a Total Reset. Whenever a node wishes to leave the Disabled state to begin communications it must take the following steps:

1. Check that the line driver is not indicating a line fault. This would indicate that the remote node is not operational or that the cable is disconnected.
2. Ensure that buffer space is available to receive at least 1 application frame. (See "Pacing" on page 9.)
3. Put the link hardware into the Enabled state.
4. If the node can operate at more than one data rate then the operating speed is determined by the receiver during the Disabled and Enabled states.
5. Wait for the hardware to enter the Ready state.
6. Wait for at least 10 more character periods.
7. If necessary, negotiate with the remote node to determine the maximum length supported for the data field in each frame. The mechanism to achieve this is defined by the upper-level protocol since some devices may use several classes of application frames with different maximum lengths.
8. Application frames can now be transmitted.

Ending communication

Since the link has to be quiesced first the method of ending communication must be determined by the application. The following example is only intended to illustrate the steps that are necessary:

1. The node that wants to cease communications waits until the remote node has responded to all of its outstanding requests. It then sends a message requesting to shut-down the link.
2. The remote node waits until the local node has responded to all of its out-standing requests and then it returns a message acknowledging shut-down.
3. Both nodes then Disable their link hardware.

Circuit switching

The link is limited to point-to-point physical connections. This simplifies the protocol and avoids the electrical and RAS disadvantages of a multi-drop bus.

However the link does support switching. The increased connectivity can be used to attach more nodes or to provide redundant paths for high availability.

Frame-switching is already supported by the address field in each frame. This provides non-blocking communication (subject to bandwidth constraints) but at significant hardware cost for the switch.

The link also supports circuit switching. This may be appropriate for master/slave configurations, eg. a controller and several attached devices, as shown in Figure 8.

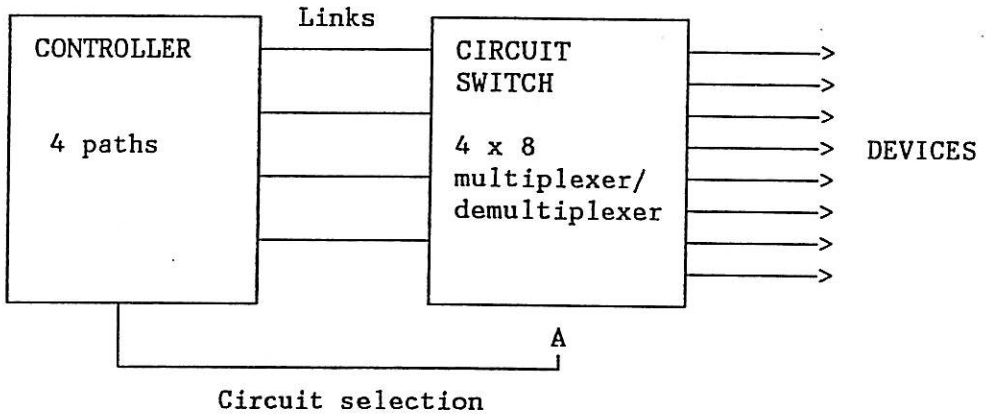


Figure 8. Circuit switching

The circuit switch requires little more than a multiplexer/demultiplexer and the additional line drivers and receivers. Thus the extra cost is low, particularly if the switch is integrated into the same chip as the 4 controller links. Each device can be accessed using any of the controller paths. Depending on the data traffic, the probability of all 4 paths being busy when a device needs one can be quite low. The devices may also incorporate some buffering. Thus the actual loss of performance may be small.

The following scenario illustrates the use of circuit switching:

1. The controller chooses any available path, selects a device via the switch and sends it a command. The command includes an instruction to disconnect. The controller is now free to use that path to communicate with other devices.
2. When the device has executed the command and it is ready to transfer data or present status then it sends Interrupt control frames repeatedly.
3. If the controller has an idle path it polls the disconnected devices via the switch to detect the Interrupts.
4. When the controller detects an Interrupt it sends a message to reset the Interrupt. The device can now initiate a data transfer or present status.

The transport layer provides an Interrupt facility to support circuit switching. The mechanism for instructing the slave to disconnect, the mechanism for resetting an Interrupt and the conditions for sending Interrupts are defined by the upper-level protocol.

While a slave is disconnected the switch sends FLAG characters to it continuously. This maintains DC balance on the line. The slave sends FLAG characters continuously until it has an Interrupt. The slave must not send any application frames while disconnected because the master may not be listening. The slave ignores link errors while it is disconnected.

Interrupt

A disconnected slave node can alert the master node that it has status to present by sending Interrupt control frames periodically. In order to allow the master node to acquire character synchronization when it is polling the slave must insert at least 10 FLAG's between successive Interrupt frames.

The format of an Interrupt frame is:

Control field Bits 4:2 are '1 0 1'B. The FSN increments normally.

Address field Ignored.

Data field Must be absent.

CRC field Must be correct.

Since the master node may not be listening the Interrupt frame does not expect an ACK or RR response.

When the receiver in the master node decodes an Interrupt frame it copies the FSN into its RSN.

Support for the Interrupt frame is optional. If a node that does not support Interrupt frames receives one then it rejects the frame.

Resets

This section defines the various types of reset that are associated with the link.

Local reset

This is an internal facility to reset all of the link hardware at power-on, after a link hardware error or after a catastrophic error in other node functions. It resets the frame buffers and their associated status. The node is also forced into the Disabled state and this resets the items listed in "Disabled state" on page 20.

Total Reset

The link architecture allows either node to reset the whole of the other node by sending a Total Reset control frame.

The format of the Total Reset frame is:

Control field Bits 4:2 are '0 0 1'B. The FSN is ignored.

Address field Ignored.

Data field Must be absent.

CRC field Must be correct.

The transmitter must be able to send the Total Reset frame when the node is in the Ready or Check states, regardless of whether it is 'waiting for RR'. Therefore if no characters are being received the node processor can force the hardware into the Check state in order to issue a reset.

The receiver must recognize a Total Reset when the node is in any state, regardless of whether its inbound buffers are full. If the frame is valid the node enters the Disabled state and the hardware issues a reset signal to the remaining logic and the node processor. ACK and RR responses may or may not be generated.

Implementation of the Total Reset function is optional. For example, if the nodes are arranged in a master/slave hierarchy then only the master may be able to issue a Total Reset to the slave. If a Total Reset is sent to a node that does not implement the function then the frame is ignored and no error is indicated.

Link Reset

Link Reset is used during the Link ERP to recover from a transmission error or a hardware error. The first node to recognize an error invokes its ERP which sends a Link Reset to alert the remote node. This invokes the Link ERP in the remote node which then returns a Link Reset. The Link Reset frame also contains the Link Status Byte that is needed by the Link ERP.

A Link Reset is designed to be repeatable in case further link errors occur during the ERP.

Either node can initiate a Link Reset. A Link Reset should only be initiated by a node that is in the Check state.

The format of a Link Reset frame is:

Control field Bits 4:2 are '0 1 0'B. The FSN is ignored.

Address field Contains the Link Status Byte. See "Link Status Byte" on page 22.

Data field Must be absent.

CRC field Must be correct.

The transmitter must be able to send a Link Reset frame when the node is in the Check state, regardless of whether 'waiting for RR' is set.

