Serial Storage Architecture
A Technology Overview

Version 3.0

Send comments to:
David Deming
Solution Technology
P.O. Box 104
Boulder Creek, CA 95006
Ph: 408/338-4285 Fx:408/338-4374
email: ddeming@scruznet.com

Introduction

An Unparalleled Connection

A revolution is occurring in the world of peripheral interconnections bringing with it higher levels of availability, fault tolerance, performance, and connectivity -- all at a lower cost.

Imagine a world where a single I/O interface can effectively address the needs of the entire spectrum of computers, from personal computers to super computers. Multiple interfaces are a thing of the past ... since today Serial Storage Architecture (SSA) provides an unparalleled connection to a wide array of storage and peripheral products. With visionary insight, SSA was designed to meet today's needs while supporting tomorrow's requirements.

Serial Storage Architecture

SSA is a powerful high performance serial interface. It has been specifically designed to connect disk drives, tape drives, CD-ROMs, optical drives, printers, scanners and other peripherals to workstations, servers, and storage subsystems. It is the only serial interface that was designed from the outset to meet the requirements of a wide range of I/O devices. SSA offers superior performance because its fundamental building block is a single port capable of carrying on two 20 MB/sec conversations at once -- one inbound and one outbound. A basic SSA connection is a dual-port node capable of carrying on four simultaneous conversations, for a total bandwidth of 80 MB/sec.

SSA's dual-port, full-duplex architecture allows peripherals to be connected in configurations with no single point of failure. Because multiple paths are inherent in the design, increased fault tolerance is far easier to implement. SSA provides "hot plugging" and automatic configuration when nodes are added or deleted. For configuration flexibility, SSA nodes can be up to 20 meters apart using low-cost shielded twisted pair. SSA will offer many advantages over future parallel interfaces due to the simple fact that cabling and connector costs can be reduced dramatically. SSA enhancements include speed, distance, simplicity and reliability. Many leading computer and peripheral companies have adopted SSA as their future serial architecture choice.

SSA has the flexibility to be implemented with multiple topologies including string, loop, and switch configurations. In a typical single loop example, 128 nodes (peripherals or hosts) can be supported. In a complex switched configuration with multiple loops, the theoretical maximum number of nodes would be just over two million.

At the same time that performance, availability, and reliability continue to improve, computing requirements are also becoming more complex. SSA's flexibility not only supports this increasing complexity, but actually permits new approaches not available with other interconnection technologies.

SSA is being developed by the American National Standards Institute (ANSI) X3T10.1. The standards being developed describes the protocol and electrical specifications of the interface. A listing of these documents are included in the Bibliography at the end of this document.

The main objective of this document is to provide a basic technical education on the SSA interface. It also addresses the advantages of SSA and why it is becoming one of the most popular serial options. Most of the information contained within this document was adapted from training documentation originally developed by Solution Technology.

Brief SSA History

Like all computer interfaces the Serial Storage Architecture Interface started out as just another proprietary computer interface. The first SSA interface was developed by IBM and was called the 9333 I/O channel. This interface was designed for high performance storage applications and today there are thousands of units installed in the field. In 1991, this technology was made available to the computer industry as one of the serial storage options for SCSI-3.

SSA started out under control of SSA-UIG (Serial Storage Architecture - User Industry Group) which was made up of many leading computer companies. Since 1994, SSA standardization and documentation has been under control of X3T10.1. This ANSI committee is a daughter committee to X3T10, which is the SCSI-3 committee. This has literally opened the architecture to any interested company and makes the interface an international serial option.

When the architecture was taken over by the X3T10.1 ANSI subcommittee many enhancements needed to be made which dramatically changed the interface. Some of these changes included the following:

- 20 MB/sec Serial Links
- Interoperability between different host systems
- The ability to operate under any Protocol, i.e. SCSI, IP, HIPPI, ATM, etc...
- Enhanced electrical specification
- Advanced error handling and warning system

Why SSA?

- **Easy Migration from SCSI**

A serial alternative that allows current and future SCSI mapping. By allowing SCSI mapping, the integrator can re-use Device Drivers and target firmware. This makes migration to the SSA interface much easier because it retains much of the SCSI logical model. Estimates could be as low as a 15% code rewrite to upgrade the SCSI firmware on a disk drive from SCSI to SSA. If you don't like SCSI mapping, choose your own protocol. SSA has the capability to map any protocol across it's physical link.

- **Addressing, Queuing, Commands, Status, and Sense Data**

SSA maintains the **address** scheme as defined in the SCSI standard where as there are initiators, targets, and logical units. Therefore you don't need to learn new terms and functions. The SCSI **queuing** model is also retained by using a 2-byte Tag for each I/0 process with Simple, Ordered and Head of Queues. SSA can use the same 6, 10 and 12-byte **Commands** and return the same **Status** codes as defined by SCSI. This means that you can issue the same Read command and determine command completion as defined by current Device Driver implementations. If an error occurs (i.e. a Check Condition status code), the Request Sense command can be issued and the **Sense Data** returned is as specified in the SCSI standard. This makes error handling for the host system software and target firmware identical to current SCSI implementations.

SSA adds a layer of software/firmware onto a product which is used to manage the link protocol, link/system configuration, frame buffers and handle link errors.

- **Advantages Over Existing SCSI Parallel Interface**

  - **Compact cables** and **connectors** for small form-factor devices:
    When SSA cabling and connectors are compared to the 50 or 68 wire parallel SCSI interface, the costs savings alone make SSA appealing. Internal connections are made with six conductor cable and connectors which consists of leaf style multiport headers on .050 inch spacing and 6 circuit insulation displacement connectors. External connectors are 9-way micro-mini 'D' connectors and shielded cables. SSA is also a point to point architecture with up to 20 meters between each node with low cost copper cables or up to 680 meters between each node if fiber optic cables are used.

  - **Excellent performance:**
    Due to the fact that there are no bus phases like Arbitration, Selection, and Reselection, the protocol has low overhead typically around 6%. Functions like arbitrating only adds overhead to a transfer. This makes SSA 94% efficient, which means that 94% of the data on the interface is customer data.

  - **Greater connectivity options:**
    This is accomplished via strings, loops, and switches. No longer are you limited to a daisy chain of devices. SSA allows up to 129 nodes in a string, 128 nodes in a loop, and a virtually unlimited node count if switches are used. Compare this to only 8 or 16 devices for parallel SCSI.

  - **Enhanced reliability:**
    This is achieved by having CRC on each transfer and a link layer recovery mechanism. CRC translates to a raw error rate of less than 1 in 10^{13} bytes. The link layer recovery mechanism that is designed into the interface means link errors are handled by each node and not by the system software. If errors are not recovered by the link layer a network administrator handles and coordinates error recovery.

- **Higher availability, fault-tolerance, and hot plugging:**
  The network can be configured into loops and switches which translates to no single point of failure. Point to point links also facilitate fault isolation and permits hot plugging. Nodes are addressed by a Path address in each frame. This means no hardware switches need to be set. Higher availability is accomplished via alternate paths and spatial reuse (this is covered later).

Other Features and Advantages of SSA

- **Cost-sensitive**

SSA has been optimized for **cost-sensitive** applications such as disk drives. The hardware can be fully integrated in 3.3 V CMOS technology. This feature combined with the fact that cabling and connectors costs are reduced truly makes SSA a cost-sensitive interface.

- **Frame routing is done in hardware**

This means that a node's firmware will not have to analyze a frame before the routing function takes place. SSA is also a forward and store interface which means that frames are routed after the first couple of bytes have been analyzed.

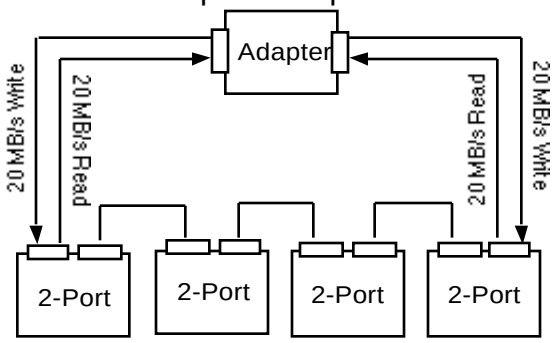
- **Disk spindle synchronization integrated**

Spindlesync is integrated into SSA protocol (No additional cables are needed). It is multiplexed onto the SSA link using a special character and is beneficial to disk array applications.

- **High data transfer rates are supported**

Initial peak data rate is 20 MB/sec in each direction per port. This will be known as the link speed, the total traffic on either line. This will equate to a maximum port speed of 40 MB/sec total bandwidth in full-duplex mode. If another port is added to a node and a cyclic loop is used, each node could have a theoretical 80 MB/sec bandwidth (minus any overhead). Note: In order to achieve the theoretical 80 MB/sec data transfer speeds, spatial reuse and full-duplex transfers must occur. These features may require advanced hardware and programming behind the SSA interface. The architecture is open-ended so that higher speeds (i.e. 40 MB/sec and 100 MB/sec per direction) will be supported. Multiple concurrent transfers with each node and optional out-of-order transfers, e.g. for RAID-5 controllers, are supported.

80 MB/sec bandwidth per dual-port node if running full-duplex.



- **Standardization**

Standardization of SSA document has been submitted to the Accredited Technical Committee X3T10.1 for review and approval as an American National Standard.

SSA is an acronym for serial storage architecture. SSA is a high performance low cost serial interface for connecting storage devices, storage subsystems, servers and workstations. Although initially implemented for storage subsystems, SSA has many networking and other features which enable innovative approaches to connectivity in many applications.

A Frame Multiplexed Interface

A frame is the way that the SSA interface transmits information. It has an exact format and will be covered later. Frame multiplexing uses special characters to indicate acceptance or completion of information (a frame) transmitted between devices. The multiplexing capability means that the system of telegraphy allows two or more messages to be sent simultaneously in either direction over the same cable.

An Interface with a Standardized Configuration and System Architecture

Network configuration is built into the standard and configuration is automatic. The network can be dynamically changed. The standard specifies a Master node which performs the configuration process of the network. The Master is responsible for informing other nodes of their location in the network by issuing special configuration messages. Other primary initiators in the network also perform a configuration process by which a Configuration table is created. This table is a list of all the nodes, i.e. devices, with pointers to represent the physical links. System architecture is specified as stated above, i.e. Master Nodes, primary initiators, targets, and standard configuration procedures. As mentioned early, there are no address switches to set.

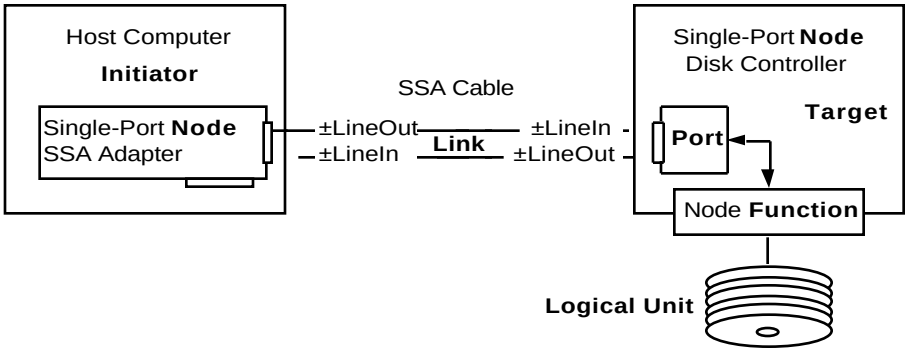
An Interface with Built-In Fault Detection

Each device has Power On Self Test (POST) capability in which the device performs a local wrap test. After POST, each port (the basic component of an SSA device) has a specific "beginning communication" procedure. Once the procedure is completed by the port, that port can now transmit frames. CRC and special protocol characters are used to maintain bus integrity. The point to point links facilitate fault isolation and permit hot plugging for concurrent maintenance.

