ECSS-E-50-12 Space Engineering

SpaceWire:

SERIAL POINT-TO-POINT LINKS

DRAFT

DOCUMENT NUMBER	UoD-DICE-TN-9201
ISSUE	ISSUE D
DATE	May 2000
ESA CONTRACT NO.	12693/97/NL/FM
ESA TECHNICAL MANAGER	J. Rosello
AUTHOR	Dr SM Parkes, University of Dundee
CONTRIBUTORS	SPACEWIRE WORKING GROUP

Document Revision History			
Revision	**Date**	**Responsible**	**Modifications/** **Reasons for Change**
Draft A	Sept. 1999	SM Parkes	Initial release
Draft B	Oct. 1999	SM Parkes	Corrections from initial comments from ESTEC and Dornier Satellitensysteme on draft A.
Draft C	Jan 2000	SM Parkes	Corrections following comments received during SpaceWire Working Group meeting on 27th October and Core Group meeting on 8th December. Also reflects comments received by email.
Draft D	May 2000	SM Parkes	Minor corrections to sections 4, 5 and 6 (physical, signal and character levels). Major changes to section 7 (exchange level) following detailed analysis and simulation exercises. Error handling added to packet level (section 8). Network level (section 9) defined and made normative. Error recovery scheme (section 10) defined. Conformance criteria (section 11) expanded to include Routing Switch and Network. Another difference between IEEE-1355 and SpaceWire added to Annex C.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

ACKNOWLEDGEMENTS

SpaceWire is the result of the efforts of many individuals within the European Space Agency, European Space Industry and Academia.

1. OVERVIEW

1.1 PURPOSE

The SpaceWire standard addresses the handling of payload data on-board a spacecraft. It is a standard for a high-speed data link, which is intended to meet the needs of future, high-capability, remote sensing instruments and other space missions. SpaceWire provides a unified high-speed data-handling infrastructure for connecting together sensors, processing elements, mass-memory units, downlink telemetry sub-systems and EGSE equipment.

The purpose of this standard is

- to facilitate the construction of high-performance on-board data-handling systems,
- to help reduce system integration costs,
- to promote compatibility between data-handling equipment and sub-systems,
- to encourage re-use of data-handling equipment across several different missions.

1.2 SCOPE

The SpaceWire standard specifies the physical interconnection media and data communication protocols to enable data to be sent reliably at high-speed (between 2 Mbps and 100 Mbps or more) from one unit to another. SpaceWire links are full-duplex, point-to-point, serial data communication links.

The scope of this standard is the physical connectors and cables, electrical properties, and logical protocols that comprise the SpaceWire data link. SpaceWire provides a means of sending packets of information from a source node to a required destination node. SpaceWire does not specify the contents of the packets of information.

The SpaceWire standard covers the following normative protocol levels

- **Physical Level:** Defines connectors and cables.
- **Signal Level:** Defines signal encoding, voltage levels, noise margins, EMC specifications and data signalling rates.
- **Character Level:** Defines the data and control characters used to manage the flow of data across a link.
- **Exchange Level:** Defines the protocol for link initialisation, flow control, link error detection and link error recovery.
- **Packet Level:** Defines how data to be transmitted via a SpaceWire link is split up into packets
- **Network Level:** Defines the structure of a SpaceWire network and the way in which packets are transferred from a source node to a destination node across a network. Defines how link errors and network level errors are handled.

SpaceWire is based on two existing commercial standards, IEEE 1355-1995 [RD1] and Low Voltage Differential Signalling [AD1, RD2] which have been combined and adapted for use on-board spacecraft.

1.3 GUIDE TO THE STANDARD

This SpaceWire standard document begins with section 1 (this section) which covers the purpose and scope of the standard, gives a list of applicable and reference documents, summarises the rationale for the standard and introduces the notation used throughout the standard document. A statement regarding intellectual property is also given. Section 2 provides the necessary definitions of conformance keywords and a technical glossary. A brief overview of the standard is given in section 3 to familiarise the reader with the basic SpaceWire concepts, prior to the detailed specification of subsequent sections. Section 3 also aims to provide some explanation of the key decisions made about the SpaceWire standard.

The body of the SpaceWire standard is presented in sections 3.6-9, which ascend through the various normative levels of the standard.

- ❑ Section 4 (Physical Level) covers cables, connectors and cable assemblies.
- ❑ Section 5 (Signal Level) deals principally with electrical characteristics, EMC specifications, coding and signal timing.
- ❑ Section 6 (Character Level) describes how data and control characters are encoded.
- ❑ Section 7 (Exchange Level) presents the way in which a SpaceWire link operates including link initialisation, normal operation, error detection and error recovery.
- ❑ Section 8 (Packet Level) describes the way in which data is encapsulated in packets for transfer across a SpaceWire network.
- ❑ Section 9 (Network Level) deals with the structure and operation of a SpaceWire network.

The error recovery scheme is described as a whole in section 10, which brings together the error detection, error recovery and error reporting mechanisms from all the protocol levels to aid comprehension.

The SpaceWire standard concludes in section 11 with a list of conformance statements, highlighting those parts of the standard that must be followed for a system to be SpaceWire compatible.

There are four annexes which present:-

- ❑ Initial requirements for SpaceWire.
- ❑ A summary of SpaceWire EMC performance related to the needs of a typical spacecraft.
- ❑ The differences between SpaceWire and IEEE standard 1355-1995 [RD1].
- ❑ List of informative references.

1.4 Normative References

This standard shall be used in conjunction with the following publications:

AD1 Telecommunications Industry Association, "Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits", ANSI/TIA/EIA-644-1995, March 1996[1].

AD2 ESA/SCC Generic Specification No. 3902[2]

AD3 ESA/SCC Generic Specification No. 3401/029

Note: Informative bibliography can be found in Annex D.

[1] ANSI publications are available from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (http://www.ansi.org)

[2] ESA/SCC publications are available from the ESA/SCC secretariat (http://www.estec.esa.nl/qcswww). ESA/SCCNo 3902 and No 3401/029 have also been posted at http://www.estec.esa.nl/tech/spacewire/techmodules.html#Cables_connectors

1.5 RATIONALE FOR STANDARD

SpaceWire technology has grown organically from the needs of on-board processing applications. This SpaceWire Standard document provides a formal basis for the exploitation of SpaceWire in a wide range of future on-board processing systems. It aims to secure the benefits of equipment compatibility and reuse.

1.5.1 Brief History of Development

The high-speed data links that evolved into SpaceWire were initially developed for use in multiple DSP processor systems. The links were seen as a means of helping to solve demanding on-board signal and image processing problems by connecting together several programmable DSP processors and other processing devices into a high performance parallel processing system [RD3, RD4, RD5]. An integrated architecture was foreseen where various processing elements were connected into a heterogeneous network. The proposed data link was based on IEEE standard 1355-1995 [RD1].

The wider application of the data link became apparent when it was used to form a versatile reconfigurable solid-state memory for space applications [RD6]. This brought benefits of modularity and fault tolerance to a simplified solid-state memory system. IEEE-1355 type links and routing switches were used to send data to and read data from an array of memory modules.

The extension of the emerging architecture to embrace sensors, down-link telemetry and EGSE into a unified on-board data handling infrastructure was natural [RD7, RD8].