The receiver must recognize a Link Reset whenever it is in the Ready or Check states, regardless of whether its inbound buffers are full. If the frame is valid the node should return an ACK response and enter the Check state if it has not already done so. The ACK response is useful for fault isolation since it indicates that the link is operational even if the node processor has hung. The states of the frame buffers, the TSN and the RSN must not be changed by the receipt of a Link Reset since they are needed by the Link ERP.

Abort

This is a control frame to allow the upper-level protocol to abort an operation. Its effect on the transport layer is to purge any inbound frames that are still buffered at the receiving node. Additional functions are defined by the upper-level protocol.

The format of a Abort frame is:

Control field Bits 4:2 are '1 0 0'B. The FSN must be correct.

Address field Ignored.

Data field Must be absent.

CRC field Must be correct.

The transmitter must be able to send Abort when the node is in the Ready state, regardless of whether it is 'waiting for RR'.

The receiver must recognize an Abort frame when the node is in the Ready state, regardless of whether its inbound buffers are full. If the frame is valid the receiver purges all inbound frames that are buffered in the transport layer. The node should then return an ACK response. It should also return an RR response if 'RR pending' was set.

Disabled state

The Disabled state is entered transiently during the Link ERP. It is therefore convenient to use this state to reset those hardware functions that must be initialized before restarting communication. These functions are as follows:

1. The transmitter's protocol FSM is reset to idle.
2. The TSN is reset to '0 0'B.
3. The RSN is reset to '0 0'B.
4. 'RR pending' and 'waiting for RR' are both set to '0'B.
5. 'ACK pending' and 'waiting for ACK' are both set to '0'B.

The frame buffers and their associated status must not be affected in the Disabled state since they are needed when communication restarts.

FLAG characters

In all states the receiver continuously monitors the inbound line for FLAG characters. Provided that character synchronization has been achieved the following actions are taken when a FLAG is detected:

1. If a data character has been received since the previous FLAG then the receiver considers the current frame to be complete.
2. The receiver's protocol FSM and CRC registers are reset. This ensures that the receiver is ready for the next frame.

Link errors

Except where explicitly stated the following errors are only detected when the node is in the Ready state prior to the error. Errors are generally ignored if the node is not in the Ready state or with circuit-switching, if a slave node is disconnected.

When an error is detected the hardware will enter the Check state and interrupt the node processor. No further application frames are accepted or acknowledged until the node returns to the Ready state.

Hardware error

This error is indicated when a node detects an internal hardware error, eg. a parity check.

Line fault

This error is indicated when the line driver detects an invalid voltage and the link hardware is not in the Disabled state. The cable may be open or short circuit or the remote node may be powered off.

ACK time-out

This error is indicated when the source node does not receive an ACK response within the specified time of transmitting the trailing FLAG of a frame other than Total Reset or Interrupt.

Receiver errors

Loss of synchronization. This is indicated when the clock recovery circuits in the receiver detect a synchronization error in the Ready or Check states.

Code violation. This error is indicated if none of the receiver errors above has occurred and the receiver decodes a character which is not in the defined alphabet or a character which causes a disparity violation.

Protocol error. This error is indicated if none of the receiver errors above has occurred and a node receives an incorrect sequence of valid characters as listed here:

1. A short frame with less than 4 data characters between 2 FLAG characters. This may be caused by noise corrupting or manufacturing a FLAG.
2. An application frame and no buffer is available, ie. when 'RR pending' is set.
3. An unexpected ACK response, ie. when 'waiting for ACK' is reset.
4. An isolated ACK character. If an ACK response is corrupted then the transmitter will also detect an ACK time-out.
5. An isolated RR character.
6. A NUL character with no intervening data character since the last FLAG.

   One half of the link will hang if an RR response is lost without any errors being detected, eg. if the RR characters are changed to FLAG characters while the link is idle. This is extremely unlikely and therefore no recovery is provided in the transport layer. Instead the application should provide a time-out for each operation in progress.

CRC error. This error is indicated if none of the receiver errors above has occurred and a received frame has bad CRC.

Sequence error. This error is indicated when a received frame has FSN ¬ = RSN, none of the receiver errors above has occurred, and the frame does not specify Total Reset or Link Reset. A previous frame has probably been lost.

Frame reject. This error is indicated when a frame is received correctly with none of the receiver errors above but the frame is unacceptable for any of the following reasons:

1. The frame is too long to fit in the available buffers. Note that the receiver must continue to accumulate the CRC after the buffer has overflowed in order to verify that there hasn't been a transmission error, eg. a corrupted FLAG.
2. The frame length is otherwise unacceptable to the implementation, eg. odd when it must be even.
3. The control field specifies a function that is not implemented or invalid.
4. The length of the data field in a control frame is not zero.

5. The address field in an application frame specifies a destination that is currently invalid or not implemented.

Errors in this class are generally due to programming, synchronization or compatibility problems.

Error recovery

The transport layer of the serial link defines a Link ERP to recover **transmission** errors at the frame level. This has the following benefits:

- The upper-level protocol is simplified since recovery is transparent if it is successful.
- There is normally no need to terminate any operations when an error occurs. However an unbuffered device may overrun as a result of the extra time taken by the Link ERP.
- There is no uncertainty about the state of the application in the remote node.
- The compatibility of different link implementations is enhanced.

Note that **hardware errors**, such as parity checks, may not be recoverable at the frame level. Data may have been lost or unknown state changes may have occurred. In this case the application must still perform the recovery using the upper-level protocol. It is suggested that the operations in progress are terminated and restarted or repeated. This is an acceptable solution since hardware errors are much less frequent than transmission errors.

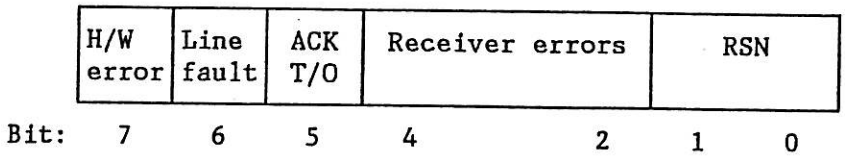
Principles

The basic principles of error recovery are as follows:

1. In normal operation the transmitter does not discard a frame until it has received an ACK response. This indicates that the frame has been received correctly by the destination node. Therefore when an error occurs the affected frame(s) are still available for retransmission without reference to the upper-level protocol.
2. When an error is detected both nodes enter the Check state, invoke the Link ERP and exchange status by means of Link Resets.
3. Recovery is performed separately for each line. Each node is responsible for recovering frames that were lost on its outbound line. Because the transmitter is allowed to start sending another frame before it receives an ACK response, up to 2 frames may need to be retransmitted.
4. Before restarting communication the Link ERP forces the hardware into the Disabled state. This synchronizes the two nodes and allows an orderly restart with the same mechanisms that are used at power-on.
5. The link protocol and ERP are designed to minimize the chances of losing or duplicating any frames when an error occurs. However the upper-level protocol should protect against these events wherever possible. For example, the byte count can be checked for zero at the end of a data transfer and time-outs can be used to detect lost messages.

Link Status Byte

During error recovery the Link ERP in each node builds a Link Status Byte and sends it to the other node in the address field of a Link Reset frame:



H/W error When '1', this bit indicates that the node detected an internal hardware error.