Each device monitors its neighbor. If a neighboring device stops transmissions, then the detecting device performs a Link Error Recovery Procedures (Link ERP). Link ERP is a standardized procedure by which a device attempts to re-establish a connection. If the Link ERP is unsuccessful then the Master is informed via an "Asynchronous alert". These alerts are used to report events that are outside the scope of the Link ERP. SSA defines guidelines for what a network should do with Link ERP's, Asynchronous alerts, targets not responding, addressed ports not operational, invalid messages, new links connected and blocked communications. This takes the guess work out of how to construct a fully compatible fault-tolerant network.

Basic SSA Definitions

The most basic network consists of a single-port host adapter connected to a single-port peripheral. The serial connection consists of six wires of which four are used to transmit frames of information. The four lines consist of a plus/minus LineOut and a plus/minus LineIn.



A **Link** (to unite, connect, or join) is a dedicated connection between two single-port nodes. An interesting note about a link is that when a link is idle, i.e. no frames are being transferred, synchronizing characters are sent between the single-ports. This keeps the link synchronized and allows the network to instantly detect a line fault (link failure).

A **Port** (a gateway) consists of the hardware and firmware to support one end of a link. Most of a ports functions are contained within currently available silicon. These chips have SERDES Serializer/Deserializer, ABUS for "CFE Local Bus operation" or "Disk Manager Operations" and a BBUS for address, data and control signals which could be used for communication with an external processor. Special protocol registers which are used to properly sequence the interface and frame buffers are present for Inbound and Outbound frames. Some silicon implementations have the DMA controller and processor logic fully integrated.

A **Node** (a point of concentration, a central point) is a system, controller or device with one or more serial links. There are three types of nodes, single-port , dual-port, or switch port. Each node has a **function** which is the specific responsibility, task, or use of the node, i.e. the reason for the nodes existence.

Initiator, the mode of the node within a device that determines what task needs to be executed and which target will perform the desired task. It creates frames of information to be sent across the SSA link and confirms that the target performed the task assigned to it. In SSA there must be one initiator in the network that is designated as the Master.

Target, the mode of the node within a device that acknowledges the receipt of frames of information. Targets inform the initiator that it is ready to receive a frame of information and it may have up to one hundred twenty eight logical units (LUN) attached to it.

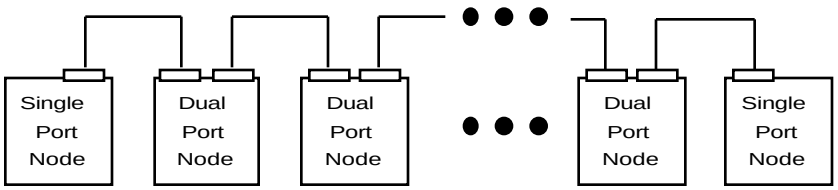
Logical units are a physical or virtual peripheral device addressable through a target (peripheral controller). They are the basic addressable unit on a target and can be a portion of a peripheral device (i.e. a partition).

SSA Topologies

The different configurations of an SSA network can be seen below. The progression will be from the most basic configuration to the most advanced. There are three basic topologies that exist in SSA and these topologies can exist individually or can be combined to form a complex network.

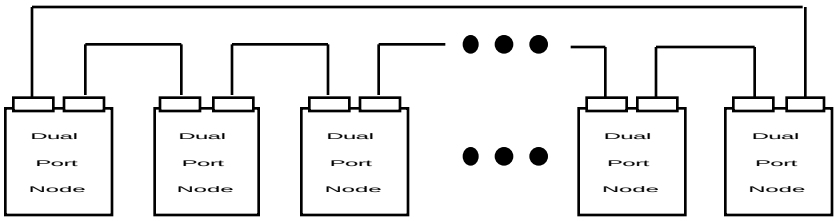
The string topology below is the most simple of topologies. This topology is not popular because it has single points of failure.

String Topology



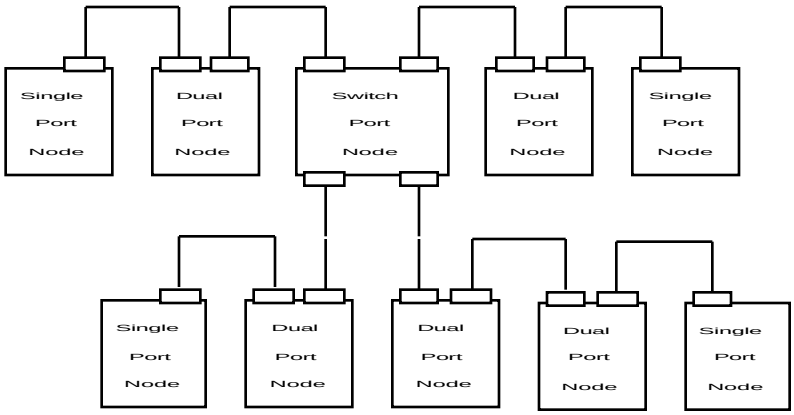
The loop topology below is the most popular of topologies because it has alternate paths to each node in the network and it eliminates a single point of failure.

Loop Topology



The switch topology below is the most complicated topology because it takes additional hardware to implement. Fortunately the currently available silicon has switch capability fully integrated and to add this functionally to a network is an exercise in hardware implementation. This configuration is the most fault tolerant and allows a virtually unlimited network configuration.

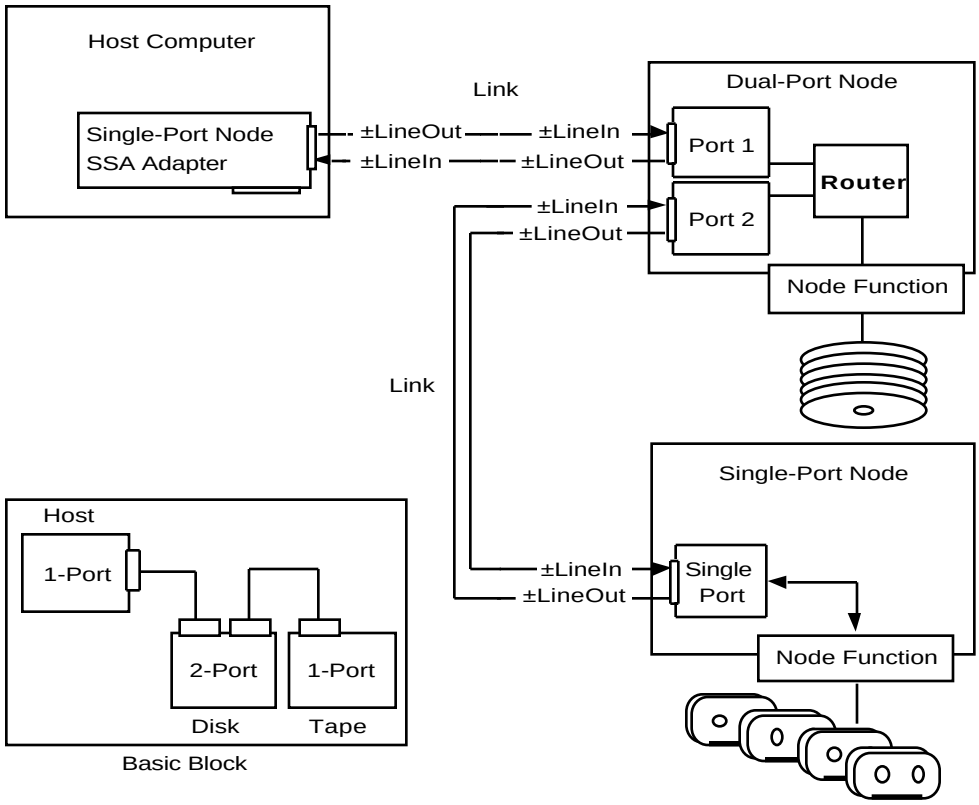
Switch Topology



String Characteristics

A string is a linear network of two or more nodes, as shown below.

The extreme nodes at either end of a string can be single-port nodes, dual-port nodes with one port not operational or a switch. The intermediate nodes (if any) are always dual-port nodes.
Each link operates independently and manages its own flow control and error recovery. An error in one link does not affect other links, except for the possible need to abort a frame that is already being forwarded. The maximum number of nodes in a string is 129, including the end nodes.



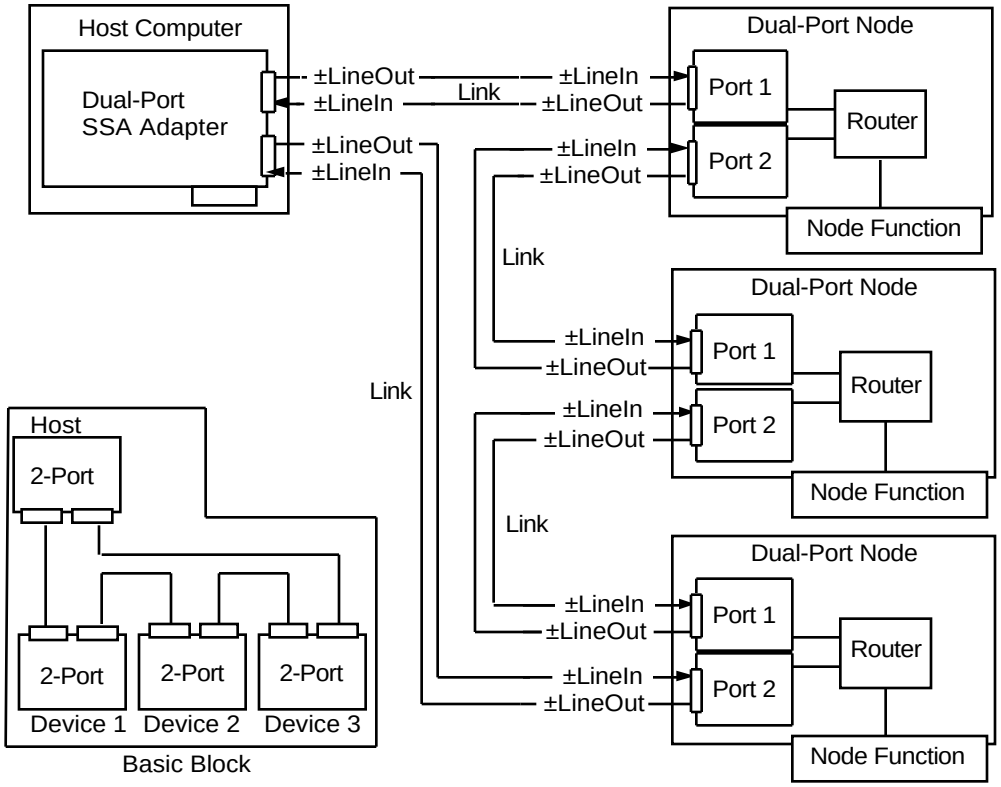
Router(the function of routing, i.e. to send by a certain route):
Depending on the address field of the inbound frame to the node, the router forwards the frame to the node function or to the outbound line of another port. When the node wants to originate a frame it instructs the router to transmit it on a specified port. Note that all **message** and **data** frames relating to a particular **command** use the same port.

Loop Characteristics

A loop is a cyclic network containing **only** dual-port nodes, as seen below.

A loop provides better availability than a string because any single-port node can fail without blocking communication between any pair of the remaining nodes. A node can also be inserted into the loop or removed from the loop dynamically without preventing communication between the other nodes. This will require reconfiguration of the network but the loop will remain functional. The maximum number of nodes is limited to 128. A loop may not contain a switch, as this would break up the continuity of the loop.

An important property of strings and loops is *spatial reuse*. Each link functions independently of the other links so that several transfers can occur concurrently at the full bandwidth provided that each uses different links. This allows much higher through-put than a token ring or bus. Translation: A port can be sending a frame of information out its *LineOut* while the port is receiving a frame of information into its *LineIn*. It also means that if the node has two ports this can simultaneously occur on each port.