With the growing interest in IEEE-1355 type links for space applications it was important to consider issues relating to space qualification. Dornier Satellitensysteme was developing IEEE-1355 encoder/decoder devices in radiation tolerant technology [RD9, RD10], but work remained to be done on the line drivers/receivers, cables and connectors and EMC performance. The Digital Interface Circuit Evaluation (DICE) study was initiated to examine these issues [RD11]. The DICE study resulted in the SpaceWire standard document.

1.5.2 Main Features

An overview of the main features necessary for a data link for use in space applications are listed below [RD11].

- **Data Rate:** A data link shall have sufficient capacity or bandwidth to carry the data for which it was intended. 100Mbaud is an appropriate minimum target for the maximum data rate.
- **Distance:** The data link shall operate over a distance of at least 10m. This distance is commensurate with the size of a large spacecraft enabling data to be transmitted from one extremity to the other.
- **Scalability:** To meet the data rate requirements of particularly demanding applications it shall be possible to use several links in parallel to increase the data rate accordingly.
- **Error Rate:** The error rate on the link shall be low, better than a BER (bit error rate) of 10^{-12} for the basic link and better than 10^{-14} for a link protected by a higher level error detection protocol.
- **Power Consumption:** The power consumption of the link shall be low.
- **Low mass and small size:** The mass and size of the data link interface and the cable shall be as small as possible.
- **Cold Redundancy:** The data link shall support connection within a cold redundant system, i.e. when part of the system is powered and another part is not powered.
- **EM Susceptibility:** The data link shall not be susceptible to interference from external electromagnetic sources. It should meet the EM susceptibility requirements of most space missions.

- **EM Emission:** The data link shall not emit electromagnetic radiation at a level that would interfere with the operation of other systems. It should meet the EM emission requirements of most space missions.
- **Magnetic Emission:** Magnetic emissions from the data link shall be low – ferrous materials should not be used in the data link components.
- **ESD Immunity:** The electronic devices forming a link shall have a high level of immunity to damage by electro-static discharge.
- **Galvanic Isolation:** It should be possible to galvanically isolate the data transmission system from the data reception system.
- **Radiation Tolerance:** The components that implement the data link shall be tolerant of radiation.

Detailed requirements are listed in Annex A.

1.5.3 Equipment Compatibility and Reuse

One of the principal aims of SpaceWire is the support of re-use at both the component and sub-system levels. In principle a data-handling system developed for an optical instrument, for example, can be used for a radar instrument by unplugging the optical sensor and plugging in the radar one. Processing units, mass-memory units and down-link telemetry systems developed for one mission can be readily used on another mission reducing the cost of development, improving reliability and most importantly increasing the amount of scientific work that can be achieved within a limited budget.

Integration and test of complex on-board systems is also supported by SpaceWire with ground support equipment plugging directly into the on-board data handling system. Monitoring and testing can be carried out with a seamless interface into the on-board system.

A unified data-handling architecture is essential for cost effective missions – SpaceWire provides the basis for such a high-speed on-board data-handling architecture.

1.6 DISCLAIMER – INTELLECTUAL PROPERTY

The implementation of this standard may require the use of intellectual property covered by patent rights. ESA shall not be responsible for identifying all the patents for which a license is required to implement the SpaceWire standard. Furthermore, ESA shall not be responsible for ensuring the existence or legal validity of any patent related to the SpaceWire standard.

1.7 DOCUMENT NOTATION

1.7.1 Signal Naming

All electrical signals are shown in uppercase letters.

The two signals making up a differential pair are given the suffixes + and – to indicate the positive and negative components of the differential signal respectively.

The SpaceWire differential signals are referred to as D+,D- and S+,S- for data and strobe respectively. When considering the driven end of a SpaceWire link these signals may be designated Dout+, Dout- and Sout+ and Sout- for data and strobe respectively. Similarly the signals at the input end of a SpaceWire link are Din+, Din- and Sin+, Sin-.

1.7.2 Packet Formats

Packet formats are represented in two ways in this document. The first way is graphical and is shown in Figure 1-1. The field at the top is the one that is transmitted first.

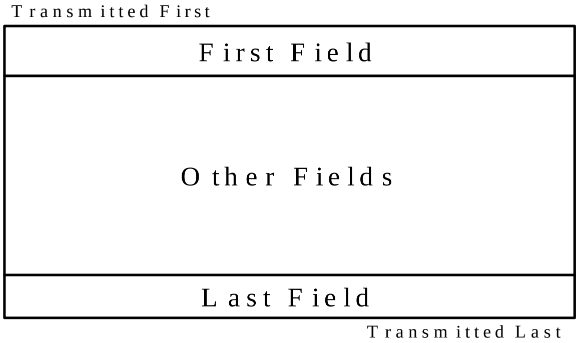


Figure 1-1 Graphical Packet Notation

The second packet representation is textual. Each field is enclosed in chevrons <>. The fields comprising a packet are written left to right in the order that they are transmitted. The example below is equivalent to that shown in Figure 1-1.

EXAMPLE:

<First Field><Other Fields><LastField>

1.7.3 State Diagram Notation

All state diagrams in this standard use the style shown in Figure 1-2. States are represented by ellipses with the state name written inside the ellipse in bold. Actions to be taken while in a particular state are written in italics inside the ellipse underneath the state name. Transitions from one state to another are indicated by arrows. The event that causes a transition is written alongside the arrow. Unconditional transitions are indicated by arrows without an event name written next to it. Reset conditions are indicated by transition arrows that start in empty space. Transitions may be enabled by a guarded condition so that the transition only takes place if the guard condition is true. Guard conditions are written in square brackets alongside the transition they affect.

State names referred to in the text of the standard are in italics e.g. *FirstState.*

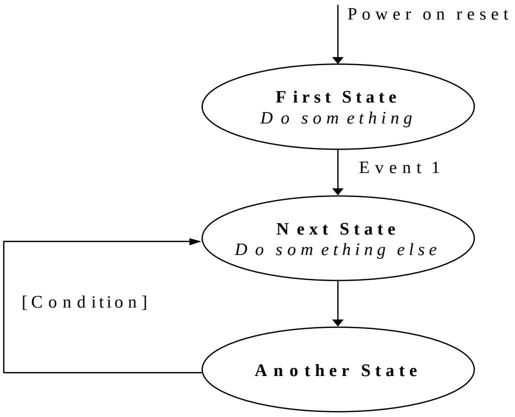


Figure 1-2 State Diagram Style

2. DEFINITIONS

2.1 CONFORMANCE GLOSSARY

The following keywords are used to differentiate between different levels of requirements and optionality, as defined in IEEE Std 100-1992 [RD12].

Shall: indicates a normative requirement. To ensure interoperability with other products conforming to this standard, all normative requirements must be followed strictly with no deviation.

Should: indicates a recommended but not normative requirement. Allows flexibility of choice between several possible alternatives while indicating a strongly preferred alternative. Indicates that a certain course of action is desirable but not normative, or indicates that a certain course of action is deprecated but not prohibited.

May: indicates a suggested course of action without implying preference over any other possible course of action.

2.2 TECHNICAL GLOSSARY

ACK: Acknowledge

Acknowledge: An indication that a message has been received successfully by its intended destination.

AWG: American Wire Gauge

BER: Bit Error Rate

Binder: A layer of tape wrapped around one or more cables to keep them together in a fixed position. The tape is usually PTFE and is wrapped in an overlapping spiral along the length of the cables to be bound.

Bit Error Rate: The ration of the number of bits received in error to the total number of bits sent across a link.

Byte: Eight bits

CAM: Content Addressable Memory

Cargo: The data that is to be encapsulated in packets and transferred from a source to a destination.