Line fault When '1', this bit indicates that the node detected a fault on the inbound line or the outbound line. It is provided for diagnostic information only and it is not referenced by the Link ERP in the destination node.

ACK T/O When '1', this bit indicates that the transmitter timed-out while waiting for an ACK response. It is provided for diagnostic information only and it is not referenced by the Link ERP in the destination node.

Receiver errors This field contains a 3-bit code to identify the first error detected by the receiver:

0 0 0	No error
0 0 1	Loss of synchronisation
0 1 0	Code violation
0 1 1	Protocol error
1 0 0	CRC error
1 0 1	Sequence error
1 1 0	Frame reject
1 1 1	Reserved

When two or more errors occur simultaneously the lowest number is reported.

RSN This is the receive sequence number for the last frame that was acknowledged by the node, excluding Link Resets. It is needed by the Link ERP in the remote node.

Link ERP

Error recovery is symmetrical for both nodes. When an error occurs both nodes will enter the Check state and invoke the Link ERP. It is expected that the Link ERP will normally be implemented in software running on the node processor. However the functions could conceivably be performed by a hardware FSM if performance is critical.

If the ERP determines that a transmission error occurred then it attempts to recover the error itself. If recovery is successful the Link ERP terminates and the upper-level protocol continues unaware of the error.

The ERP cannot recover some errors transparently, eg. hardware errors or permanent line faults. In these cases the ERP exits. The application should then perform a reset and abort the operations in progress. The ERP has been carefully designed so that both nodes will always recognize an unrecoverable error and remain synchronized. To facilitate cross-referencing by other documents each 'ERP Exit' is identified by a name in the description below.

The first (or only) node that detects the error enters the Check state and invokes its Link ERP. The Link ERP functions as follows:

1. The ERP waits until the transmitter has finished sending the current frame, if any.
2. The ERP then builds the Link Status Byte by reference to the hardware.
3. If the line driver or receiver has detected a line fault then the ERP tries to reset the error. If this fails then the application is alerted via an ERP exit ('Permanent line fault').
4. The ERP checks whether the receiver is detecting DIS characters. If so, the remote node may have entered the Disabled state due to a catastrophic error. The application is alerted via an ERP exit ('Remote node Disabled').
5. The ERP saves the local TSN for use later.
6. The ERP constructs a Link Reset frame containing the Link Status Byte. It then sends two successive Link Reset frames to the remote node. Repeating the Link Reset in this way allows for either frame to be corrupted by noise. The remote node should now enter the Check state, if it has not already done so. Either way it will invoke its Link ERP and return two Link Resets containing the remote Link Status Byte.
7. The ERP checks whether a Link Reset has already been received from the remote node. If not the ERP starts a time-out and waits to receive a Link Reset. If no Link Reset has been received within 1 ms after the local node sent its second Link Reset then the application is alerted via an ERP exit ('Link Reset failed').
8. The implementation must protect against the ERP looping if there is a permanent error. Since both nodes are always involved in error recovery it is sufficient if only one node provides this protection, eg. the master node in a hierarchical system.

   The following is an example of one method that can be used. Each invocation of the ERP increments a retry counter that is reset to zero periodically by a timer. If the number of retries in one period of the timer exceeds some maximum value then the ERP waits 10 ms to ensure the remote node recognizes that retry is being aborted. The application is then alerted via an ERP exit ('Retry limit exceeded'). This scheme also protects against excessive use of the ERP in the event of severe external noise.
9. If either node has detected a hardware error then the application is alerted via an ERP exit ('Hardware error'). To allow the fault to be isolated the ERP exit should also indicate which node detected the error. (Local node, remote node or both.)
10. If either node has indicated 'frame reject' then further communication may be meaningless. The upper-level protocol is alerted via an ERP exit ('Frame rejected'). The ERP exit also indicates the node that detected the error. (Local node, remote node or both.)
11. Otherwise the ERP calculates the number of frames that have been started but not acknowledged. When using strict cyclic buffers this can be obtained from,

    ```
    Q = (Transmit_pointer - Retry_pointer) modulo N
    ```

    where N is the number of transmit buffers that are provided. Q should be 0, 1 or 2 frames.

    The ERP also calculates the number of frames that have been transmitted but not received,

    ```
    P = (Saved_local_TSN - Remote_RSN) modulo 4
    ```

    P should be less than or equal to Q.

    If either of these checks fails the ERP waits 10 ms to ensure that the remote node recognizes an unrecoverable error. The application is then alerted via an ERP exit ('Invalid retry status').

12. Otherwise the ERP arranges to resend the lost frames by subtracting P from its transmit pointer, modulo N.
13. Those outbound buffers that do not need to be retransmitted must now be discarded using the following algorithm:

```
Do while Retry_pointer ¬= Transmit_pointer;
  Deallocate buffer at Retry_pointer;
  Increment Retry_pointer modulo N;
End;
```

14. If the node has received a frame containing any of the errors listed in "Receiver errors" on page 21 then it must be discarded. The appropriate inbound buffer may be deallocated automatically by the receiver hardware or the ERP may have to do it explicitly. Otherwise the ERP does not need to deal with the inbound buffers. If any are full they will be emptied by the upper-level protocol.
15. The ERP Disables the link and resets all of the latches for hardware errors, ACK time-out and receiver errors.
16. The ERP waits until the remote node enters the Disabled state, as indicated by the receiver detecting DIS characters. This is required to synchronize the two Link ERP's and prevent the transmitter sending a frame while the remote node is still in the Check state.

    If the receiver does not detect DIS characters within 1 ms after the local node is Disabled then the application is alerted via an ERP exit ('Time-out waiting for Disabled state'). The remote node may have detected an unrecoverable link error.
17. Otherwise the ERP checks that buffer space is available for at least one inbound frame and then it Enables the link.
18. The ERP waits for the link to become Ready. This indicates that the remote node has completed its recovery.

    In a hierarchical system the slave node may wait indefinitely for the Ready state. Otherwise a time-out should be provided as follows. If the link does not become Ready within 1 ms after the local node entered the Enabled state then the application is alerted via an ERP exit ('Time-out waiting for Ready state'). This may indicate that the remote node has powered-off or suffered a catastrophic error.
19. Otherwise the ERP terminates successfully.

Path-lengths. To avoid false time-outs the implementation must ensure that the following paths in the Link ERP do not exceed 1 ms each:

1. From receiving a Link Reset to sending the second Link Reset.
2. From the later of sending the second Link Reset or receiving a Link Reset to disabling the local node.
3. From the later of detecting DIS characters or disabling the local node to enabling the local node.

Example. Figure 9 on page 26 illustrates the operation of the Link ERP. The local node sends 2 frames back-to-back. The remote node receives them correctly but the ACK response for the first frame is corrupted. The local node then detects a code violation. No frames need to be retransmitted since only the ACK response was lost.: To illustrate the operation of the transmit pointer (TP) and the retry pointer (RP) it is assumed that the local node has 4 transmit buffers. The remote node has 2 receive buffers.