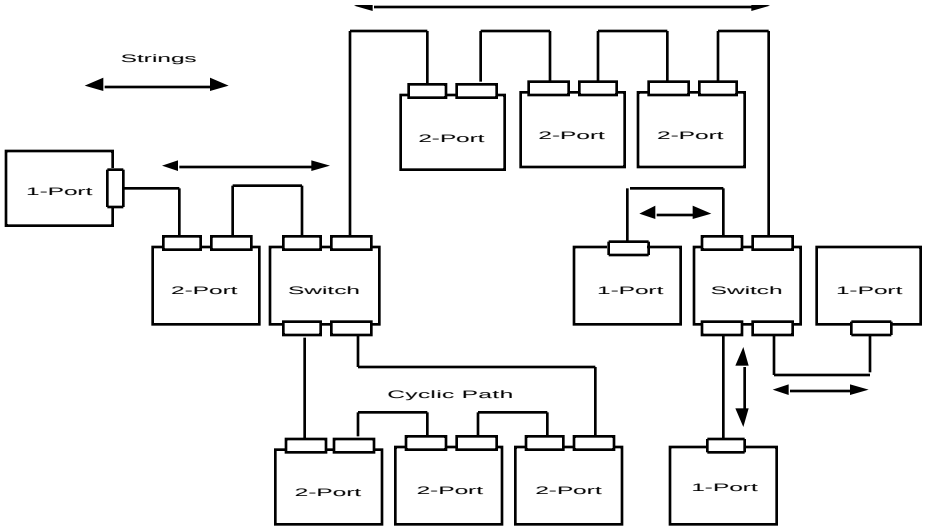


Switch Characteristics

Switches allow a number of strings to be connected to achieve an almost unlimited number of nodes.

Allows alternative paths to be provided to achieve fault-tolerance. The figure below shows a complex network including 2 switches, 5 strings and a cyclic path (if a cyclic path includes a switch it is not a loop).

The maximum number of ports that can be addressed in a switch is 96. There is no standard method on how port addresses are determined on a switch. This is usually a function of the hardware. There will always be an even number of ports (from 4 - 96) and an internal router in every switch.



Summary of Topologies

An SSA interface may consist of one or more of the following:

Strings: A linear network with up to 129 nodes.

Loops: A cyclic network with up to 128 dual-port nodes.

Switches: A frame switch with up to 96 ports. Switches can be inter-connected and combined with strings to allow an almost unlimited choice of topologies.

How the Protocol Works

In this section the basic concepts regarding the SSA protocol are covered. Like any interface the protocol holds the key to understanding just how information is transferred from one device to another. There are two main components that hold the key to understanding SSA protocol:

1. The **frame**
2. Special characters

SSA Transfer Characteristics

In a serial interface there is only one signal line so Data and Control information share the same bus. When information is transferred across the cable it is sent as a stream of bits. When these bits are transmitted they must be self clocking and the synchronous clocking restricts the bit patterns. It is also undesirable to have long sequences of consecutive zeros or ones. What if the data being transferred is supposed to be all zeros or ones? In order to accomplish a successful serial transfer an encoding/decoding algorithm is required to convert the arbitrary data (bit stream) into patterns suitable for transmission. These arbitrary patterns are determined by the use of an **8B/10B** code. The 8B/10B encoding/decoding mechanism, was developed by IBM and is used in many serial interfaces including Fibre Channel.

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. This is all done in hardware and the important information that needs to be understood is that there are two basic types of characters, 'data' characters and 'special' protocol characters.

Character Classifications

Data and special protocol characters are encoded into 10-bits of information for transmission on the physical medium (i.e. the cable). These 10-bit characters are translated into 8-bits, hence the name 8B/10B. This also means that the serial link transfers information at 200 Mbits/sec (200 MBaud). The valid characters are classified as follows:

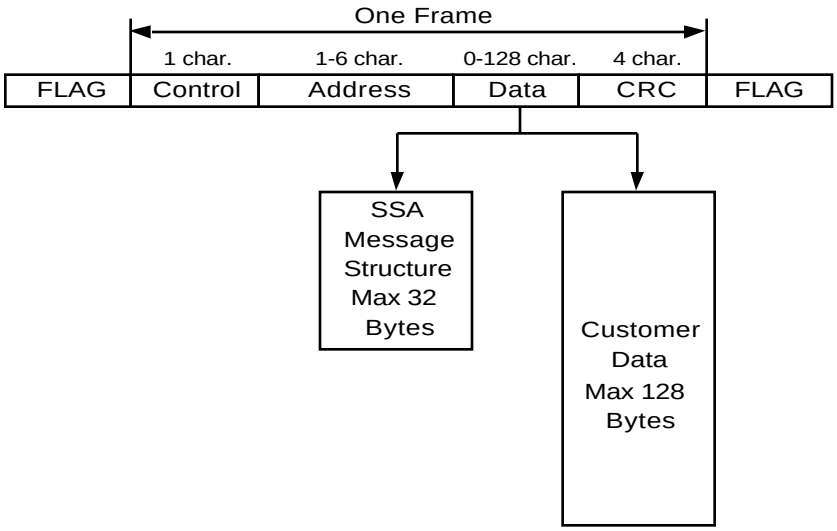
Function - Character Type	Quantity	Description
Protocol - Special	8	These characters are used by the Link as delimiters, to provide flow control, keep the Link synchronized, and other various protocol functions.
User Defined - Special	3	These characters are available to the upper-level protocol for special functions. Currently the only function is the transmission of the SYNC character.
DATA - Data	256	These characters represent an actual byte of Data and are used to make up all of a frames contents, i.e. Control, Address, etc...

Frame Characteristics

Frames provide the unit of pacing and the pacing ensures the transmitting port does not overrun the buffer in the receiving port. Frames are used to transfer all information across the serial link and there are three types of frames:

Frame Type	Description
Application	Frames used by the transport layer for data transfer and to transport any SSA Message Structure (SMS) sent by the upper level protocol (ULP). ULP SMSs are used for commands, data, status and vendor specific information.
Privileged	Frames used by the transport layer for configuration and error recovery. These SMSs are used by the lower level protocol (i.e. the physical interface).
Control	Frames used by the transport layer that are only used for node or link Resets. Node resets are Total and Absolute and the link reset is a "Link Reset".

If the frame is an **Application** or **Privileged** frame, the frame typically consists of a sequence of at least seven 'data' characters delimited at each end by FLAG characters as shown below. Note that the FLAG characters are 'special' characters and are encoded differently than the 'data' characters that make up the frame. The 'data' characters include the Control, Address, actual Data and CRC.



The frame can be made up of four fields. The **control** field, which is always one byte, determines the frame type. The **address** field which can be from two to six bytes is used to route the frame to the receiving node. The **data** field which is optional and can be from 0 to 128 bytes. This field carries the payload which consists of the SSA Message Structure (SMS) or actual customer data. The frame ends with a **CRC** field which is always four bytes.

If the frame is a **Control** frame, the frame consists of a sequence of at least six 'data' characters delimited at each end by FLAG characters (see below).

	1 char.	1-4 char.	4 char.	
FLAG	Control	Address	CRC	FLAG

The frame is made up of three fields which perform the same functions as described above. Seeing how these frames are only used for resets there is no need for a Data field. The type of reset being sent is decoded in the Control field. Resetting the node consists of Total and Absolute resets and there is a Link Reset that is used during the Link ERP.

Special Character Characteristics

Special characters are used to perform many functions. Seeing how the interface is serial, special characters are required for the link protocol to function. The special characters are decoded in the hardware and cause special bits in hardware registers to be set or cleared. These characters can be interleaved into actual frames because the hardware can pluck them out of the data characters. By interleaving special characters into the frame, overhead can be reduced and throughput increased. Below is a complete list of special characters along with a brief description.

Special Character	Description
FLAG	This character is used to delimit frames and is sent when the link is idle.
DIS	This character is used to synchronize the serial link when the port is in the disabled state. The disabled state is a special state of a port that is used when the port has just come out of its Power On Self Test (POST) procedure and the port is on its way to becoming ready for frame transmission. The disabled state is also used to synchronize the Link Error Recovery Procedures of two ports when a link error occurred.
ACK	This character is used to acknowledge the receipt of valid frames. These protocol characters are used in pairs, i.e. ACK-ACK, to protect the ACK response being lost or corrupted by transmission errors.
RR	This character is used to notify a local port that a remote port is ready to receive a frame. These protocol characters are used in pairs, i.e. RR-RR, to protect the RR response being lost or corrupted by transmission errors.
ABORT	This character is used to abort a frame once it has started to transmit the frame. The ABORT character must be preceded by a data character and followed by a FLAG character to action an abort function.
SAT SAT'	These characters are used to implement a fairness algorithm. These algorithms allow nodes to share the link bandwidth equitably when there is contention for the use of a particular link. The SAT' translates to SAT reflect which is the reflection of a SAT character when it reaches the end of a string.
NUL	This character is used to pad frames when a port has started transmission of a frame but the data for the frame isn't ready.

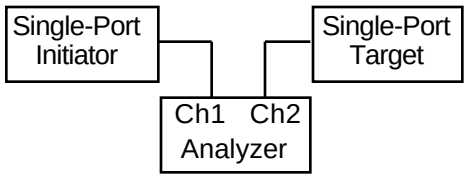
Frame Level Protocol Characteristics

Like any system, each I/O process could have up to three pieces of information to transfer between devices before the process is completed. These components are the:

1. **Command**: which tells the Target what operation to perform, i.e. READ, WRITE.
2. **Data** (optional): which is the actual data bytes associated with command.
3. **Status**: which tells initiator what the target completion status is.

In SSA, this is true if the frame was an **application** frame and the **SCSI_command** SMS was issued by an Initiator. Application SMSs are used by the upper level protocol to read and write data to a target's logical unit. Because the interface is serial, the initiator simply sends a SCSI_command SMS to the Target. Once the Target receives the SMS it decodes it and may return some data via a data frame. After the Target completes the command it sends a SCSI_status SMS to the Initiator. An analyzer display may look like the following:

```
Frm Ch Path     Type Decode                    sssss.mmm_uuu_nnn
  1  1 00       SCMD SCSI_command                          7 450
  1  2 00       DATA                                       7 200
  2  2 00       STAT SCSI_status                           7 450
```



Block diagram of analyzer connection.

To analyze a serial link, an analyzer will have to sit between the two nodes and capture the frames as they go whizzing by from one node to another.

Some SMSs only require a **response** that contains a return code. These SMSs are used at the transport layer (i.e. Privileged SMSs) which are mainly used for configuration and error recovery. The components of this type of I/O process include:

1. A **Privileged** SMS which consists of one of the following: Configure_port, Async_alert, Master_alert, Query_port, Query_node, Query_protocol, and Quiesce.
2. A **response** SMS which can come in the form of the **Response** message or a complementary SMS, i.e. Async_reply, Query_port_reply, Query_node_reply and Query_protocol_reply.

Below is an analyzer display of what a frame level protocol sequence might look like of transport level SMSs.

```
Frm Ch Path     Type Decode                    sssss.mmm_uuu_nnn
  1  1 00       QNOD Query_node                            7 150
  1  2 00       QNRP Query_node_reply                     94 700
  2  1 00       CNFG Configure_port                67 574 848
  2  2 00       RSPS Successful                       208 150
```

Parallel (SCSI-2 or SCSI-3) and Serial (SSA) Differences

Parallel - SCSI-2	Serial - SSA
• only two devices can communicate at one time	• multiple devices can use link(s) providing there is enough buffer space and bandwidth
• bus phases	• no phases but plenty of states and modes; no arbitration, selection or reselection phases
• connects, disconnects and reconnects	• no equivalent concepts. Messages are sent and FLAG determines end of frame. Full multiplexing capabilities, multiple I/O processes can be started simultaneously before completion received.
status	
• commands	• messages
• messages	• messages
• status	• another message
• interlocked interface	• frame multiplexed interface
• bus signals consist of control and data lines	• control and data share same line
• 50 to 68 pin connectors and cables	• six internal and nine external wires
• untagged and tagged queuing	• only tagged queuing, untagged queuing must be emulated
• Contingent Allegiance	• Auto Contingent Allegiance Condition

Protocol Comparison

Below is a comparison between the bus phases of the parallel interface verses the frames of the serial interface. An example of a single Request Sense command is used.