Character: A control character or data character

Character Level: The protocol level that deals with the encoding of data and control characters into a bit-stream.

Check Sum: A byte or word added to the end of a message or packet used for error detection purposes. The check sum is formed by adding together all the bytes in the message. The addition is done modulo the size of the check sum e.g. 256 for a byte-sized checksum. The receiver also calculates the message checksum and compares it against the one sent with the message to check for errors.

Coding: Translation from one set of bits to another new set of bits.

Content Addressable Memory: A memory array which is accessed by searching for a match between an input data value in the contents of the memory array. The output from the memory array is the index of the location that holds the searched for value.

Control Character: A character that is used to control a link. Control characters include the Link-characters (NULL and Flow Control Token, FCT) and the end of packet markers (EOP and EEP).

Data Character: A data byte encoded ready for transfer across a link.

Data Rate: The rate at which the application data is transferred across a link.

Data Signalling Rate: The rate at which the bits constituting control and data characters are transferred across a link.

Data-Strobe: An encoding scheme in which a sequence of data bits (and clock) is encoded as the original data bit sequence, together with another bit sequence (strobe) which changes state whenever the data bit sequence does not.

Decoding: The act of translating an encoded set of bits to the original set of bits prior to coding.

De-serialization: Transformation of a serial bit stream into a sequence of control and/or data characters.

Destination: The node or unit that a packet is being sent to.

Destination Address: The route to be taken by a packet in moving from source to destination.

Destination List: A list of destination identifiers which forms the destination address of a packet.

Destination Identifier: The address, or partial address, of the packet destination.

Driver: An electronic circuit design to transmit signals across a particular transmission medium.

DS: Data-Strobe.

DS-DE: Data-Strobe, Differentially Ended. Used in IEEE-1355 [RD1] to indicate a link with differentially encoded data and strobe signals.

DSP: Digital Signal Processing.

ECL: Emitter-Coupled Logic.

EEP: Error End of Packet – used to indicate that an error occurred in the current packet.

EGSE: Electronic Ground Support Equipment.

EMC: Electro-Magnetic Compatibility.

EMI: Electro-Magnetic Interference.

End of Packet Marker: A control character which indicates the end of a packet.

EOP: End Of Packet market type one – used to indicate the normal end of a packet.

Error Recovery Scheme: A method for handling errors detected within a SpaceWire link.

ESC: Escape character is defined in the Character Level. ESC followed by FCT forms the NULL token.

Exchange Level: The protocol level which defines the mechanisms for link initialisation, flow control, error detection and error recovery.

FCT: Flow Control Token.

Filler: A cylindrical piece of PTFE used to fill the gap between insulated wires or cables being grouped together and formed into a larger cable. The filler enhances the structure of the cable helping to keep the constituent wires in a fixed position relative to one another.

Flow Control Token: A control character used to manage the flow of data across a link. Each flow control token indicates that there is space for 8 more normal-characters in the receiver buffer.

Host Receive Buffer: The buffer within a host system for receiving data from a link interface.

Host System: The system that a link interface is connected to.

Host Transmit Buffer: The buffer within a host system for holding data prior to transmission through a link interface.

Input Port: The receive side of a link interface on a routing switch.

Jitter: Random errors in the timing of a signal.

Lay Length: Length of lay (twists per foot). Lay length refers to the number of twists per foot and is expressed in inches or decimals as the length between one complete turn of a single end in the cable.

L-Char: See Link-character.

Link: A link represents the connection of one unit to another unit. A SpaceWire link is bi-directional. Each end contains a link transmitter and a link receiver. Data and control information is passed along a link from one end to the other.

Link-character: A control character used to manage the flow of data across a link (NULL and FCT).

Link Destination: Used to refer to the end of the link that is receiving a particular set of data or control information.

Link Interface: A SpaceWire interface comprising a transmitter which takes data from a host system and transmits it across a SpaceWire link, and a receiver which accepts data from a SpaceWire link and passes to the host system.

Link Receiver: Used to refer to the receiver at one end of a link.

Link Source: Used to refer to the end of the link that is sending a particular set of data or control information.

Link Transmitter: Used to refer to the transmitter at one end of a link.

Low Voltage Differential Signalling: A particular form of differential signalling using low voltage signals.

LVDS: Low Voltage Differential Signalling.

Mbps: Mega bits per second.

N-Char: See Normal-character.

Network: A set of units connected together via links and routing switches.

Network Level: The protocol level that defines the SpaceWire network routers and defines how packets of data are transferred across the network from source node to destination node.

Node: A source and/or destination of a packet. A node may be a processor, memory unit, sensor, EGSE or some other unit connected to a SpaceWire network.

Normal-character: A data character or control character that is passed from the exchange level to the packet level (EOP, EEP or Escape Sequence).

NULL: The NULL or token is sent to keep the data link active when there are no data or control characters to be sent.

Output Port: The transmit side of a link interface on a routing switch.

Packet: A sequence of normal-characters comprising a destination address, packet payload and an end of packet marker.

Packet Level: The protocol level that defines how data is organised in packets ready for transfer across a link or network.

Packet Cargo: The data that is to be transferred from a source to a destination.

PECL: Pseudo-ECL, ECL referenced to +5Volts.

PFA: A type of plastic used to cover wires in cables.

Physical Level: The protocol level that specifies the physical interconnection medium – cables and connectors.

PTFE: Polytetrafluroethylene. A type of plastic used to cover wires in cables.

Receiver: An electronic circuit design to receive signals sent across a particular transmission medium.

Router: Routing switch.

Routing Switch: A switch connecting several links that routes packets from one link to another. The destination address of each packet by the switch is used to determine which link a packet will be sent out on.

SCI: Scalable Coherent Interface is an IEEE standard (IEEE 1596) for connecting processors and peripherals in a high performance multiprocessor architecture.

Serialisation: Transformation of a sequence of control and/or data characters into a serial bit stream.

Signal: A measurable quantity that varies with time to transfer information. A signal propagates along a transmission medium.

Signal Level: The protocol level which defines the electrical signals used for SpaceWire together with the Data-Strobe encoding and signal timing.

Skew: The difference in time between the edges of two signals which should ideally be concurrent.

Source: The node or unit that is sending a packet.

TBC: To Be Confirmed

TBD: To Be Determined

Transmission Medium: The medium over which data is transferred – screened twisted pair cables.

Unit: A box, board or sub-system (that may have one or more SpaceWire interfaces).

3. OVERALL DESCRIPTION

This section provides an overview of the SpaceWire standard giving the rationale behind key decisions made in the development of the standard.

SpaceWire is based on the “DS-DE” part of the IEEE 1355-1995 standard [RD1] combined with the TIA/EIA-644 [AD1] and IEEE-1596.3 [RD2] Low Voltage Differential Signalling (LVDS) standards. See Annex C for details of the differences between SpaceWire and IEEE 1355 and the reasons for those differences.

SpaceWire is a full-duplex, bi-directional, serial, point-to-point data link. It encodes data using two differential signal pairs in each direction. That is a total of eight signal wires, four in each direction.

3.1 PHYSICAL LEVEL

The physical level of the SpaceWire standard covers cables, connectors and EMC specification.

3.1.1 Cables

The SpaceWire cable comprises four twisted pair wires with a separate shield around each twisted pair and an overall shield.

To achieve a high data signalling rate with SpaceWire over distances up to 10m the cable must have the following characteristics:-

- Characteristic impedance matched to the line termination impedance.
- Low signal-signal skew between each signal in a differential pair and between Data and Strobe pairs.
- Low signal attenuation.
- Low cross-talk.
- Good EMC performance.