LOCAL NODE	O/B LINE ——>	I/B LINE <——	REMOTE NODE
TP=0, RP=0 TSN=0, RSN=0			TP=0, RP=0 TSN=0, RSN=0
Tx frame 0	Frame 0		
	"	RR response	Tx RR response
	"		
	"		
TSN=1, TP=1	"		RSN=1
Tx frame 1	Frame 1	** Error **	Tx ACK response
Code violation	"	RR response	Tx RR response
ACK time-out	"		
Enter Check state	"		
TSN=2, TP=2	"		RSN=2
		ACK response	Tx ACK response
Invoke Link ERP Wait for Tx complete Assemble LSB			
Tx Link Reset	Link Reset		
		RR response	Tx RR response
		ACK response	Tx ACK response
			Enter Check state
Tx Link Reset	Link Reset		
Wait to Rx Link Reset		RR response	Tx RR response
"		ACK response	Tx ACK response
"			Invoke Link ERP
"			Wait for Tx complete
"			Assemble LSB
"		Link reset	Tx Link Reset
Tx RR response	RR response		
Tx ACK response	ACK response		
		Link reset	Tx Link Reset
Tx RR response	RR response		Q=0, P=0, TP=0
Tx ACK response	ACK response		Discard 0 frames
Q=2, P=0, TP=2			RP=0
Discard 2 frames RP=2			
Enter Disabled state	DIS	DIS	Enter Disabled state
RSN=TSN=0	"	"	RSN=TSN=0
Wait to detect DIS	"	"	Wait to detect DIS

.
.
.

(Continued on next page)

Figure 9. Recovery from corrupted ACK response (Part 1)

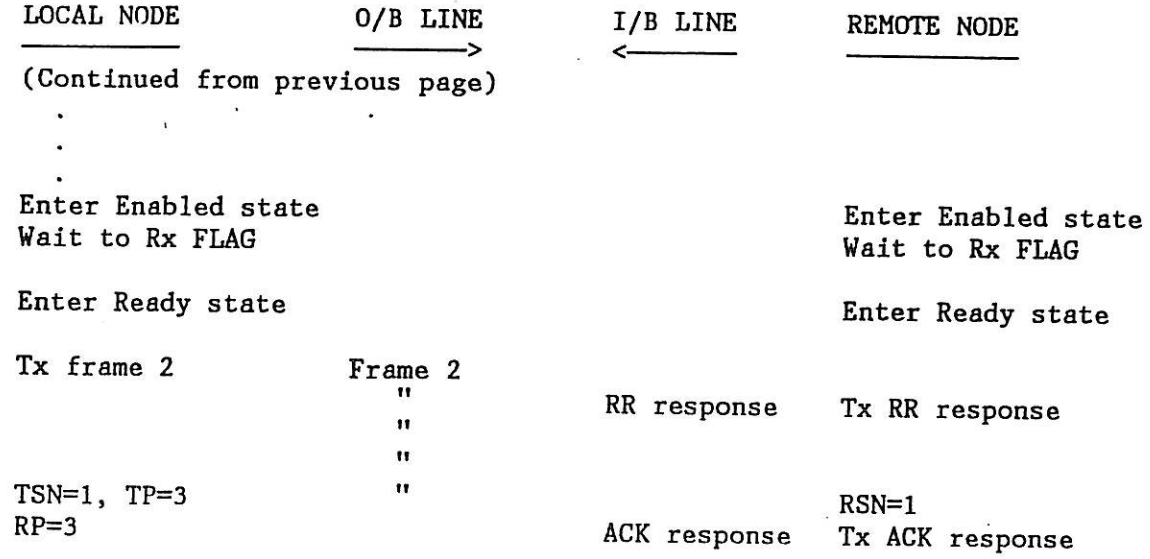


Figure 10. Recovery from corrupted ACK response (Part 2)

Physical medium

8B/10B code

Synchronous clocking restricts the bit patterns that the transmitter can use since it is undesirable to have long sequences of consecutive zeros or ones. Hence an encoding algorithm is required to convert the arbitrary data that one may wish to send into patterns suitable for transmission.

The serial link uses an 8B/10B code which is described fully in Reference (1). The 8B/10B code provides a (0,4) run-length constraint. This means that the minimum run-length of consecutive zeros or ones is 1 bit and the maximum run-length is 5 bits. The 8B/10B code also provides DC-balance with a maximum digital sum variation of 6.

The 8B/10B mapping can be achieved efficiently by partitioning each incoming byte into a 5-bit sub-block and a 3-bit sub-block. The sub-blocks are then encoded with separate 5B/6B and 3B/4B encoders.

In order for the encoded data to be DC-balanced, the combined 8B/10B encoder accumulates a running 'disparity'. As a result, the encoder and decoder require sequential logic in addition to combinatorial logic. The disparity of a sub-block is the difference between the number of '1's and '0's in the encoded data; positive and negative disparities refer to an excess of '1's and '0's respectively. For both the 6B and the 4B sub-blocks the permitted disparity is 0, +2 or -2. The 8B/10B code requires that the polarity (positive or negative) of encoded sub-blocks with non-zero disparity must alternate. The 4B and 6B encodes with non-zero disparity are allocated in complementary pairs to allow this. With a few exceptions, encoded sub-blocks with zero disparity are generally independent of the running disparity.

Data characters

Bits 7:3 are encoded into bits a, b, c, d, e, i by the 5B/6B encoder using the rules in Figure 11 on page 30.

The first encode is used when the polarity of the running disparity on entry to the sub-block is as shown in the column 'D-1'. (The polarity of the running disparity is equal to the polarity of the most-recent sub-block that had non-zero disparity.) An 'x' in column 'D-1' means that the first encode is selected independently of the entry disparity. The column 'D0' indicates the disparity of the first encode. This can be 0, +2 or -2.

The alternate encode is used if the polarity of the running disparity does not match the entry in column 'D-1'. In this case the encode disparity is the complement of column 'D0'.

Bits 2:0 are encoded into bits f, g, h, j by the 3B/4B encoder using the rules in Figure 12 on page 31. This is interpreted in the same way as the 5B/6B table.

Note that although symbols D.7 in the 5B/6B encodes and D.3 in the 3B/4B encodes have zero disparity they are assigned complementary encodes. This reduces the maximum digital sum variation and eliminates certain undesirably long run lengths.

The encode for bits 2:0 = '1 1 1'B in the 3B/4B encoder has two possible values, each of which also has a complement. Symbol D.A7 replaces D.P7 under either of the following conditions:

In the previous sub-block e = i = 1 and the running disparity is negative.

In the previous sub-block e = i = 0 and the running disparity is positive.

Symbol	Input 76543	D-1	Encode abcdei	D0	Alternate abcdei
D.0	00000	+	011000	-	100111
D.1	10000	+	100010	-	011101
D.2	01000	+	010010	-	101101
D.3	11000	x	110001	0	
D.4	00100	+	001010	-	110101
D.5	10100	x	101001	0	
D.6	01100	x	011001	0	
D.7	11100	-	111000	0	000111
D.8	00010	+	000110	-	111001
D.9	10010	x	100101	0	
D.10	01010	x	010101	0	
D.11	11010	x	110100	0	
D.12	00110	x	001101	0	
D.13	10110	x	101100	0	
D.14	01110	x	011100	0	
D.15	11110	+	101000	-	010111
D.16	00001	-	011011	+	100100
D.17	10001	x	100011	0	
D.18	01001	x	010011	0	
D.19	11001	x	110010	0	
D.20	00101	x	001011	0	
D.21	10101	x	101010	0	
D.22	01101	x	011010	0	
D.23	11101	-	111010	+	000101
D.24	00011	+	001100	-	110011
D.25	10011	x	100110	0	
D.26	01011	x	010110	0	
D.27	11011	-	110110	+	001001
D.28	00111	x	001110	0	
D.29	10111	-	101110	+	010001
D.30	01111	-	011110	+	100001
D.31	11111	-	101011	+	010100

Figure 11. 5B/6B encoding rules

Special characters

The 8B/10B code provides 12 'special' characters that comply with the constraints. These characters are shown in Figure 13 on page 32.