Bus Phases	Frames
Bus Free	
Arbitration Start	
Arb_win	
Selection Start	
Selection Complete	
Message Out	
Command Out Request_sense	SCMD Request Sense
Data In	DATA Sense Data Bytes
Status In	STAT Good Status
Message In	
Bus Free	

SSA has simplified the protocol with no Arbitration, Selection, Message or Bus Free phases. These bus phases add substantial overhead to a transfer and the SSA architecture virtually eliminates this type of overhead. However, for each SCSI I/O process the same results must occur between the two interfaces in that, each I/O process could have three main components:

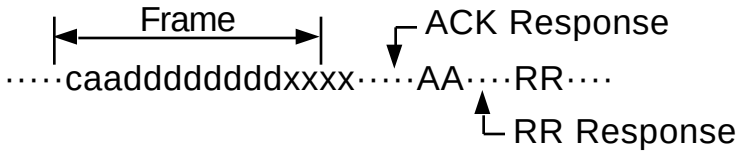
1. the command
2. the data (optional)
3. the status

Character Level Protocol

Now that the protocol has been covered from the frame level, the next step is to understand it from the character level. This information shows what is going on with the special and data characters and how they influence the protocol. The first information that must be discussed is how the characters are presented in this document for demonstration purposes.

Character	Symbol	Description
Control	c	Represents the control field of the frame.
Address	a	Represents the address field of the frame.
Data	d	Represents the data field of the frame.
CRC	x	Represents the a single CRC byte of the frame.
Flag	•	Represents the FLAG character that delimits the frame and is also used as the idle character.
ACK	A	Represents a single ACK response. It takes two of these for a proper protocol sequence.
RR	R	Represents a single Receiver Ready response. It takes two of these for a proper protocol sequence.

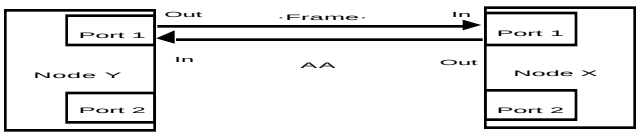
The example below shows a frame with a two byte address and an eight byte data field. There is also an example of an ACK and Receiver Ready response.



ACK and Receiver Ready Response Characters

In SSA there are two protocol characters that are used to control the flow of frames across an SSA link and this information should be understood before continuing. This section provides important information as to how frames are responded to by a destination node. To implement the necessary flow control, the receiving (destination) node sends the transmitting (source) node two 'responses' for each frame it receives:

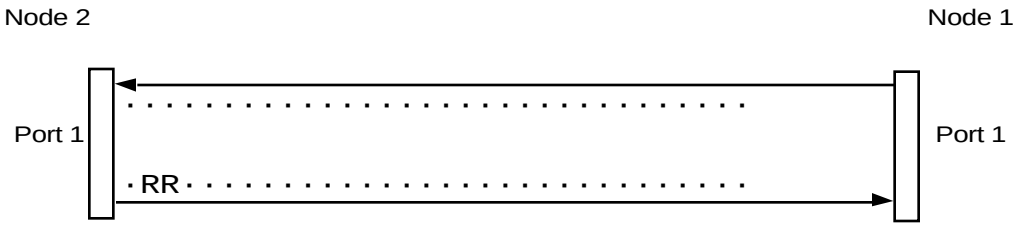
Receiver Ready A pair of consecutive RR protocol characters, i.e. RR - RR. This response is used to notify a node at the opposite end of a link that the receiving node has a buffer available to receive a frame of information. The RR response is also used to pace the transfers. As mentioned earlier, pacing ensures that a transmitting port does not overrun the available buffer space in the receiving port and the unit of pacing is a frame.

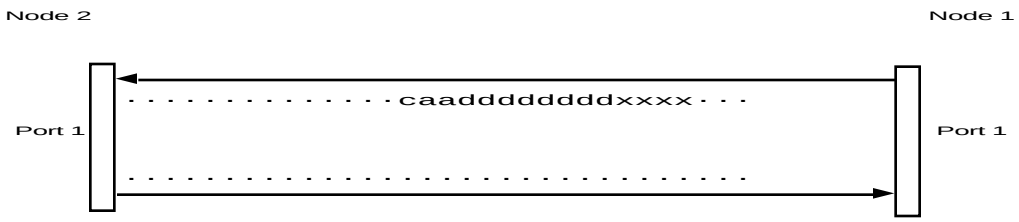


Acknowledgment A pair of consecutive ACK protocol characters, i.e. ACK - ACK. This response is used to notify the node at the opposite end of the link that the receiving node has received a frame and has validated the frames CRC. The link protocol requires a port to acknowledge every valid received frame. The remote port transmits an ACK Response when it receives a valid frame. When the local port receives the ACK Response, it can discard the transmitted data.

In simplistic terms the following events occur for every frame that is transferred across a link.

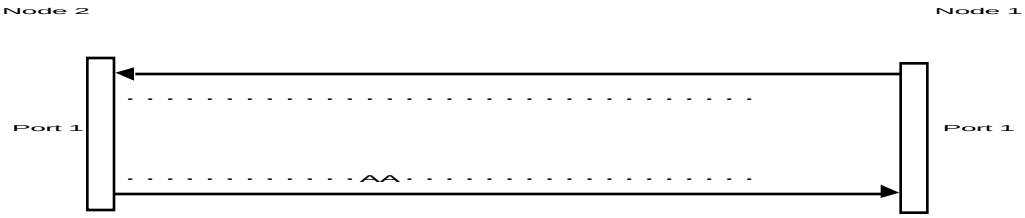
Node 2 sends an RR response to Node 1 when it has a buffer available to receive a frame. Also note that the two Nodes are exchanging FLAG characters to maintain link synchronization.





As soon as the RR response propagates down the cable to Node 1, it can transmit a frame to Node 2. This diagram demonstrates this.

Once the frame is received by Node 2 and the CRC is verified, Node 2 sends an ACK response to Node 1.



As soon as Node 1 receives and recognizes the ACK response it may remove the frame from its outbound buffer and make the buffer available for another frame.

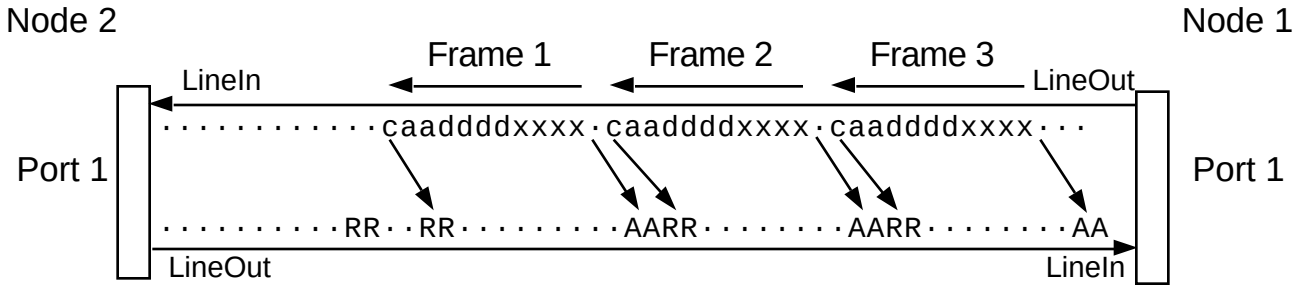
Note:
This is also happening in the opposite directions from Node 1 to Node 2. The next page demonstrates how this works in the full-duplex transfer example.

ACK and RR Sequence Examples

The following examples illustrate the link protocol. For compactness, the diagrams show a 2-byte address field and a 4-byte data field. Each port has a pair of A/B transmit frame buffers and a pair of A/B receive frame buffers, which is true in current silicon. 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.

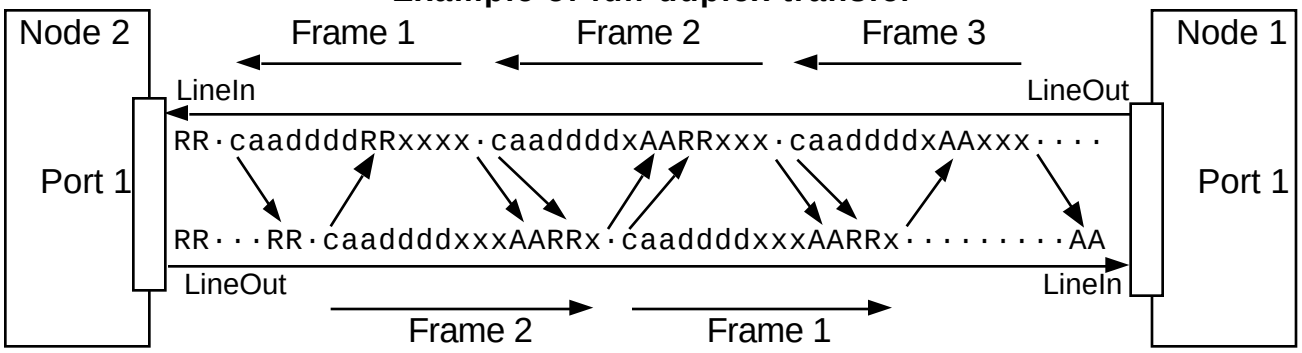
In the examples below, each character on the top line corresponds to the character immediately below on the bottom line. Propagation delay is not taken into consideration. The characters on the bottom line are in reality flipped over because the transmission is from left to right. However, both lines are shown as transmission from right to left for demonstration purposes.

Example of half-duplex transfer



The example above shows a half-duplex transfer with Node 1 acting as a source and Node 2 acting as a destination. Node 2 sends an RR response as soon as it detects the start of a frame since its receiver has buffer space for another frame (i.e. A/B frame buffering). This allows Node 1 to start sending another frame immediately after the trailing FLAG of the current frame. In addition, Node 2 issues an ACK response immediately when it detects the trailing FLAG of each frame.

Example of full-duplex transfer



The example above shows a full-duplex transfer with Node 1 acting as a source of 3 frames and Node 2 acting as a source of 2 frames. This scenario shows how 'responses' can be interleaved within a frame.

Note:
It is not necessarily true that the RR response is immediately after the ACK response as shown in the above examples. This was purely done for example purposes. The last example demonstrates the full-duplex bi-directional capability of the link.

Addressing and Routing

As mentioned earlier, in SSA there are no address switches that need to be set on each node and the network can change dynamically. This section will provide the information as to how this is accomplished in SSA.

The frame address field begins in the next data character following the control field. The format of the address field is identical for Privileged and Application frames. The format of the address field in Control frames is slightly different but the address and routing of these frames are handled in the same manner. The only difference between Application/Privileged and Control frames, is that the Control frames do not have a Channel component.

Accordingly the address field in Application and Privileged frames contains from 2 to 6 bytes, depending on the complexity of the network and the number of Channels implemented by the destination node. It consists of a Path component followed by a Channel component, as shown in the Figure below. Each component is a minimum of 1 byte and it can be extended a byte at a time.

Address field format for Application and Privileged frames

Path				Channel	
Byte 1	Byte 2	Byte 3	Byte 4	Byte 1	Byte 2

The address field is first used to route the frame over a selected **Path** to the destination node. Path addresses are geographic relative to the source node. This simplifies the routing hardware and it avoids the need to assign absolute node addresses at power-on.

The second function of the address field is to select a **Channel** within the receiving port at the destination node. A Channel consists of the facilities to receive a SMS or to receive a single data transfer. A SMS is a single frame that carries control information such as a command or status.

The size of the path and channel address components are determined by the preceding byte. Bytes 1, 2 and 3 of the path component and byte 1 of the channel component have the same format in that there is a special bit called the Extent bit (see below).