3.1.2 Connectors

The SpaceWire connector is required to have eight signal contacts plus a screen termination contact. A nine pin micro-miniature D-type is specified as the SpaceWire connector. This type of connector is available qualified for space use.

3.1.3 EMC Specifications

The EMC specifications for SpaceWire have been derived from the EMC specifications for the Rosetta [RD13] and ENVISAT [RD14] missions. The EMC specification is given in Annex B. Initial EMC testing was performed by Patria Finavitec Oy with support from the University of Dundee. The testing covered:

- Radiated emission, electric and magnetic fields,
- Radiated susceptibility, electric and magnetic fields,
- Conducted susceptibility,
- Conducted emission,
- Electro-static discharge,
- Signalling rate,
- Bit error rate,
- Fault isolation, and

- ❑ Power consumption.

A summary of the EMC test results is provided in Annex B. Full details are reported in [RD15].

3.2 SIGNAL LEVEL

The signal level part of the SpaceWire standard covers signal voltage levels, noise margins and signal encoding.

3.2.1 Signal Level and Noise Margins

Low Voltage Differential Signalling or LVDS is specified as the signalling technique to be used in SpaceWire.

LVDS uses balanced signals to provide very high-speed interconnection using a low voltage swing (350 mV typical). The balanced or differential signalling provides adequate noise margin to enable low voltages to be used in practical systems. Low voltage swing means low power consumption at high speed. LVDS is appropriate for connections between boards in a unit, and unit to unit interconnections over distances of 10m or more.

A typical LVDS driver and receiver are shown in Figure 3-1, connected by a media (cable or PCB traces) with 100 ohm differential impedance.

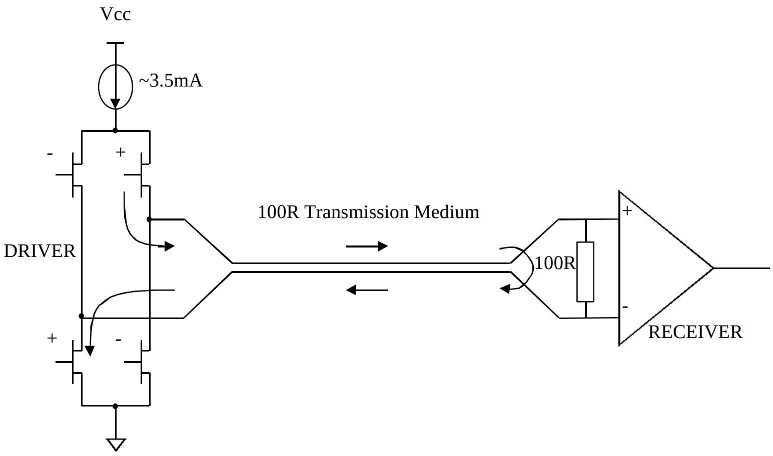


Figure 3-1 LVDS Operation

The LVDS driver uses current mode logic. A constant current source of around 3.5mA provides the current that flows out of the driver, along the transmission medium, through the 100-ohm termination resistance and back to the driver via the transmission medium. Two pairs of transistor switches in the driver control the direction of the current flow through the termination resistor. When the driver transistors marked “+” in Figure 3-1 are turned on and those marked “-” are turned off, current flows as indicated by the arrows on the diagram creating a positive voltage across the termination resistor. When the two driver transistors, marked “-”, are turned on and those marked “+” are turned off, current flows in the opposite direction producing a negative voltage across the termination resistor. LVDS receivers are specified to have high input impedance so that most of the current will flow through the termination resistor to generate around ±350mV with the nominal 3.5mA current source.

LVDS has several features that make it very attractive for data signalling [RD16]:-

- Near constant total drive current (+3.5mA for logic 1 and -3.5mA for logic 0) which decreases switching noise on power supplies.
- High immunity to ground potential difference between driver and receiver - LVDS can tolerate at least ±1V ground difference.
- High immunity to induced noise because of differential signaling normally using twisted-pair cable.
- Low EMI because small equal and opposite currents create small electromagnetic fields which tend to cancel one another out.
- Not dependent upon particular device supply voltage(s).
- Simple 100 ohm termination at receiver.
- Failsafe operation - the receiver output goes to the high state (inactive) whenever
  - the receiver is powered and the driver is not powered,
  - the inputs short together,
  - input wires are disconnected.
- Power consumption is typically 50mW per driver/receiver pair for LVDS compared to 120mW for ECL/PECL.

There are two standards, which define LVDS

1. TIA/EIA-644 that defines the driver output characteristics and the receiver input characteristics only [AD1].
2. IEEE 1596.3 Low Voltage Differential Signaling (LVDS) for Scalable Coherent Interface (SCI) that defines the signalling levels used and the encoding for packet switching used in SCI data transfers [RD2].

The signal levels and noise margins for SpaceWire are defined using the TIA/EIA-644 standard [AD1] since this deals with LVDS only whereas IEEE 1596.3 [RD1] is concerned with the use of LVDS specifically for SCI.

3.2.2 Data Encoding

SpaceWire uses Data-Strobe (DS) encoding. This is a coding scheme which encodes the transmission clock with the data into data and strobe so that the clock can be recovered by simply XORing the data and strobe lines together. The data values are transmitted directly and the strobe signal changes state whenever the data remains constant from one data bit interval to the next. This coding scheme is illustrated below in Figure 3-2. The DS encoding scheme is also used in the IEEE 1355-1995 [RD1] and IEEE 1394-1995 (Firewire) standard [RD17].

The reason for using DS encoding is to improve the skew tolerance to almost 1-bit time, compared to 0.5 bit time for simple data and clock encoding.

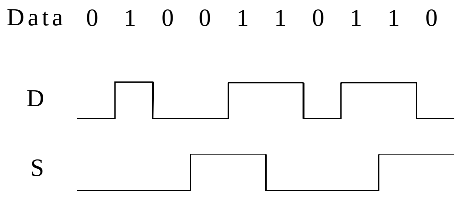


Figure 3-2 Data-Strobe (DS) Encoding

A SpaceWire link comprises two pairs of differential signals, one pair transmitting the D and S signals in one direction and the other pair transmitting D and S in the opposite direction. That is a total of eight wires for each bi-directional link.

3.3 CHARACTER LEVEL

SpaceWire employs the character level protocol defined in IEEE 1355-1995 [RD1].

There are two types of characters:-

- Data characters which hold an eight-bit data value, transmitted least-significant bit first. Each data character contains a parity-bit, a data-control flag and the eight-bits of data. The parity-bit covers the previous eight-bits of data, the current parity-bit and the current data-control flag. It is set to produce odd parity so that the total number of 1's in the field covered is an odd number. The data-control flag is set to zero to indicate that the current character is a data character.
- Control characters which hold a two-bit control code. Each control character is formed from a parity-bit, a data-control flag and the two-bit control code. The data-control flag is set to one to indicate that the current character is a control character. Parity coverage is similar to that for a data character. One of the four possible control characters is the escape code (ESC). This can be used to form longer control codes. One longer control code is specified which is the NULL code. NULL is formed from ESC followed by the flow control token (FCT). NULL is transmitted whenever a link is not sending data or control tokens to keep the link active and to support link disconnect detection.

The data and control characters are illustrated in Figure 3-3.