3 of the special characters contain a sequence (bits a, b, c, d, e, i, f = '0 0 1 1 1 1 1'B) which does not occur in any bit position, either within other characters or in any overlap of characters. They are called 'comma' characters and their properties make them suitable for establishing byte synchronization.

SSA-0 uses 5 'protocol' characters. The other 7 special characters are 'user-defined' characters which are available to the upper-level protocol.

Symbol	Input 210	Encode D-1	fghj	D0	Alternate fghj
D.0	000	+	0100	-	1011
D.1	100	x	1001	0	
D.2	010	x	0101	0	
D.3	110	-	1100	0	0011
D.4	001	+	0010	-	1101
D.5	101	x	1010	0	
D.6	011	x	0110	0	
D.P7	111	-	1110	+	0001
D.A7	111	-	0111	+	1000

Figure 12. 3B/4B encoding rules

Initialization

On entry to the Disabled state the running disparity of the transmitter is not known to the receiver. Therefore the receiver must acquire character synchronization in the Disabled or Enabled states by accepting one FLAG character with either disparity. This establishes the running disparity in the receiver. To be considered valid subsequent characters must then follow the disparity rules.

The receiver must follow a similar procedure to re-initialize its running disparity after it has detected a disparity error.

Code violations

The receiver should check for illegal 6B sub-blocks, 4B sub-blocks or special characters which are outside of the defined alphabets.

The receiver should also check that each sub-block and special character is consistent with the required entry disparity. Note that the error leading to a violation may have occurred in a previous character.

Finally, the receiver should check that the 4B sub-blocks D.P7 and D.A7 are consistent with the previous 6B sub-block.

Code violations are ignored until character synchronization has been achieved. If the receiver detects an error while the node is in the Ready state then it indicates an 'code violation'.

Modulation

The encoded data is transmitted as a base-band digital signal using the non-return-to-zero (NRZ) method.

Symbol	Function	D-1	Encode abcdei fghj	D0	Alternate abcdei fghj
K.28.0	User-def	-	001111 0100	0	110000 1011
K.28.1 *	FLAG	-	001111 1001	+	110000 0110
K.28.2	User-def	-	001111 0101	+	110000 1010
K.28.3	User-def	-	001111 0011	+	110000 1100
K.28.4	User-def	-	001111 0010	0	110000 1101
K.28.5 *	DIS	-	001111 1010	+	110000 0101
K.28.6	User-def	-	001111 0110	+	110000 1001
K.28.7 *	User-def	-	001111 1000	0	110000 0111
K.23.7	ACK	-	111010 1000	0	000101 0111
K.27.7	RR	-	110110 1000	0	001001 0111
K.29.7	NUL	-	101110 1000	0	010001 0111
K.30.7	User-def	-	011110 1000	0	100001 0111

* - Comma characters (K28.7 must not be contiguous to another K28.7)

Figure 13. Special characters

Data Rate

The serial link provides a choice of two data rates initially:

- **10 MB/s**, which is equivalent to 100 Mbits/s on the line.
- **20 MB/s**, which is equivalent to 200 Mbits/s on the line.

Higher speeds (eg. 40 MB/s) may be defined in the future.

The minimum overhead imposed by the frame format is 5 characters for a FLAG and the control, address and CRC fields. In addition if the link is being operated in full-duplex mode there will be an additional overhead of 4 characters for the ACK and RR responses. Assuming an instantaneous data rate of 10 MB/s these overheads lead to the sustained data rates tabulated in Figure 14.

Frame size	Half-duplex	Full-duplex
32 bytes	8.6 MB/s	2 x 7.8 MB/s
64 bytes	9.3 MB/s	2 x 8.8 MB/s
128 bytes	9.6 MB/s	2 x 9.3 MB/s
256 bytes	9.8 MB/s	2 x 9.7 MB/s

Figure 14. Sustained data rate

These figures assume that there are no transmission errors and that the transmitters and receivers have buffering for at least 2 frames each. The sustained data rate of a 20 MB/s link will be exactly double the values above.

The transmit clock should be accurate to +/- 0.01%.

The receiver must extract a suitable clock from the transitions in the transmitted data. The clock recovery scheme should achieve character synchronization after a maximum of 5 consecutive FLAG's. If an asynchronous sampling technique is used then the design must take metastability into account.

To enhance compatibility between different products some nodes may need to operate at more than one data rate. Such a node should employ the following procedure to negotiate the operating data rate:

1. When a node is in the Disabled or Enabled states and the receiver has not yet acquired character synchronization then the transmitter operates at the highest data rate implemented.
2. In the Disabled and Enabled states the receiver attempts to acquire character synchronization by recognizing 3 consecutive FLAG characters at any of the data rates that it implements.

   Suppose, for example, that the local node can operate at either 10 MB/s or 20 MB/s and the remote node can only operate at 10 MB/s. Then the local node could sample the inbound line at 200 MHz and look for 3 FLAG characters in either 30 consecutive output bits or 30 alternate bits in a stream of 60 consecutive output bits. In this example the latter case occurs indicating that the remote node is transmitting at 10 MB/s.
3. When the local receiver has acquired character synchronization it instructs the local transmitter to operate at the speed of the incoming data. In this example the transmitter will operate at 10 MB/s.
4. Meanwhile the remote receiver will have been sampling the line at 100 MHz. However during the time that the local node was transmitting at 20 MB/s it will not be able to achieve consistent bit synchronization. (Some transitions will appear to fall in the middle of a bit cell.) Also the remote receiver will not find 3 consecutive FLAG characters. (It will only be sampling every other bit and the pattern chosen for the FLAG character avoids aliasing.)

   The remote receiver will only decode 3 consecutive FLAG characters after the local transmitter switches to 10 MB/s. The remote node will then acquire character synchronization.
5. Both nodes are now ready to communicate at 10 MB/s.

This scheme is open-ended in that a number of higher speeds can be defined in the future. (At least in powers of 2.) It also has an advantage over techniques that exchange messages because it does not require all future implementations to negotiate initially at the slowest speed defined (10 MB/s).

Electrical specifications

The electrical configuration of the serial link is shown in Figure 15 on page 34.