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Extent	Index						

When the **Extent** is set to a one it indicates that the current component (Path or Channel) is extended in the following byte. If the extent bit is set to zero it indicates that the current byte is the last byte of the current Path or Channel component. Typically the channel component is always one byte. The extend bit is only set in the path component when the frame is to be routed through a switch.

The **Index** component of the Path is a 7-bit unsigned integer which controls the routing of the frame to the destination node. If the Node is a dual-port node and the Extent bit is set to 1b, the Path component is decremented by a value of 1 and the Frame is forwarded to the next Node. If the Node is a **switch** then the Index directs the Frame to the Port (whose value is contained in the Index) and that byte of the Path component is deleted.

Path Addressing

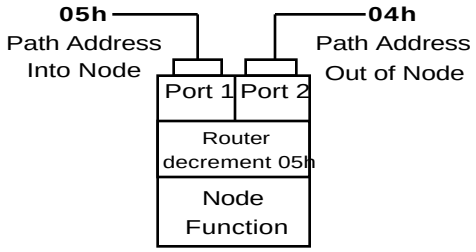
The path component routes the frame to the destination node and the length of the Path component is 1 - 4 bytes. Each byte corresponds to a string and/or a switch in the path from the source node to the destination node.

Note:
Each node in the network thinks its address is zero. This means that when a frame arrives at a node and the Path component is zero, then the frame is intended for the node. The node accepts the frame into its node function.

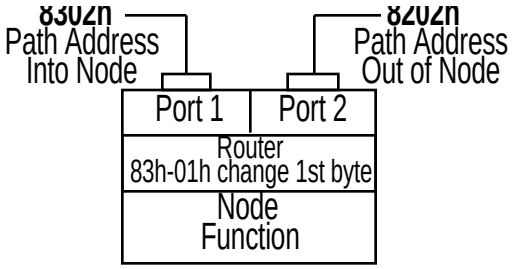
Addressing Rules:

- When a frame is forwarded by the router in a dual-port node the first byte of the Path address is decremented.
- When a frame is forwarded by a switch the first byte of the Path will be deleted. If it is deleted the next byte is exposed for subsequent routing decisions.

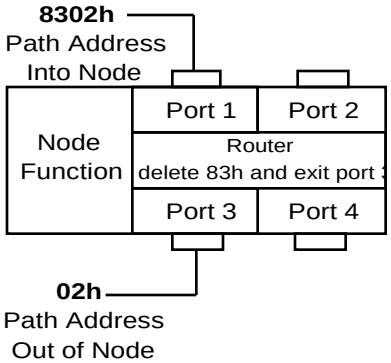
Example when a frame enters a dual-port node:
If the address field is not zero, the first byte of the path component is decremented and the frame is routed out the other port.



Example with the Extent bit set and a two-byte path component:
Notice that when a two-byte path component is routed through a dual-port node only the first byte is decremented and the second byte of the path component remains the same.



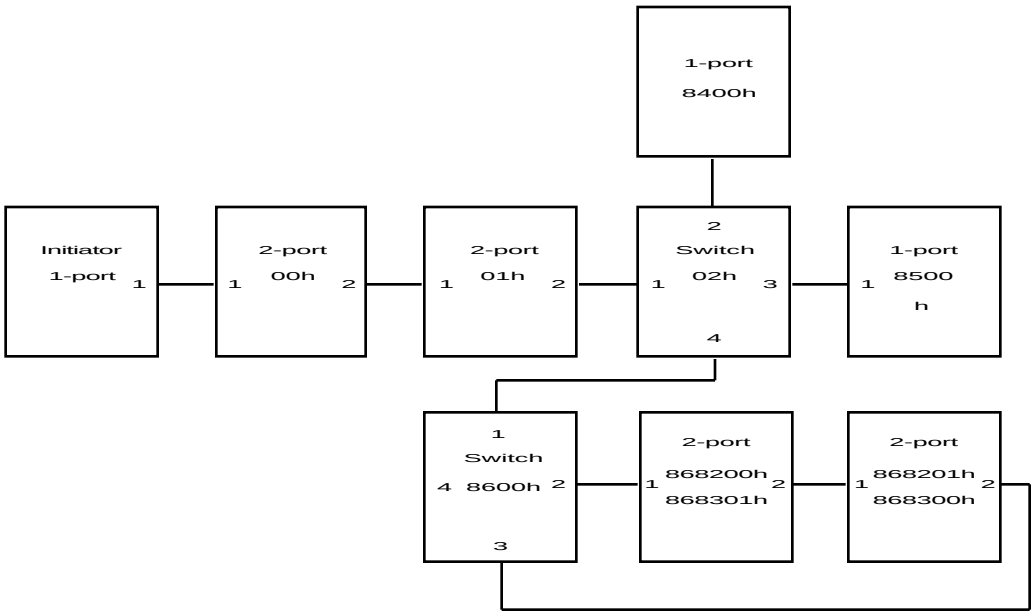
Example of routing through a switch:
When a frame is routed through a switch, the switch forwards the frame to the port indicated by the value of the Index component and the first byte of the path address is deleted.



Path Example

The Figure below represents a complex network. The numbers in the edges of the boxes are the (assumed) port numbers. For illustration purposes only the ports are numbered in a clockwise direction. The number inside each box is the hexadecimal Path address of that node relative to the indicated 'Initiator'.

All Initiators execute a Configuration process to discover the links and nodes that are operational within the network. An Initiator must also select a primary Path address to each node and calculate the corresponding return path. The dual-port nodes on the cyclic path beyond the switch (8600h) can also be reached by alternative paths from the Initiator. Alternate paths gives the interface benefits such as redundancy (i.e. when a link fails there is an alternate path to the node) and spatial reuse (which will be discussed later).



Routing

All routing is done in hardware. Depending on the address field of the inbound frame to the node, the router forwards the frame to the node function or to the outbound line of another port. When the node wants to originate a frame it instructs the router to transmit it on a specified port. Note that all **message** and **data** frames relating to a particular **command** use the same port.

Dual-port node routing:
When a frame enters a dual-port node and the path address is not zero, the first byte of the path address is decremented, the frame is routed through the serial interface chip to the other port, and new CRC is generated. This is all done in hardware.

Switch node routing:
When a frame enters a switch node and the address is not zero, the first byte of the path address is removed, the frame is routed out the serial interface chip to the port specified in the Index field of the path address, and new CRC is generated. This is all done by the hardware.

This routing capability demonstrates a main feature of the SSA interface, in that all routing is done in hardware. This means that the firmware is not involved in the forwarding of frames through a node, which gives the interface the benefit of speed and no software involvement.

Configuration and Error Recovery

Configuration

Every Initiator in the SSA network will have to perform a configuration process. This process is defined in the standards developed by the X3T10.1 committee. The following information will provide details as to how the configuration process works and how a network can be dynamically changed.

Each Initiator in the network performs the following configuration process:

(1) The Initiator finds an operational port on its node. This is determined by the fact that the port is receiving FLAG characters on its inbound line.
(2) The Initiator issues a Query_node SMS to a node with an address of zero.
(3) The node that received the Query_node SMS returns a Query_node_reply SMS with information about the number of ports on the node, which ports are operational, and a unique ID (an ID that is assigned by the node's manufacturer).
(4) The Initiator then sends a Query_node SMS out an operational port on the node to the next node in the string and so on, and so on, until the entire network is investigated.
(5) The Initiator does this until every node in the network has been discovered and entered into its 'Configuration table'. Targets also create 'Initiator tables' and make an entry into the table for every Initiator that sent it a Query_node SMS.

Master Node

In the SSA network there is an Initiator that must be configured as a Master node. The Master node is responsible for two main functions:

- Configuration of each node's port in the network as to where all Asynchronous Alerts are reported by the node.
- Coordinate error recovery procedures for the network.

In order to achieve the first objective, the Master node sends a Configure_port SMS to every port in the network, even the ports that are not operational.

Once the node/port combination has received the Configure_port SMS, the node knows where to send any Asynchronous Alert. Asynchronous Alerts are used to notify the Master node of an error or some other important condition. For instance, if a node detects an unrecoverable link error or detects that a new node has been connected to a previously not operational port, the node sends an Async_alert SMS to the Master node.

Due to the architecture, a port can detect when a new port has been added to a link because there is a beginning communication procedure that occurs when the node is powered-on. This procedure starts with each port on the node executing its POST. Once POST is completed each port transmits DIS characters on its outbound line and looks for DIS characters on its inbound line. When the port detects DIS characters, the ports at the opposite ends of the link synchronize with one another and start sending FLAG characters.

As soon as the ports are operational, the node(s) sends an Async_alert SMS indicating 'port now operational' to the Master. Once the Master receives this notification, it walks the new link and any node connected beyond it.

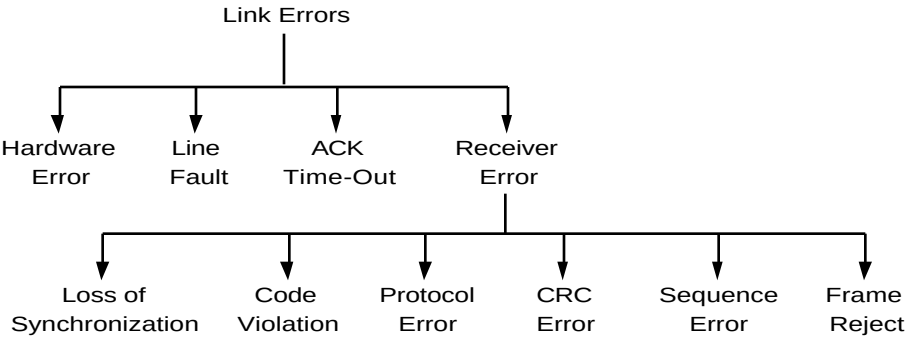
In order to achieve the second objective, the Master then notifies all other Initiators in the network that a new link has been detected.

This is accomplished by the Mater sending a Master_alert SMS to each Initiator in the network. Once each Initiator receives the Master_alert, they perform their configuration process on the new link and any node that exists beyond the new link. This capability makes configuration automatic and dynamic.

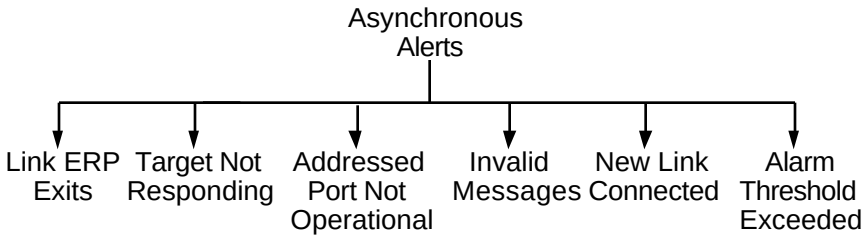
Error Recovery

In SSA, errors are typically handled in one of two levels:

Low level, i.e. by the firmware. This procedure, called Link ERP, is designed to recover from an error with minimal impact on the network and other operations. The diagram below represents the hierarchy and types of errors that can occur across a Link. Most of these errors can be recovered at the link level.



Higher Level, i.e. by the Master Initiator. If the error cannot be processed and handled between the nodes, via Link ERP, then the error must be passed back to the Master Initiator. As mentioned earlier these errors are known as Asynchronous Alerts and the diagram below represents the type of errors that are reported to the Master.



Link ERP

The Link Error Recovery Procedure is defined in the standard and provides the architecture with the ability to recover at the link level. Link ERP is used to recover from unrecoverable transient errors. These errors are typically due to transmission errors, i.e. noise.

This strategy ensures data integrity and it minimizes the impact to other operations. For example, an error will not result in bad data being written to a disk drive. Also an error affects only the commands that are currently using the failing link or node. All other commands in the network are free to execute.

Link ERP provides numerous benefits to the interface which include the following:

- The upper-level protocol is simplified since recovery is transparent if it is successful. This reduces overhead of the transfer because the Device Driver will not have to get involved in the recovery.
- There is normally no need to terminate any commands when a transient error occurs.
- There is no uncertainty about the state of the remote node.
- The compatibility of different SSA implementations is enhanced.