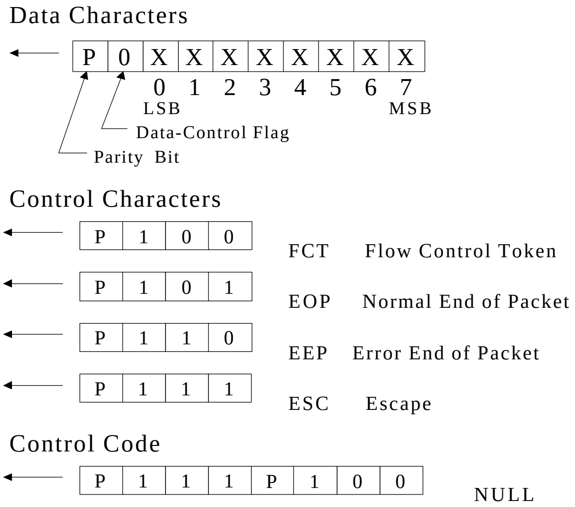


Figure 3-3 Data and Control Characters

3.4 EXCHANGE LEVEL

The exchange level protocol is a significantly improved version of that defined in IEEE 1355-1995 [RD1] which provides the following services:-

- **Initialisation:** Following reset the link output is held in the reset state until it is instructed to start and attempt to make a connection with the link interface at the other end of the link. A connection is made following a handshake that ensures both ends of the link are able to send and receive characters successfully. Each end of the link sends NULLs, waits to receive a NULL, then sends FCTs and waits to receive an FCT. Since a link interface cannot send FCTs until it has received a

NULL, receipt of one or more NULLs followed by receipt of an FCT means that the other end of the link has received NULLs successfully and that full connection has been achieved.

- **Flow Control:** A transmitter is only allowed to transmit data characters if there is space in the host system receive buffer for them. The host system indicates that there is space for eight more data characters by requesting the link transmitter to send a flow control token (FCT). The FCT is received at the other end of the link (end B) enabling the transmitter at end B to send up to eight more FCTs. If there is more room in the host receive buffer then multiple FCTs may be sent, one for every eight spaces in the receive buffer. Correspondingly, if multiple FCTs are received then it means that there is a corresponding amount of space available in the receiver buffer e.g. four FCTs means that there is room for 32 data characters.
- **Detection of Disconnect Errors:** Link disconnection is detected when following reception of a data bit no new data bit is received within a link disconnect timeout window (850 nsec). Once a disconnection error has been detected the link attempts to recover from the error (see below).
- **Detection of Parity Errors:** Parity errors occuring within a data or control character are detected when the next character is sent, since the parity bit for a data or control token is contained in the next character. Once a parity error has been detected the link will attempt to recover from the error (see below).
- **Link Error Recovery:** Following an error or reset the link attempts to re-synchronise and restart using an “exchange of silence” protocol (see Figure 3-4). The end of the link that is either reset or that finds an error, ceases transmission. This is detected at the other end of the link as a link disconnect and that end stops transmitting too. The first link resets its input and output for 6.4 μs to ensure that the other end will detect the disconnect. The other end will also wait for 6.4 μs after ceasing transmission. Each link then waits a further 12.8 μs before starting to transmit. These periods of time are sufficient to ensure that the receivers at both ends of the link are ready to receive characters before either end starts transmission. The two ends of the link go through the NULL/FCT handshake to re-establish a connection and ensure proper character synchronisation.

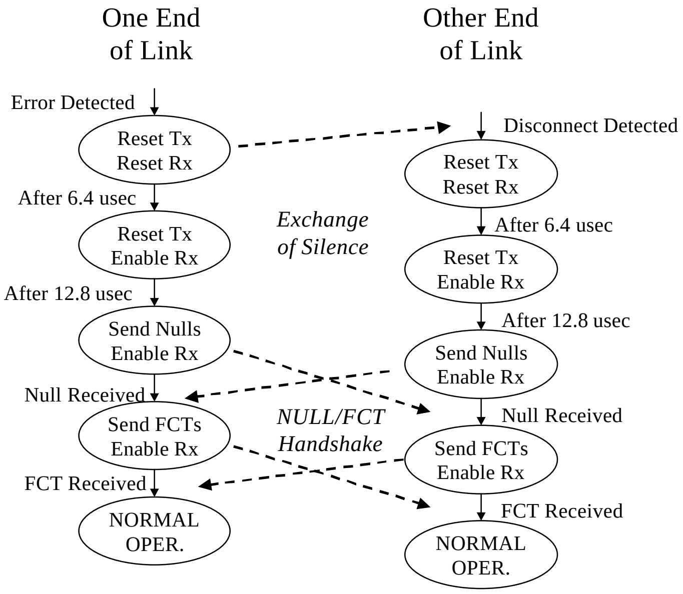


Figure 3-4 Link Restart

3.5 PACKET LEVEL

The packet level protocol follows the packet level protocol defined in IEEE-1355. It defines how data is encapsulated in packets for transfer from source to destination. The format of a packet is illustrated in Figure 3-5.

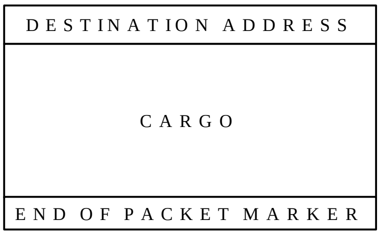


Figure 3-5 Packet Format

The "Destination Address" is a list of one or more data characters that represent the destination identity. This list of data characters represents either the identity code of the destination node or the path that the packet will take to get to the destination node.

The "Cargo" is the data to be transferred from source to destination.

The "End of Packet Marker" is used to indicate the end of a packet. Two end of packet markers are defined.

1. EOP Normal end_of_packet marker - indicates end of packet
2. EEP Error End_of_packet marker - indicates that the packet has been terminated prematurely due to a link error.

Since there is no start of packet marker, the first data character following an end_of_packet marker (either EOP or EEP) is regarded as the start of the next packet.

The packet level protocol provides support for packet routing via wormhole routing switches [RD18].

3.6 NETWORK LEVEL

The network level defines what a SpaceWire network is, describes the components that make up a SpaceWire network, explains how packets are transferred across a SpaceWire network and details the manner in which the SpaceWire network recovers from errors.

A SpaceWire network is made up of a number of SpaceWire nodes interconnected by SpaceWire routing switches. SpaceWire nodes are the sources and destination of packets and provide the interface to the application system(s). SpaceWire nodes may be directly connected together using SpaceWire links or they may be interconnected via SpaceWire routing switches using SpaceWire links to make the connection between node and routing switch. A SpaceWire routing switch has several link interfaces which are connected together inside the routing switch by a switch matrix which allows any link input to pass the packets that it receives on to any link output for re-transmission.

3.7 APPLICATION INTERFACE

The application interface is not defined in the SpaceWire standard. However, a typical application interface will comprise the following services:-

- Open Link – Starts a link interface and attempts to establish a connection with the link interface at the other end of the link.
- Close Link – Stops a link and breaks the connection.
- Write Packet – Sends a packet out of the link interface.
- Read Packet – Reads a packet from the link interface.

- Status and Configuration – Reads the current status of the link interface and sets the link configuration.

4. PHYSICAL LEVEL

The physical level provides the actual interface between nodes including both the mechanical and electrical interface. This level of the standard covers:-

- Cable construction
- Connectors
- Cable assemblies, and
- PCB / backplane tracking.

4.1 CABLES

The SpaceWire cable shall be constructed according to ESA/SCC Generic Specification No. 3902 [AD2] and the specific details given below.