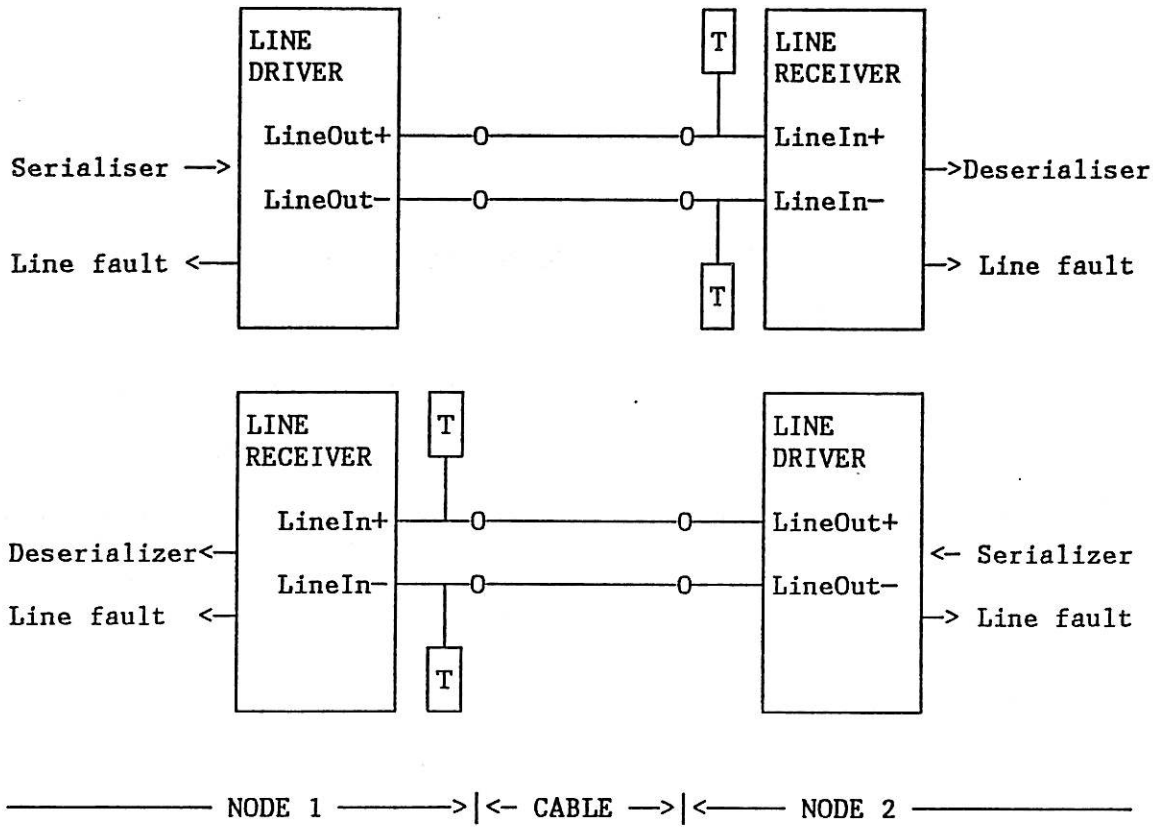


Figure 15. Electrical configuration.

Line Driver

This is an analogue circuit which converts a logic signal from the serializer into a differential current to drive the outbound line. Each driver output is a high-impedance current sink with sufficient compliance to accommodate ground-shift between the two nodes. The nominal sink currents are as follows:

Signal	'0' state	'1' state	Units
LineOut+	10	0	mA
LineOut−	0	10	mA

The line driver must also indicate a 'line fault' when the voltage at either output is less than a threshold. This output indicates the following conditions:

1. Either or both wires of the outbound line is open-circuit.
2. Either or both wires of the outbound line is shorted to ground.
3. The remote node is powered off.

Note that severe external noise may also cause transient line-fault indications.

The driver must be undamaged by an electrostatic discharge of 1500 V to either output. This test should be performed using the standard human-body model when the cable is disconnected.

The operating specifications for the driver are summarized in Figure 16.

Parameter	Minimum	Nominal	Maximum	Units
Pulse width @ 20 MB/s	5			ns
Rise/fall times, 10 - 90%	1		2	ns
Pulse-width distortion			+/- 0.5	ns
On-state output current	- 8	- 10	- 12	mA
Output voltage range	2.0		7.5	V
Line-fault threshold	0.5		1.0	V

Figure 16. Line driver specifications

Line Receiver

This is an analogue circuit that converts the differential voltage on the inbound line into a logic signal for the deserializer. When the line is correctly terminated the nominal input signals are as follows:

Signal	'0' state	'1' state	Units
LineIn+	4.5	5.0	V
LineIn-	5.0	4.5	V

To provide a measure of noise rejection the receiver incorporates hysteresis.

The line receiver must indicate a 'line fault' when the magnitude of the differential voltage at the input is less than a threshold. A line fault should be detected within the specified maximum delay. However the detector must not be so fast that the finite rise and fall times of a normal signal cause false indications.

The line-fault output indicates the following conditions:

1. Both wires of the inbound line are open-circuit.
2. Both wires of the inbound line are shorted together.

3. The remote node is powered off.

Note that severe external noise may also cause transient line-fault indications.

To guard against marginal operation if one of the wires is open-circuit, the receiver should either indicate a line fault or, alternatively, not detect any transitions.

Since the line is terminated by a resistor to the local +5V supply the receiver does not need to accommodate ground shift.

The receiver must be undamaged by an electrostatic discharge of 1500 V to either input. This test should be performed using the standard human-body model when the line termination is connected and the cable is disconnected.

The operating specifications for the receiver are summarized in Figure 17.

Parameter	Minimum	Nominal	Maximum	Units
Input pulse width @ 20 MB/s	4.5			ns
Pulse-width distortion			+/- 0.5	ns
Input differential voltage	+/- 300	+/- 500	+/- 700	mV
Input voltage range	3.9		5.5	V
Total hysteresis	100	200	300	mV
Line-fault threshold	+/- 50		+/- 200	mV

Figure 17. Line receiver specifications

Line termination

Each node must terminate both wires of its inbound line close to the receiver inputs. The termination for each wire consists of a 51 ohms, +/- 5% resistor to + 5.0 V, +/-10%.

We are currently considering changing the termination voltage to 3.3 V to reduce power consumption and facilitate compatibility with future 0.5 micron CMOS technology. However this would reduce the common-mode range of the line driver. It would also affect the DC specifications of the line driver and line receiver.

Ground shift

The using system must guarantee that the maximum ground-shift between nodes is +/- 2.0 V when the link is operating. To avoid circuit damage it must never exceed +/- 3.0 V when the cable is connected.

Inter-connections

This section is preliminary. The contents are requirements only and they are subject to change. The component specifications will be provided in a future update.

Nodes can be inter-connected in one of two ways:

- Via an 'internal' connection within the same physical box. In this case the cables are implementation-dependent, eg. twisted-pair or Flex. However the device connector will be specified to ensure that different devices can be inter-changed.
- Via an 'external' connection between separate boxes. To ensure compatibility the cables and connectors must be as specified in the following sections.

Device connector requirements

The device should provide a header with 5 inline pins on 0.1" centers for the following 5 signals:

- LineOut+ and LineOut-
- LineIn+ and LineIn-
- Logic ground

The requirements for the connectors are as follows:

- **Polarized.**
- **Positive retention.**
- **Durability.** 20 mate and unmate cycles minimum.
- **Low-cost components.** Less than ¢50 per plug or header.
- **Low-cost cable assembly.** For example, insulation displacement or crimping.

Internal connections

The card wiring and internal cables must not cause significant signal degradation due to:

- Resistance or high-frequency attenuation.
- Noise, eg. cross-talk.
- Reflections from stubs or other discontinuities in the transmission line.