Link ERP Characteristics

(1) It is expected that the Link ERP will normally be **implemented in firmware** running on the node processor. However the functions could conceivably be performed by a hardware finite-state machine if performance is critical.

(2) Link 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.

(3) Link ERP is a multiple step process that is specified by the standard and all nodes recovery link errors via the same procedure.

(4) No matter how good the Link ERP is, some errors cannot be recovered from transparently, e.g. hardware errors or permanent line faults. The ERP has been carefully designed so that both nodes will always recognize an unrecoverable error and remain synchronized (if possible). In these cases the ERP exits. Where possible, recovery is then attempted by command retry, as described in "Asynchronous alerts".

Link ERP Basic Procedure

Note that Link ERP is a nineteen step procedure which is specified in the Transport Layer documentation approved by X3T10.1. The table below shows an overview of what happens when Link ERP is invoked by a port.

Local Port	Remote Port
Local port Detects Error, enters Check State and invokes Link ERP.	
Node prepares a Link Reset Frame which contains the Link Status Byte and transmits to remote port.	Remote port receives Link Reset Frame, checks the CRC, enters the Check State, and invokes its Link ERP.
Local port waits for remote port to ACK the Link Reset Frame.	Remote port transmits ACK response.
Node waits for remote port to return a Link Reset Frame.	Remote port prepares a Link Reset Frame which contains its Link Status Byte and transmits to Local port.
Local port receives Link Reset Frame, checks the CRC and returns ACK response.	Remote port waits for ACK response.
Link ERP makes important calculations concerning lost frame(s), if any.	Link ERP makes important calculations concerning lost frame(s), if any.
Port disables, synchronizing Link ERP and setting each ports hardware registers back to initial positions.	Port disables, synchronizing Link ERP and setting each ports hardware registers back to initial positions.
Port enters Ready State.	Port enters Ready State.
Re transmit frame(s) if necessary.	Re transmit frame(s) if necessary.

Protocol Example of Link ERP

The table below illustrates the operation of the Link ERP. Port X sends 2 frames back-to-back. Port Y receives the first frame correctly but the ACK response is corrupted by noise. Port X then detects a code violation and aborts the second frame. After the Link ERP completes Port X retransmits the second frame.

PORT X	-------LINE------>	<------LINE-------	PORT Y
Tx frame 1	Frame 1	FLAGS	
	Frame 1	RR response	Tx RR response
	Frame 1	FLAGS	
	FLAG	FLAG	Good CRC
Tx frame 2	Frame 2	**ACK **Error****	Tx ACK response
Port detects code	Frame 2	FLAGS	
violation, enters "Check	Frame 2	RR response	Tx RR response
state" and Aborts current	Frame 2	FLAGS	
frame.	ABORT	FLAGS	
	FLAGS	FLAGS	
Port invokes Link ERP	FLAGS	FLAGS	
Assemble LSB	FLAGS	FLAGS	
Tx Link Reset	Link Reset	FLAGS	
Wait to Rx Link Reset	FLAGS	ACK response	Tx ACK response
Wait to Rx Link Reset	FLAGS	FLAGS	Port enters Check state,
Wait to Rx Link Reset	FLAGS	FLAGS	and invokes its Link
Wait to Rx Link Reset	FLAGS	FLAGS	ERP.
Wait to Rx Link Reset	FLAGS	FLAGS	
			Assemble LSB
	FLAGS	Link reset	Tx Link Reset
Tx ACK response	ACK response	FLAGS	
	FLAGS	FLAGS	
Discard 1 frame	FLAGS	FLAGS	
	FLAGS	FLAGS	
Enter Disabled state	DIS	FLAGS	
Wait to detect DIS	DIS	DIS	Enter Disabled state
	DIS	DIS	Wait to detect DIS
Enter Enabled state	FLAGS	FLAGS	Enter Enabled state
Wait to Rx FLAG	FLAGS	FLAGS	Wait to Rx FLAG
Enter Ready state	FLAGS	FLAGS	Enter Ready state
Tx RR response	RR response	RR response	Tx RR response
Tx - Retransmits frame 2	Frame 2	FLAGS	
	Frame 2	FLAGS	
	Frame 2	FLAGS	Good CRC
	FLAGS	ACK response	Tx ACK response
	FLAGS	FLAGS	

Legend:
LSB Link Status Byte Tx Transmit Rx Receive

Asynchronous Alerts
This mechanism is used for many reasons, one of which includes when the Link ERP cannot recover from an error condition. All asynchronous alerts are handled by the Master node and the Master coordinates all recovery with other Initiators.

Asynchronous alerts are used to report events that are outside the scope of the Link ERP.

- A Link ERP exit. This could be a transient error that is unrecoverable at the frame level or a permanent error such as a disconnected link. The node detecting the error sends an Async_alert SMS to the Master indicating the reason for the Link ERP exit.

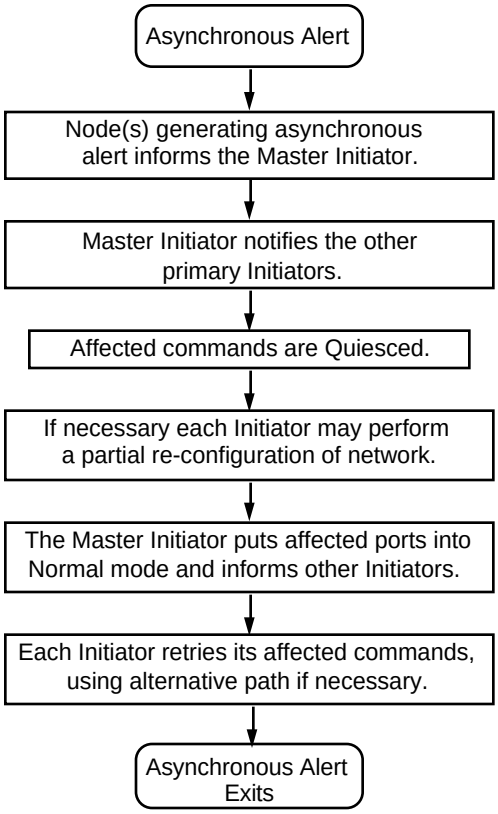
- A router receives a frame which is addressed to a port that is not operational. The node detecting the error sends an Async_alert SMS to the Master indicating 'Addressed port not operational'.

- A node receives an invalid message. Each message undergoes three levels of validation which includes byte 0 validation, message length validation, and field validation. The node sends an Async_alert SMS to the Master indicating conditions such as 'Unsupported Message' and 'Unknown Return_path'.

- A new link is connected to the network. When a new link is connected to the network a port on one of the existing nodes will become operational and the node sends an Async_alert SMS to the Master specifying 'Port now operational'.

- A ports Alarm threshold has been exceeded. When the Link ERP exceeds a pre-programmed limit the node that detects the condition sends an Async_alert SMS to the Master indicating 'Alarm Threshold Exceeded'.

The **Master** coordinates processing of asynchronous alerts as follows:



Each node in the network is guaranteed interface bandwidth i.e. each node is guaranteed that it will be allowed to originate a pre-programmed amount of frames. This is accomplished by the passing of a token around the network and quotas that are configured by the Master. When the capability of spatial reuse is added, the interface truly becomes a full-duplex, multi-transfer, and multiple I/O process network.

SAT Algorithm

SAT stands for 'satisfied'. A port is satisfied when it has originated at least a pre-defined frame quota, since it previously forwarded a SAT token, or if it has no frames to originate. A token is passed around the network and is used to make sure that each node in the network becomes satisfied.

The algorithm consists of the usage of two quota's, the a_quota and the b_quota. The quotas are configured by the Master and they could be different for each port. This algorithm permits spatial reuse of the individual links.

Quotas

One of the quotas (b_quota) is used when the node needs to originate a frame and the port that the frame is to be routed through is not holding a SAT token. This quota gives the port an upper limit of frames that can be sent through the port when there is no token. This number is typically four times the a_quota.

The a_quota is used when the port is holding the token. When holding the token, the port is allowed to originate up to a_quota before it has to forward the token. After the node has originated the number of frames specified by a_quota it forwards the token as pre-programmed by either reflecting or allowing the token to pass-through.

Note that if a node needs to forward or route a frame, the hardware will route the frame before it allows the node to originate a new frame, i.e. frame routing has priority over origination. Another SSA benefit is the fact that all of the routing is taken care of by the hardware. The application creates the frame (i.e. payload, address, etc...) in some local RAM and sets a bit in a register that notifies the hardware to originate a frame when there is bandwidth.

Current implementations require the firmware to determine if a token needs to be sent or created and whether the tokens are reflected or passed-through the port.

What does the SAT token do?

It gives each node's ports a fair shot at each links bandwidth. It also causes the quotas to be renewed when the port passes the token.

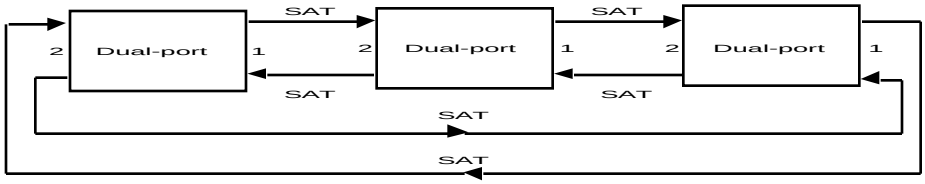
What does the token control?

The number of Application frames a node can origination through a specific port.

What happens to the token when it enters a nodes port?
Either the token is **passed through** the node to the other port or it is **reflected** out the same port in the opposite direction.

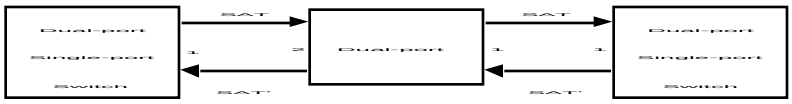
(1) **Pass through** Mode:
- (a) Only used in a loop or by a dual-port node in the middle of a string
- (b) When a SAT token enters Port 1 (Port 2), the node holds token until it is satisfied, then the SAT token is passed out Port 2 (Port 1) of the same node.

SAT operation for a loop (Pass through)



(2) **Reflect** Mode:
- (a) Is used by a node at the end of a string. The end of string is a single-port node, a dual-port node with one port not operational or a switch. Thus all ports must implement the SAT algorithm.
- (b) Can also be used by a dual-port node. In this case the algorithm is used to specify a SAT region (this will be demonstrated later).
- (c) When SAT enters a port at the end of the string, the port holds the token until it is satisfied and then reflects SAT as SAT' traveling in the opposite direction.
- (d) When SAT' enters a port at the end of the string, the port holds the token until it is satisfied and then reflects SAT' as SAT traveling in the opposite direction.

SAT operation for a string



When is a port Satisfied?
A port is SATISFIED if it has originated at least a pre-defined frame quota, a_quota, since it previously forwarded a SAT token, or if it has no frames to originate.

When is a port NOT Satisfied?
A port is NOT satisfied if:
- (a) it has to originate a frame,
- (b) it has no token,
- (c) and its b_quota is zero since previously forwarding the token.

When is a port allowed to originate an Application frame?

Subject to the link protocol, a port is allowed to originate an Application frame only if,

(1) It is holding the SAT token, OR,

(2) It has originated less than a pre-defined frame quota, b_quota, since it previously forwarded a SAT token, AND, there is no frame waiting to be forwarded from the other port.

What else is there to know about 'Quotas'?

(1) When a node forwards or reflects a token, it renews its frame quotas.

(2) Quotas are programmed by the Configure_port message and they can be different for each port.

(3) Quotas can be dynamically scaled by a node, however, the node must be able to support multiple concurrent outbound data transfers. This dynamic scaling is typically used by Initiators because they utilize multiple channels.