The SpaceWire cable shall comprise four twisted pair wires with a separate shield around each twisted pair and an overall shield as illustrated in Figure 4-1.

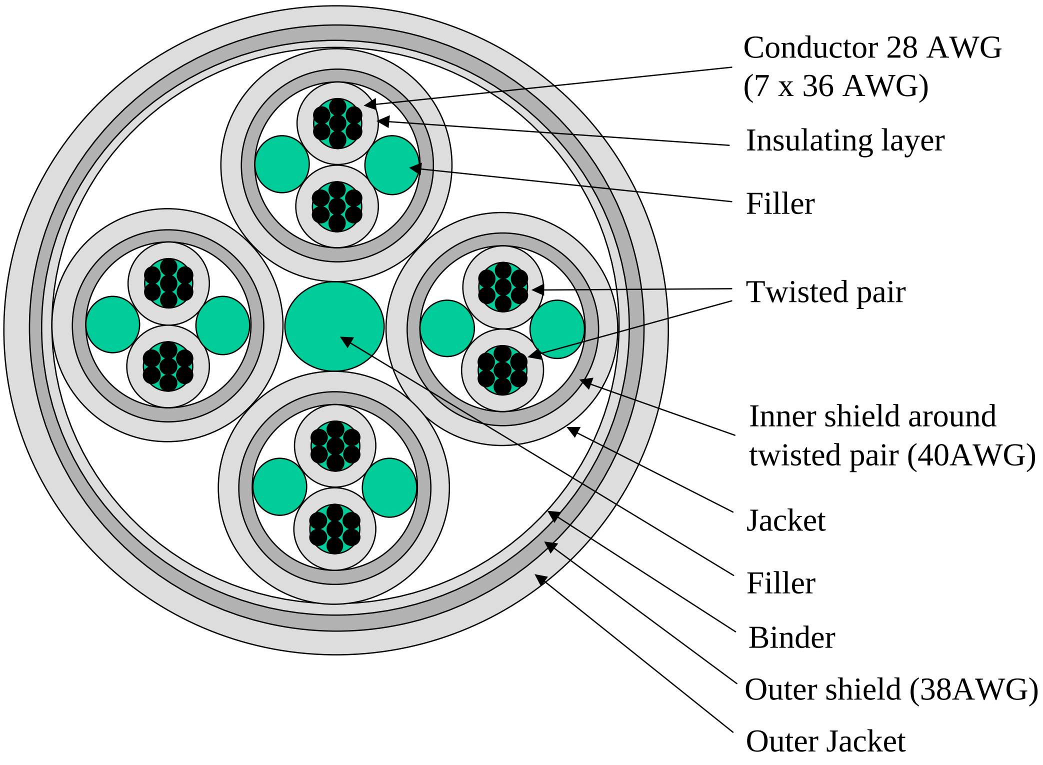


Figure 4-1 SpaceWire Cable Construction

4.1.1 Inner Conductors

4.1.1.1 Conductor

Each signal wire shall be 28 AWG, constructed from seven strands of 36 AWG silver-coated, high-strength copper alloy. The thickness of the silver coating shall be 2.0 m minimum.

4.1.1.2 Tensile Characteristics

The minimum elongation of each strand shall be 6.0%.

The tensile strength of each strand shall be at least 35kg/mm^2.

4.1.1.3 Insulator

Each signal shall be insulated using expanded, microporous PTFE with only those additives necessary for processing and pigmentation.

4.1.1.4 Insulator Colour

The insulator around the signal wires shall be white.

4.1.1.5 Electrical Characteristics

The maximum D.C. resistance of the inner conductor shall be 256 ohm/km (TBC).

4.1.2 Twisted Pair

4.1.2.1 Lay Length

The lay of length of the two insulated conductors comprising a differential signal pair shall not be less than 12 times and not more than 16 times the outside diameter of the unshielded twisted pair.

4.1.2.2 Fillers

Fillers shall be used with the differential signal pairs so as to ensure a smooth and uniform diameter under the shielding in order to contribute to a uniform impedance over the cable.

4.1.2.3 Filler Material

The filler material as used for the differential signal pairs shall be expanded microporous PTFE with only those additives necessary for processing.

4.1.2.4 Construction of Filler

The filler material shall be extruded or wrapped from tapes to a diameter of 1.0mm TBC.

4.1.2.5 Shield

Each differential signal pair shall be shielded by a braided shield. The braided shield type shall be of push-back type and provide not less than 90% coverage.

4.1.2.6 Shield Wire Size

The shield wire size shall be 40 AWG.

4.1.2.7 Shield Material

All strands used in the manufacture of the braided shield shall be silver-coated, soft or annealed oxygen-free high conductivity copper. The thickness of silver shall be 2.5 microns minimum. Any strand shall show an elongation of 10% minimum.

4.1.2.8 Protective Sheath

The protective sheath for the shielded differential signal pairs shall be a layer of extruded fluorpolymer PFA with only those additives necessary for processing and pigmentation.

4.1.2.9 Protective Sheath Wall Thickness

The wall thickness of the protective sheath for the shielded differential signal pair shall be 0.15mm nominal.

4.1.2.10 Protective Sheath Colour

The jacket colour of the differential signal pairs shall be white.

4.1.2.11 Characteristic Impedance

The characteristic impedance of each differential signal pair shall be 100 ± 6 ohm differential impedance.

4.1.2.12 Skew

The skew between each signal in each differential signal pair shall be less than 0.1nsec/m (TBC).

4.1.3 Complete Cable

4.1.3.1 Construction

Four sets of differential signal pairs shall be twisted together not less than 12 times and not more than 16 times of the outside diameter of a shielded and jacketed differential signal pair.

4.1.3.2 Filler

A filler shall be used in the centre of the four differential signal pairs so as to ensure a smooth and uniform diameter under the shielding in order to contribute to a uniform impedance over the cable.

4.1.3.3 Filler Material

The filler material as used for the differential signal pairs shall be expanded microporous PTFE with only those additives necessary for processing.

4.1.3.4 Construction of Filler

The filler material shall be extruded or wrapped from tapes to a diameters of TBD mm.

4.1.3.5 Binder

A binder shall be applied over the four differential signal pairs and central filler to keep the signal pairs and filler together in a fixed position.

4.1.3.6 Binder Material

The material shall be virgin, wrapped, expanded microporous PTFE with only those additives necessary for processing.

4.1.3.7 Binder Construction

The material shall be wrapped with an overlap of 50% maximum.

4.1.3.8 Outer Shield

The set of four jacketed and screened differential signal pairs shall be shielded by an outer braided shield. The braided shield type shall be of push-back type and provide not less than 90% coverage.

4.1.3.9 Outer Shield Wire Size

The shield wire size shall be 38 AWG.

4.1.3.10 Outer Shield Material

All strands used in the manufacture of the braided shield shall be silver-coated, soft or annealed oxygen-free high conductivity copper. The thickness of silver shall be 2.5 microns minimum. Any strand shall show an elongation of 10% minimum.

4.1.3.11 Shield Isolation

The twisted pair shields shall NOT make contact with one another NOR with the outer shield.

4.1.3.12 Outer Jacket

The outermost jacket over the four twisted screened and jacketed differential signal pairs shall be a layer of extruded Fluoropolymer PFA with only those additives necessary for processing and pigmentation.

4.1.3.13 Outer Jacket Wallthickness

The wall thickness of the jacket for the shielded differential signal pair shall be 0.25mm nominal.