External connector requirements

Each node should provide a 6-pin bulk-head receptacle for the following 6 signals:

- LineOut+ and LineOut-
- LineIn+ and LineIn-
- Logic ground.
- +5 V supply. This should provide 200 mA maximum with a regulation of +/- 10% and short-circuit protection. It could be used to power a fiber-optic extender, for example.

Both ends of the cable are terminated with mating 6-pin female plugs. The signal wires cross over in one of the plugs so that 'LineOut+' in one plug is connected to 'LineIn+' in the other, etc.

The requirements for the connectors are summarized below:

- **Continuous RFI shield.** The connectors should have metal shells that ground to the outside of the bulkhead. The shell contact resistance should be less than 5 mOhms. The RFI specifications should be met without using ferrite chokes on the cable.

  The shell should connect before the signal pins to reduce ESD stress on the line driver and receiver.

- **Physically compact.** A MicroChannel[1] card must be capable of accommodating at least 4 receptacles. However the plug must still accept 7 mm diameter cable.

  A modular connector system would provide the most flexibility, particularly for nodes that require many links.

- **Polarized.**
- **Self-latching plug.**
- **Customer set-up.**
- **Durability.** 100 mate and unmate cycles minimum.
- **Low-cost components.** Less than $3 per plug or receptacle.
- **Low-cost cable assembly.** For example, insulation displacement or crimping.

External cable requirements

The cable contains 2 differential pairs. The maximum length of cable allowed is 10 Meters between nodes. The requirements are summarized below:

- **Attenuation.** The attenuation should be less than 0.15 dB per Meter at 100 MHz. It will probably be necessary to use Twinax but twisted pair will be considered if it can meet the requirements.
- **Cross-talk.** Each pair should have an individual foil shield and a drain wire. The drain wires will be connected to logic ground at the driver end only.
- **RFI.** To meet the RFI specification the cable should have an overall braid shield. This will be connected to frame ground via the connector shells at both ends of the cable.
- **Characteristic impedance.** 100 ohms, differential.

  We are considering a change to 150-ohm cable to reduce power dissipation. If adopted this would affect the termination resistor and the output current of the line driver.

- **DC resistance.** Less than 0.3 ohms per Meter.
- **Diameter.** The overall diameter should be less than 7 mm.
- **Flexibility.** The cable should allow a 75 mm bend radius with a life of 100 flexures at 180 degrees.
- **Low cost.** Less than $2 per Meter.
- **UL approved.**

Environment

The link nodes should operate over the temperature range from 10 degrees to 40 degrees Centigrade.

---

1 Trademark of IBM Corporation

Radiated RFI

The cable and connectors that are specified for 'external' connections should meet the following standards for radiated RFI with a margin of at least 10 dB:

- FCC Class A
- VDE Class A
- VCCI Class A
- CISPR Class A

The specified margin allows simultaneous operation of at least 12 links with an allowance for RFI from other components in the system.

Error rates

The error rate that is obtained depends on several factors that are beyond architectural control, eg.,

- The detailed design of the receiver, particularly the clock recovery function.
- The noise that is experienced by an 'external' connection.

Experience with a similar link in the IBM[2] 9333 indicates that a raw error rate of less than 1 in 10**13 characters is achievable.

The CRC will detect all error bursts with a span of 16 bits or less. The probability of the CRC not detecting a burst greater than 16 bits is 1 in 65536.

In addition the 8B/10B code has approximately 75% probability of detecting an error in each character that is corrupted. It will detect all bursts that cause the receiver to saturate at logic '0' or logic '1' for more than 5 bit periods.

Thus the error detection capabilities of the CRC and the 8B/10B code are largely complementary. The overall undetected error rate of the link should be less than 1 in 10**18 characters.

[2] Trademark of IBM Corporation

Appendix A. Future extension

This section outlines a proposed extension to SSA that uses distributed frame switches to allow up to 16 nodes to be connected in either a daisy-chain or a loop.

The daisy-chain configuration is particularly attractive for connecting I/O devices to a personal computer, eg. via a MicroChannel adapter. It reduces the attachment cost per device and it avoids wiring congestion at the adapter.

Use of the additional connectivity is optional. If it is not appropriate in a particular environment then dedicated point-to-point links can still be used.

If this proposal is adopted then it will be integrated into a later version of the SSA-0 specification. The facilities for circuit switching would probably be deleted since the distributed frame switches provide a more powerful mechanism.

Frame switch

SSA-1 already provides for a dual-ported disk drive with two serial links. The second link allows the disk drive to be accessed via a backup path in the event of a failure in the primary attachment path. In conjunction with disk arrays this allows configurations with no single point of failure.

The basic idea in the current proposal is to add a 3-port frame-switch between the two link interfaces and the device proper, as shown in Figure 18.

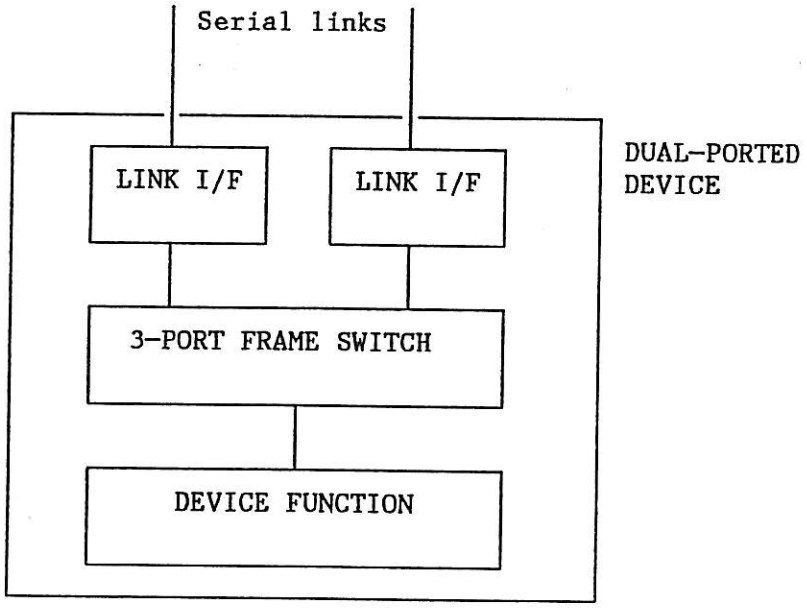


Figure 18. Dual-ported device with a frame switch

Depending on the address field, the frame switch routes an inbound frame from either link to the device or to the outbound line of the other link. When the device wants to send a frame it instructs the switch to transmit it on a specified link.

The frame switch and second link would be optional. If a node does not implement them then it can only be used with a dedicated link or possibly at the end of a daisy chain. (See below.) In practice, all disk drives would probably be dual-ported, either to provide fault tolerance in large systems or to increase fan-out in personal computers.

Daisy-chain configuration

The frame switch allows a number of devices to be connected to a single adapter port in a 'daisy chain', as shown in Figure 19. The devices provide a distributed routing mechanism that supports peer-to-peer communication between any pair of nodes. The physical links are still point-to-point and they operate independently using the protocol and error recovery procedures already defined by SSA-0. (However Total Reset needs to be redefined as a global function.)