(4) Immediately after a port is reset A_quota = 1 and B_quota = 4. These default quotas are subsequently reprogrammed by a Configure_port message from the Master. A switch port may need higher static quotas than single-port and dual-port nodes if it is forwarding a lot of traffic from other ports.

What else is there to know about 'Tokens'?

(1) All ports incorporate a timer to detect loss of the SAT token due to a link error. If a port does not receive the next SAT token within 100 ms of forwarding the previous token then it generates a new token and renews its quota. This may lead to multiple SAT tokens circulating in the same direction if several nodes generate new tokens simultaneously. However the fairness algorithm will still operate correctly and the tokens will eventually merge into a single token as described below.

(2) If a port receives a second SAT token while it is still holding the previous SAT then it discards the second SAT.

(3) Each of the following functions shall not introduce a delay exceeding 10 character periods per node:
   (a) Forwarding the SAT token when the node is Satisfied.
   (b) Reflecting SAT as SAT' when the node is Satisfied.
   (c) Forwarding SAT'.
   (d) Reflecting SAT' as SAT.

(4) For a string or loop with only a few nodes the SAT token can rotate very rapidly if all nodes are Satisfied. This could impact frames being transmitted in the other direction. Therefore a node must not forward SAT until at least 100 character periods have elapsed since it received the *previous* SAT or SAT'.

(5) The SAT and SAT' tokens are special 10-bit characters so that they can be inserted within a frame. They can be transmitted whenever a User-defined character is permitted.

(6) A port shall be able to receive SAT and SAT' tokens while it is in any state, provided that character synchronization has been achieved. However, it only needs to transmit them in the Ready state. If the port is not Ready then it can hold any outgoing tokens until it becomes Ready.

Spatial Reuse

A feature that gives the network Simultaneous Multi-tasking capability. The network configuration example below demonstrates multi-tasking where multiple Initiators could be transmitting different processes to different targets simultaneously.

The topology of the network below is a simple loop topology. Initiator number 1 has been chosen as the Master, however any one of the other Initiators could perform this functionality. In fact any Initiator in an SSA network must be capable of performing the responsibilities of a Master. This requirement protects the network if the current Master breaks or some other Initiator really wants to become a Master (via Master Priority).

The loop can be divided into different 'regions' by configuring the ports to either pass-through or reflect the SAT token. In this example Initiators are going to reflect the token and Targets will pass the token through to the other port.

When required Initiators will cross regions to route configuration and error recovery SMSs to and from the Master. These SMSs are not controlled by the SAT algorithm. If users want to exchange Data, Application frames can cross region boundaries.

Spatial Reuse:
Initiator 1 Writes to Target 1 and Reads from Target 2,

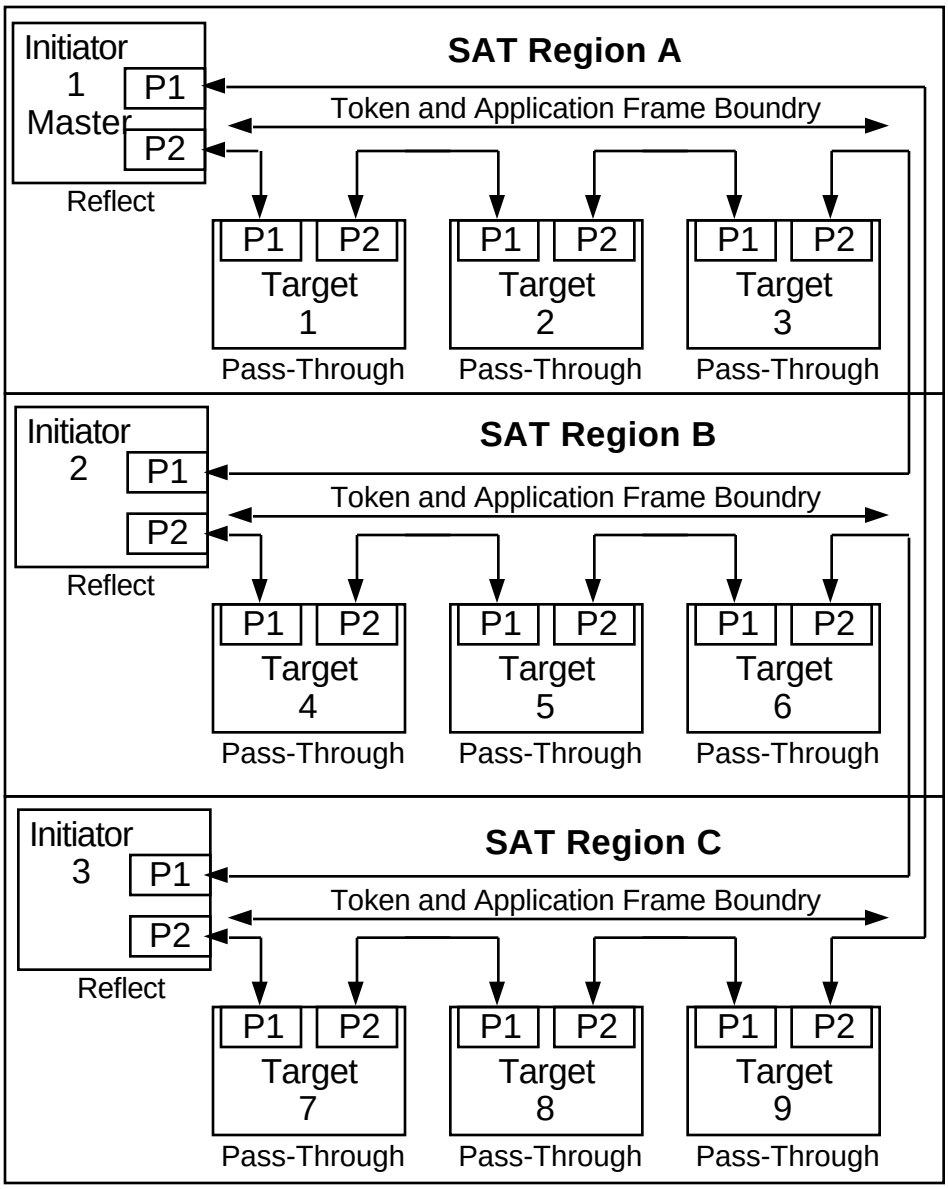
while

Initiator 2 Reads from Target 4 and Writes to Target 6,

while

Initiator 3 Reads from Target 8 and Writes to Target 9.

Simultaneous multi-tasking within a single network.



SSA Message Structures

The SSA Message Structures (SMSs) are used to transmit information between nodes. As mentioned earlier, there are three types of SMSs:

1. Privileged - used at the transport layer for configuration and error recovery.
2. Application - used by the upper level protocol to transmit command, actual customer data and status information. Also used at the transport level to prepare nodes for data transfers.
3. Control - only used at the transport level for node or link resets.

The Application and Privileged SMSs are sent in the **Data** field of a frame. Control frames have no Data and the type of reset being sent is determined in the Control field of the frame.

Privileged SMSs

Below are lists of all of the Privileged SMSs, a brief description, what type of device will send the SMS, and what is sent in response. The SMSs have been categorized into their respective functions. The first category will be the SMSs used for error recovery.

Error Recovery SMSs

SMS and Response	Originator	Description
Async_alert Async_reply	Initiators Targets	A node sends this SMS to inform the Master of an asynchronous alert. The Master will then forward the alert to each other Initiator as a Master_alert SMS.
Master_alert Response	Master Only	To forward an Async_alert SMS or an equivalent alert detected within the Master node itself. If the alert is for a Link ERP exit then the Initiator will Quiesce any I/O processes that were using the affected port before returning a Response SMS to the Master. If the error is permanent the Initiator will also de-configure the affected nodes.
Quiesce Response	Initiators	This SMS is sent from an Initiator to a Target during error recovery to abort all I/O processes that were started by a specified Initiator. The Target aborts the relevant I/O processes, sets the paths field to zero in the Initiator table, invalidates the associated Initiator_ID, and returns a Response SMS.
Async_reply no response	Master Only	This SMS is sent by the Master to acknowledge an Async_alert SMS.
Query_port Query_port_reply	Initiators	This SMS may be sent by an Initiator to request the error statistics of the specified port. This is useful when an Initiator wants to monitor a node's links for excessive invocation of Link ERP. The Target will respond to this SMS with a Query_port_reply SMS.
Query_port_reply no response	any	A node must return this SMS when it receives a Query_port SMS. This SMS returns the Link_error_count and Alarm_threshold values.

Below is the list of the Transport Layer SMSs used for network configuration.

Configuration SMSs

SMS and Response	Originator	Description
Query_node Query_node_reply	Initiators	This SMS is sent from a primary Initiator to every other operational node during the configuration process. Could also be used as a diagnostic tool to monitor all nodes in the network.
Query_node_reply no response	any	Each node must return this SMS when it receives a Query_node. The returned information includes the total number of ports implemented by the addressed node, which ports are operational, the number of the port currently being used and the Unique_ID of the node.
Configure_port Response	Master Only	During the Configuration process the Master sends this SMS to each port on every node in the network. Configure_port specifies the Return path and the Tag to be used when a node wants to send the Master an Async_alert SMS. It also establishes the operating parameters of the port.
Response no response	any	This SMS is used to report the completion status of the following Privileged SMSs: • Configure_port • Master_alert • Quiesce This SMS may also be used to report errors in other Privileged SMSs.
Query_protocol Query_protocol_reply	Initiators	This SMS may be sent by a primary Initiator to determine which upper level protocols (ULP) the destination node can receive. The receiving node will respond by returning a Query_protocol_reply SMS.
Query_protocol_reply no response	any	A node must return this SMS when it receives a Query_protocol SMS. This SMS indicates the particular ULP(s) that the node supports.

Application SMSs

Currently the only upper level protocol that is being mapped across the SSA interface is SCSI. However, the SSA architecture has been carefully designed to accommodate multiple/different upper level protocols within the same network and/or node. Below are lists of all the SMSs that are used to map SCSI onto an SSA Interface.

Command, Status and Response SMSs:

These SMSs are used to transfer Command information from the Initiator to the Target and Status information from the Target to the Initiator.

SMS and Response	Originator	Description
SCSI_command Data_ready, Data_request, data frames, SCSI_status	Initiators	This SMS is sent from the Initiator to a Target to initiate an I/O process or to send the next command in a linked list. The Target will respond by initiating a data transfer and/or by sending a SCSI_status SMS.
SCSI_status no response	Targets	This SMS is sent from a Target to the Initiator to indicate that a command and any associated data transfer have been terminated or completed. The SMS is returned using the path specified by the Initiator_ID in the associated SCSI_command SMS.
SCSI_response no response	any	SCSI_response is used to report the completion status of the following SMSs: • Abort_tag • Abort • Clear_queue • Device_reset • Clear_ACA_condition SCSI_response may also be used to report errors in other Application SMSs.

SCSI Architecture SMSs:
These SMSs are used by the upper level protocol to perform SCSI architectural functions.