4.1.3.14 Jacket Colour

The jacket shall be white in colour.

There shall be NO identifying marking on the cable jacket. Applying pressure to the cable during the marking process can adversely affect the electrical properties of the cable.

4.1.3.15 Signal Skew

The skew between the differential signal in one differential signal pair and the differential signal in each other differential signal pair within the cable shall be less than 0.1nsec/m (TBC).

4.1.4 Cable Physical Parameters

4.1.4.1 Cable Diameter

The outside diameter of the complete cable shall be less than 7mm (TBC).

4.1.4.2 Cable Minimum Bend Radius

The minimum bend radius of complete cable shall be less than 50mm (TBC).

4.1.4.3 Adhesion of Inner Conductor

The minimum stripping force shall be 1.0 N.

4.1.4.4 Cable Weight

The maximum weight of the SpaceWire cable shall be 60 g/m (TBC).

4.1.4.5 Cable Maximum Ratings

The maximum ratings defined in Table 4-1 shall be met.

Table 4-1 SpaceWire Cable Maximum Ratings

No.	Characteristics	Symbol	Maximum Ratings	Unit	Remarks
1	Operating Voltage (Continuous)	V_{op}	200	V_{rms}	
2	Current	I	1.5	A	
3	Operating Frequency	F_M	400	Mbits/ sec	
4	Operating Temperature Range	T_{op}	-200 to +180	C	T_{amb} NOTE 1
5	Storage Temperature Range	T_{stg}	-200 to +180	C	
NOTE 1: The above specified current will generate a temperature rise of approximately 50 C above ambient temperature in a vacuum environment. Precautions shall be taken to prevent the total temperature of the wire (ambient plus rise) exceeding the continuous operating temperature of the wire.					

4.2 CONNECTORS

The SpaceWire connector shall be a nine contact micro-miniature D-type as defined in ESA/SCC Detailed Specification No. 3401/029 [AD3].

4.2.1 Sockets

Sockets shall be used on board and unit assemblies.

The socket used on board and unit assemblies should be of type 340102901B 9SFR114 as defined in ESA/SCC Detailed Specification No. 3401/029 [AD3]. Note: These sockets with flying leads are recommended for connection to a PCB rather than PCB mounting connectors to improve mechanical shock and vibration resistance of the unit.

4.2.2 Plugs

Plugs shall be used on cable assemblies.

The plugs used on cable assemblies should be of type 340102901B 9P (as defined in ESA/SCC Detailed Specification No. 3401/029 [AD3]) with the SpaceWire conductors directly crimped to the crimp-type contacts before assembly and potting of the plug (see section 4.3 for SpaceWire cable assembly details).

4.2.3 Connector Pin-Out

The connector pin-out given in Table 4-2 and Figure 4-2 shall be used.

Table 4-2 Connector Pin-Out

Pin Number	Signal Name
1	DIN+
2	SIN+
3	Inner Shield
4	SOUT-
5	DOUT-
6	DIN-
7	SIN-
8	SOUT+
9	DOUT+

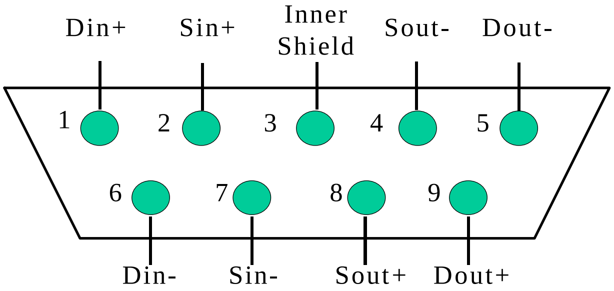


Figure 4-2 SpaceWire Connector Pin-Out

The inner shield connection is for connection to the inner shield of the SpaceWire cable. This should be connected to signal ground (possibly via a parallel resistor and capacitor) according to the EMC requirements or guideline for a particular mission. See section 4.3 for the cable connection to pin 3 of the connector.

4.2.4 Flying Lead Connectors

Flying lead connectors are recommended for connection to a PCB. Flying lead connectors used for connection to a PCB should have all the leads cropped to the same short length (less than 25mm) and the wires comprising the differential signal pairs should be twisted together. This will help to minimise the discontinuity in impedance caused by the connector.

4.2.5 PCB Mounting Connectors

PCB mounting right-angled connectors are not recommended. If a PCB mounting right-angled connector has to be used care must be taken to compensate in the PCB layout for the different lengths of signal path though the connector. The topmost row of pins on the right-angled connect have longer leads than the bottom row. Signals connected to the top row must be given correspondingly shorter PCB track lengths than tracks going to the bottom row. Track length compensation must be performed at the connector end of the PCB tracks to maintain the differential signal across the PCB.

4.3 CABLE ASSEMBLY

Cable assemblies shall consist of two identical plug connectors joined by a length of cable.

4.3.1 Cable Length

The maximum length of the cable assembly should normally be 10m to ensure that the end to end skew and jitter introduced by the cable assembly does not exceed the maximum budget for the cable. Longer length cables may be used at slow data signalling rates provided that the signal attenuation (see section 5.4) and system jitter and skew limits are not violated at the operating data signalling rate (see section 5.6.4).

4.3.2 Cable Connections

The connector contacts shall be terminated as shown in Figure 4-3. The cable signal wires cross over to achieve a transmit to receive interconnection, i.e. DOUT+ is connected to DIN+ etc.

The individual shields of the differential signal pairs carrying the output signals DOUT+, DOUT- and SOUT+ and SOUT- shall be connected together and to pin 3 of the connector. The shields are terminated at the end of the cable that the signals are being driven, following good EMC practice. In this way two of the differential pairs are connected at one end of the cable and the remaining two at the other end. A symmetrical arrangement results, avoiding the problem of having to know which end of the cable is which during installation.

A metal shell shall be used for each connector to provide necessary shielding of the connector.

The outer shield of the cable shall be bonded to the connector shell via a low impedance connection.