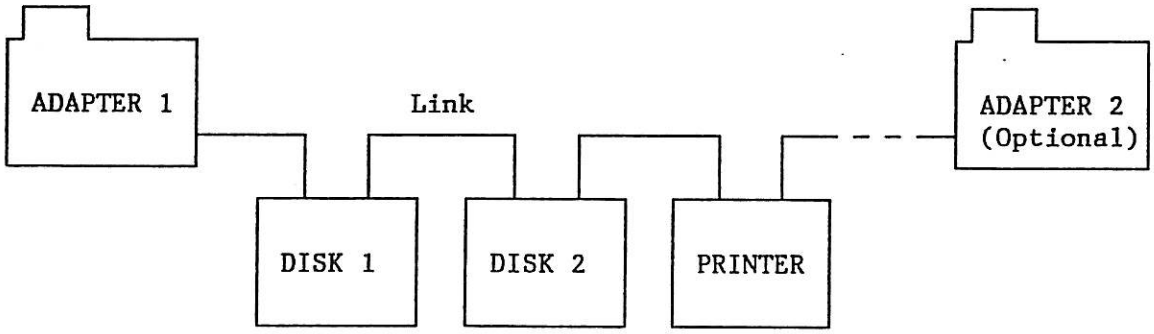


Figure 19. Daisy-chain configuration

Optionally, a second adapter can be connected at the far end of the chain. This provides a backup path in the event that the primary adapter or computer fails.

Since the link segments are independent and they support full-duplex communication several data transfers can occur concurrently at the full link bandwidth. ie. 10 MB/s or 20 MB/s. For example, either of the following scenarios could occur:

- DISK1 to/from ADAPTER1 and DISK2 to/from ADAPTER2. (Different links.)
- DISK2 to ADAPTER1 and DISK1 to ADAPTER2. (Inbound & outbound lines.)

This spatial reuse can provide a considerable increase in the effective bandwidth compared to a bus or token ring.

At lower speeds frame multiplexing allows a large number of simultaneous transfers between unrestricted nodes. Of course the total traffic on either line of any link is limited to the link speed. (10 or 20 MB/s.)

Loop configuration

The architecture above can be generalized if the adapter provides dual ports and a frame switch in the same way as the devices. This permits more than 2 adapters per daisy chain and it also allows a closed loop, as shown in Figure 20 on page 43.

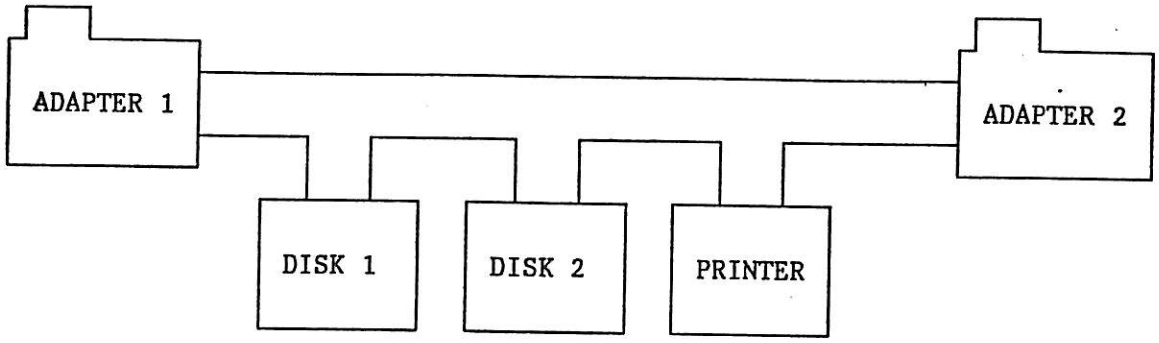


Figure 20. Loop configuration

The loop requires a 'hop count' in each frame to prevent frames circulating indefinitely, eg. if there is an error in the frame address.

The main advantage of the loop is improved RAS. Any single node or link can fail and all of the remaining nodes can still communicate. A node can also be inserted into the loop or removed from the loop dynamically without preventing communication between the other devices.

In normal operation the loop doubles the bandwidth available to any particular node. For example, the adapter has two alternative routes to each device. For optimum efficiency it should normally use the shorter path.

Dual-ported adapters and controllers might be restricted to high-end applications which need the loop to improve RAS.

Frame addressing

For normal transfers the address field in each frame is divided into a 4-bit 'identifier' to select the destination node and a 4-bit 'channel' number within that node. Thus there can be up to 16 nodes in all.

Data transfers are initiated by the exchange of message frames between the source node and the destination nodes. These messages specify the value of the address field to be used in the following data frames.

One channel is reserved for inbound messages and the other 15 channels can be dynamically allocated to inbound data transfers. Thus the link allows up to 15 concurrent inbound data transfers per node. In practice a device may support only 1 channel but an adapter should support several.

Some applications need to send a frame to more than 1 destination node. These 'multi-cast' frames can be identified by a bit in the frame control field. In a multi-cast frame the address field indicates the source node and channel rather than the destination.

Each node on a given daisy-chain or loop needs to have a unique node identifier for frame routing. Since manual switches are prone to error the identifier should be assigned automatically by a configuration process.

Switch functions

To reduce latency the switches should preferably use 'worm-hole' routing. Provided that the out-bound line is available each switch forwards an inbound frame character-by-character as soon as it has received the address field. This reduces the minimum latency to 3 characters per switch.

The switches need a 'fairness' algorithm to guarantee that all nodes receive an equitable share of the bandwidth. At the same time the algorithm must permit spatial reuse. This can be achieved by passing a token from node to node, eg. a special character. Each time a node receives the token it is allowed to originate frames up to a predefined quota. A node forwards the token when it has consumed its quota or if it does not wish to originate further frames.

Some real-time applications require 'synchronous' transfers, eg. audio and video. This can be facilitated by adding a bit to the frame control field so that the switches give these frames priority over asynchronous traffic. However the communicating nodes must still provide sufficient buffering to allow for a delay of 1 frame per switch traversed.

To avoid blocking communication between other nodes a node should never withhold RR for long periods.

Glossary

Application. A process that is communicating via the link

Character. A sequence of 10 encoded bits that represents a data byte or a protocol function

CMOS. Complementary Metal Oxide Semiconductor

CRC. Cyclic Redundancy Check

Destination. The node that receives a particular frame

DMA. Direct Memory Access

ERP. Error Recovery Procedure

Field. A group of related data characters in a frame, eg. the CRC field

Frame. A sequence of 4 or more data characters surrounded by FLAG characters

FSM. Finite-State Machine

FSN. Frame Sequence Number

Line. A physical connection between a transmitter and a receiver

LSI. Large Scale Integration

Node. One of the two ends of a link

NRZ. Non-Return-to-Zero

POR. Power-On Reset

POST. Power-On Self-Test

RAS. Reliability, Availability and Serviceability

Receiver, Rx. The logic that decodes the signal on the inbound line

Response. A pair of ACK or RR characters that is sent in reply to a received frame

RFI. Radio-Frequency Interference

RSN. Receive Sequence Number

SCSI. Small Computer Systems Interface

Source. The node that originates a particular frame

SSA. Serial Storage Architecture

Transmitter, Tx. The logic that drives the outbound line

TSN. Transmit Sequence Number

References

1. A. X. Widmer, P. A. Franaszek, "A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code", IBM Journal of Research and Development, Volume 27 number 5, September 1983.