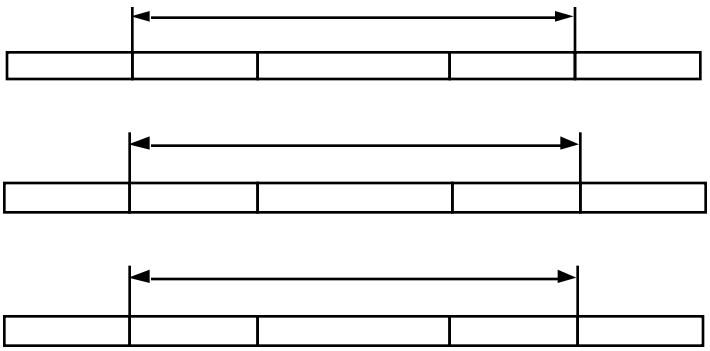
SMS and Response	Originator	Description
AER_enable Data_ready and then data frame.	Initiators	This SMS is sent from the Initiator to the Target to enable future Asynchronous Event reporting by supplying a Tag and Return_path to be used by the Target when reporting Asynchronous Event Sense Data. This SMS is only valid in SCSI-3.
Abort_tag SCSI_response	Initiators	This SMS is sent from the Initiator to the Target to abort a particular I/O process. Other I/O processes are not affected. The Target terminates execution of the I/O process if it has already begun or the Target removes the I/O process from a queue if execution has not begun. The Target always replies to the Abort_tag with a SCSI_response SMS.
Abort SCSI_response	Initiators	This SMS is sent from an Initiator to a Target to abort all I/O processes from that Initiator for a selected Logical Unit. I/O processes from other Initiators are not affected. The Target always replies to the Abort SMS with a SCSI_response SMS.
Clear_queue SCSI_response	Initiators	This SMS is sent from an Initiator to a Target to abort all I/O processes from all Initiators for a selected Logical Unit. The Target sets Unit Attention for all Initiators for which an I/O process was aborted. The Target always replies to the Clear_queue SMS with a SCSI_response SMS.
Device_reset SCSI_response	Initiators	This SMS is sent from an Initiator to a Target to clear all I/O processes for all Initiators on all Logical Units. The Target resets all Logical Units and sets Unit Attention for ALL Initiators. The Target always replies to the Device_reset SMS with a SCSI_response SMS.
Clear_ACA_condition SCSI_response	Initiators	This SMS is sent from an Initiator to a Target to clear an Auto Contingent Allegiance condition for that Initiator and a selected Logical Unit. After the Auto Contingent Allegiance condition is cleared, any suspended queued command for that Initiator may become an active I/O process subject to the ordering rules. Auto Contingent Allegiance conditions for other Initiators, Logical Units or Target Routines are not affected. The Target always replies to the Clear_ACA_condition SMS with a SCSI_response SMS.

Data Transfer Control SMSs:
Below is the list of the transport layer SMSs used for data transfer control. Even though these are Application SMSs, they are specified in the Transport Layer specification. This fact requires all upper level protocols to implement data transfers in the same manner. This ensures that compatibility across different protocol platforms, i.e. SCSI-2, SCSI-3, IPI, etc..., is accomplished.

SMS and Response	Originator	Description
Data_ready Data_reply	Targets	This SMS is sent by the **Target to** the **Initiator** during the execution of a command that will transfer data to the Initiator (i.e. READ operation). The Initiator returns one or more Data_reply SMSs for each Data_ready SMS. A Data_ready SMS is considered outstanding from the time it is sent until the last data frame associated with it is sent.
Data_reply Data frames from Target to Initiator	Initiators	This SMS is sent by the Initiator to the Target in response to a Data_ready SMS. The Target replies by sending the requested data frames. A Data_reply SMS is considered outstanding from the time it is sent until the last data frame associated with it is received.
Data_request Data frames from Initiator to Target	Targets	This SMS is sent from a **Target to** an **Initiator** during the execution of a command that will transfer data to the Target (i.e. WRITE command). The Initiator responds by sending the requested data addressed to the specified Target Channel. The Target may use more than one Data_request SMS to transfer all of the data for a particular command. For example, the Target may have limited buffer space. A Data_request SMS is considered outstanding from the time it is sent until the last data frame associated with it is received.

Control SMSs

These SMSs are only used to reset a node or link. The node resets are Total and Absolute. The Link Reset is only used during the Link Error Recovery Procedure. As mentioned earlier, the frame type is determined in the control field of the frame. The pictures below show the format of each reset frame.



Examples of SMS Protocol

The tables below show the SMSs exchanged between the Initiator and Target for a typical SCSI READ command.

DDRM (Disable Data Ready SMSs) = 0b in the SCSI_command.

(1) This is a more common case in a system with limited buffer pointers and large numbers of outstanding commands (heavy queuing).
(2) The Data_ready and Data_reply SMSs add a very slight bus overhead, a slight latency hit (can be reduced by early SMSs and overlapped SMSs), and some processor overhead (unless done in hardware).
(3) There can be more than one Data_ready SMS, e.g. an out-of-order READ could have two Data_ready SMSs.
(4) There can be more than one Data_reply SMS for each Data_ready SMS.

Initiator	Target
SCSI_command	
	Queue command
	Execute command
	Data_ready
Initialize Channel and set buffer pointers	
Data_reply	
	DATA frame
	•
	•
	•
	DATA frame
Free Channel when Byte_count satisfied	
	SCSI_status

DDRM = 1b in the SCSI_command.

(1) When DDRM is set to 1b the Data_ready and Data_reply SMSs are not generated.
(2) This case is used when buffer pointer sets are not a critical resource and can be held during command execution.
(3) This can be the case when the system has a large number of buffer pointers (for DMA context) and there are only a few commands outstanding at any given time.

Initiator	Target
Initialize Channel and set buffer pointers	
SCSI_command	
	Queue command
	Execute command
	DATA frame
	•
	•
	•
	DATA frame
Free Channel when Byte_count satisfied	
	SCSI_status

The table below shows the SMSs exchanged between the Initiator and Target for a typical SCSI WRITE command.

There can be more than one Data_request SMS, e.g. if the Target has limited buffer space.

Initiator	Target
SCSI_command	
	Queue command
	Execute command
	Initialize Channel and set buffer pointers
	Data_request
DATA frame	
•	
•	
•	
DATA frame	
	Free Channel when Byte_count satisfied
	SCSI_status

Bibliography

SSA documentation is split into many documents. Since October 1994 the standards have been drastically changed and reorganized. These changes were necessary to make the interface more complete in its application. The documentation is split between the UIG documents, which will be used prior to the approval of the X3T10.1 documents, and the actual X3T10.1 documents that must go through the standard approval process.

Before SSA is Approved by X3T10.1

- These documents will be used by system and device manufacturers developing products **prior** to the X3T10.1 standards being approved. The UIG has taken the necessary precautions to ensure that these documents include any changes included in the X3T10.1 documents.
- **UIG95PH**; This document is under control of the SSA User Industry Group. It is a combination of the X3T10.1 SSA-PH1 and SSA-TL1 documents and most all information in this document is identicle to those standards.
- **UIG95SP**; This document is under control of SSA-UIG and is the basis document for mapping the SCSI-2 protocol across the SSA physical interface. Mapping includes the use of SCSI-2 commands, status, and sense data. This document is an exact copy of the SSA-S2P document that will be approved by the X3T10.1 standards committee.

SSA When Approved By X3T10.1

- These will typically be systems and devices developed **after or during** 1995.
- **SSA-PH1**; This document is under control of X3T10.1 and it covers the Physical Interface which includes: cables, connectors, transfer speeds, etc...
- **SSA-TL1;** This document details the Transport Layer which includes: link management, link protocol, framing, addressing, buffering, resets, configuration, and error recovery.
- **SSA-S2P**; This document details the SCSI-2 mapping of SSA interface across the SSA Transport Layer.
- **SSA-S3P**; This document details the SCSI-3 mapping of SSA interface across the SSA Transport Layer.

Future SSA

- These documents will enhance the SSA interface to who knows what, faster transfer speeds, bigger frames, etc... no one really knows yet. Currently it is the objective of the X3T10.1 standard committee to keep this level of implementation compatible with the PH1 and TL1 standards.
- **SSA-PH2**; Yet to be developed but will remain under control of X3T10.1 and cover the Physical Interface which includes: cables, connectors, transfer speeds, etc...
- **SSA-TL2;** This document will detail the future Transport Layer which will include: link management, link protocol, framing, addressing, buffering, resets, configuration, and error recovery.

Sources of Information

Copies of SSA Documentation

Global Engineering
15 Inverness Way East
Englewood, CO 8011-5704
(800) 854-7179 or (303) 792-2181

Additional Copies of This Paper

SSA Industry Association
Dept. H65/B-013
5600 Cottle Road
San Jose, CA 95193-0001
408/256-5656 Fax: 408/256-0595

What is a reflector you may ask. It is a method by which all persons involved in developing a standard can communicate with one another via E-mail. Following is a list of all the available reflectors and their E-mail address.

Reflector Name	Subscribe/Unsubscribe	Broadcast to Reflector
SCSI	scsi-request@wichitaks.ncr.com	scsi@wichitaks.ncr.com
SSA	majordomo@dt.wdc.com	ssa@dt.wdc.com

All of the majordomo reflectors are automatic. To subscribe or unsubscribe, send a message to the majordomo address

majordomo@dt.wdc.com

Make sure the subject line is blank and type a line in the message body of the following format:

```
subscribe ssa yourname@company.com
```

Also make sure to turn off any 'signature' information that your email program may append to the message.

The SCSI BBS can also be used to gain access to documentation in electronic format.

SCSI BBB: 719-574-0424

Glossary

ACA	Auto Contingent Allegiance
Application	A process that operates the node
Application frames	Frames used by the upper-level protocol.
BER	An acronym for Bit Error Rate
CFE	Common Front End bus
channel	The facilities in a port to receive a message or a single data transfer
character	A sequence of 10 encoded bits that represents a data byte or a protocol function
CMOS	Complementary Metal Oxide Semiconductor
Control frames	Frames used for resets.
CRC	Cyclic Redundancy Check
Cut-through routing	A property of the SIC where a frame is automatically routed through the node.
destination	The node that receives a particular frame
DMA	Direct Memory Access
EMI	Electro-magnetic interference
ERP	Error Recovery Procedure
ESD	Electro Static Discharge
FCS	Fiber Channel Standard
FDDI	Fiber Distributed Data Interface
field	A group of related data characters in a frame, e. g. the CRC field
FLAG	The characters that mark the beginning and end of a frame.
frame	a unit of data transfer that contains a control field, an address field, a data field and a CRC field. A sequence of 4 or more data characters surrounded by FLAG characters
frame multiplexing	the system of telegraphy allows two or more messages to be sent simultaneously in either direction over the same cable.
FSN	Frame Sequence Number
IDC	An acronym for Insulation Displacement Connector, a common connector which press fits onto an insulated ribbon cable.
Initiator	The node that issued a particular command
line	A physical connection between a transmitter and a receiver
Link	A serial connection between 2 nodes
Link ERP	Link Error Recovery Procedure, the procedure that is used to allow a link to recover from an error, usually implemented in firmware.
Loop	A cyclic network containing dual-port nodes only
LRG	Longitudinal Redundancy Check
LSI	Large Scale Integration

Master	A unique Initiator that coordinates error recovery and network configuration
message aka SMS	A single frame that carries control information, e.g. a command or status
modulo 4	the method used to increment FSN, RSN and TSN, two bit binary counting, values include 0 to 3, then back to 0 and start over.
Network aka Web	consists of two or more nodes inter-connected by serial links.
Node	A system, controller or device with one or more serial links.
Node Function	The specific responsibility, task, or use of the node. The reason for the nodes existence.
NRZ	Non-Return-to-Zero
Path	The links and intermediate nodes used to transfer a frame from the source to the destination
POR	Power-On Reset
Port	The hardware and firmware that implements one end of a link.
POST	Power-On Self-Test
Privileged frames	Frames used by the transport layer for configuration and error recovery.
RAS	Reliability, Availability and Serviceability
receiver	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
SERDES	Serializer/Deserializer, converts parallel frames coming into the SIC into a serial stream (serialize), and converts serial frames in the SIC to a parallel form ready for transmission.
SIC	Serial Interface Chip
SMS	SSA Message Structure
Spatial Reuse	An important property of strings and loops by which data is only routed over the links specified between the source and destination (as opposed to broadcast to all links). This allows several transfers to occur at full bandwidth, provided they use different links.
Spindlesync	A mechanism used in Disk Array applications to keep all disk drives index marks synchronized with one another.
String	A linear network of dual-port nodes. The extreme nodes at either end can also be switches or single-port nodes.
Switches	The component used to allow a number of strings to be connected to achieve an almost unlimited number of nodes.
Target	The node that executes a particular command
transmitter	The logic that drives the outbound line
TSN	Transmit Sequence Number