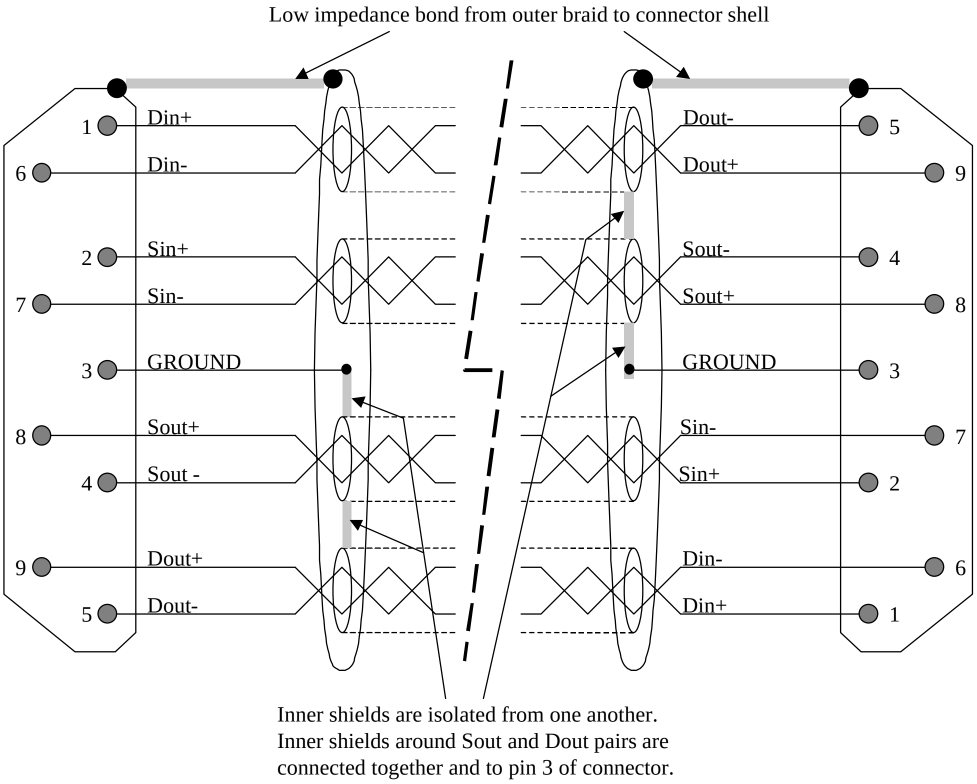

Low impedance bond from outer braid to connector shell
1
Din+
6
Din-
2
Sin+
7
Sin-
3
GROUND
8
Sout+
4
Sout -
9
Dout+
5
Dout-
Dout-
5
Dout+
9
Sout-
4
Sout+
8
GROUND
3
Sin-
7
Sin+
2
Din-
6
Din+
1
Inner shields are isolated from one another.
Inner shields around Sout and Dout pairs are
connected together and to pin 3 of connector.

Figure 4-3 SpaceWire Cable Assembly

4.3.3 Connector Back Shell

TBD

4.4 PCB/BACKPLANE LINK

As well as routing SpaceWire signals through a cable the signals may also be transmitted across a PCB or along a backplane.

4.4.1 Differential Signal Pairs

4.4.1.1 Differential Impedance

Differential pair signals shall run on a pair of close, parallel PCB tracks with a differential impedance of 100 ± 6 ohms. The required differential impedance may be achieved by adjusting the track thickness, width, separation and height above the ground plane.

4.4.1.2 Difference in Track Length

The difference in track length between the two signals from a differential pair shall be less than 5% of the track length and no more than 5mm. This is to avoid skew between the two parts of the differential signal.

4.4.2 Differential Signals

4.4.2.1 Skew

The skew introduced between the data and strobe (D and S) signals must be minimised. For PCB tracks this is controlled by making the tracks all close to the same length. The difference in track length between the data and strobe signals shall be less than 5% of the track length and no more than 5mm.

4.4.3 Example Track Dimensions

4.4.3.1 Microstrip

TBA

4.4.3.2 Stripline

TBA

5. SIGNAL LEVEL

5.1 LVDS

SpaceWire shall use Low Voltage Differential Signalling (LVDS) with electrical characteristics as defined in the TIA/EIA-644 standard “Electrical Characteristics of Low Voltage Differential Signalling (LVDS) Interface Circuits” [AD1].

The signalling levels used by LVDS are illustrated in Figure 5-1.

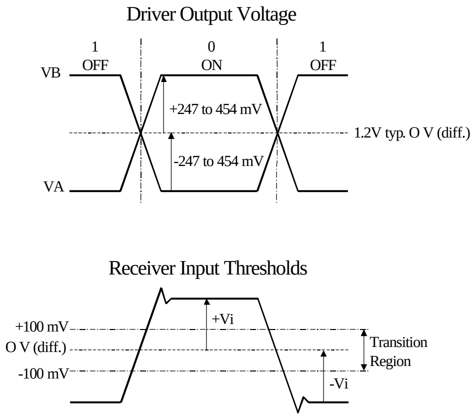


Figure 5-1 LVDS Signalling Levels

5.2 FAILSAFE OPERATION OF LVDS

When any of the following fault conditions occur the receiver outputs shall not oscillate and shall be locked to logic high provided that a noise threshold of 10 mV (TBC) is not exceeded at the receiver input.

1. Driver not powered
2. Driver disabled
3. Driver not connected to receiver
4. Receiver inputs open circuit (i.e. cable or wire in cable disconnected)
5. Receiver inputs are shorted together.

When the driver is not powered its output should be high impedance i.e. > TBD kohm.

When the receiver is not powered its input should be high impedance i.e. > TBD kohm.

5.3 SIGNAL CODING

5.3.1 DS

SpaceWire shall use Data-Strobe (DS) encoding as defined in section 5.3.5 of IEEE 1355-1995 [RD1]. Note DS encoding is also defined in IEEE 1394-1996 [RD17]. See Annex C for details of the differences between SpaceWire and IEEE 1355 and the reasons for those differences.

The data bit stream to be transmitted shall be encoded using two signals Data and Strobe. The Data signal shall follow the required data bit stream i.e. be high when the data bit is 1 and low when the data bit is 0. The Strobe signal shall change state whenever the Data does not change from one bit to the next.

The DS encoding is illustrated in Figure 5-2.

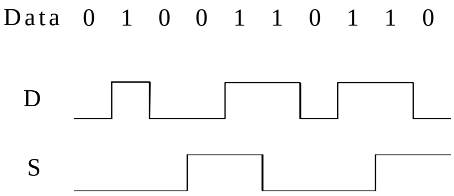


Figure 5-2 Data-Strobe (DS) Encoding

5.3.2 Simultaneous Transition on Data and Strobe Signals

Simultaneous transitions on the data and strobe lines are not part of the normal operation of SpaceWire. They can occur, however, either when a SpaceWire cable is plugged in while the transmitter is trying to make a connection, or when the LVDS driver or receiver circuits are enabled while the transmitter is trying to make a connection.

The SpaceWire receiver shall be tolerant of simultaneous transitions on the data and strobe lines i.e. the receiver shall not hang-up. Data corruption following simultaneous transitions on the data and strobe lines is to be expected.

When the SpaceWire transmitter is reset it shall be a controlled reset avoiding simultaneous transitions of data and strobe signals. For example, after stopping transmission the strobe signal may be reset first followed by the data signal.

5.4 DIFFERENTIAL DS

SpaceWire shall use low voltage differential signalling (LVDS) for the data and strobe signals.

5.5 SPACEWIRE LINK

A SpaceWire link shall comprise two pairs of differential signals, one pair transmitting the D and S signals in one direction and the other pair transmitting D and S in the opposite direction. That is a total of eight wires for each bi-directional link.

5.6 DATA SIGNALLING RATE

5.6.1 Minimum Data Signalling Rate

The minimum data signalling rate is the lowest data signalling rate at which a SpaceWire link can operate. The minimum data signalling rate is set by the disconnect time-out (section 7.10.1) to greater than 1.18 Mbps, i.e. 1/(850 nsec). The minimum data signalling rate at which a SpaceWire link shall operate is 2 Mbps.

5.6.2 Maximum Data Signalling Rate

The maximum data signalling rate is the highest data signalling rate at which a SpaceWire link can operate and is defined by consideration of signal skew and jitter (see section 5.6.4).

5.6.3 Operational Data Signalling Rate

A SpaceWire link may operate at any data signalling rate between the minimum data signalling rate and the maximum possible data signalling rate.

The link in one direction may operate at a different data signalling rate to the same link in the opposite direction. Links within a system may operate at different data signalling rates.

5.6.4 Effects of Skew and Jitter

The maximum data signalling rate that can be achieved will be different from one system to another depending on cable length, driver/receiver technology, encoder/decoder design etc, and is limiter by skew and jitter. Figure 5-3 illustrates the effect of skew and jitter on the data and strobe signals.

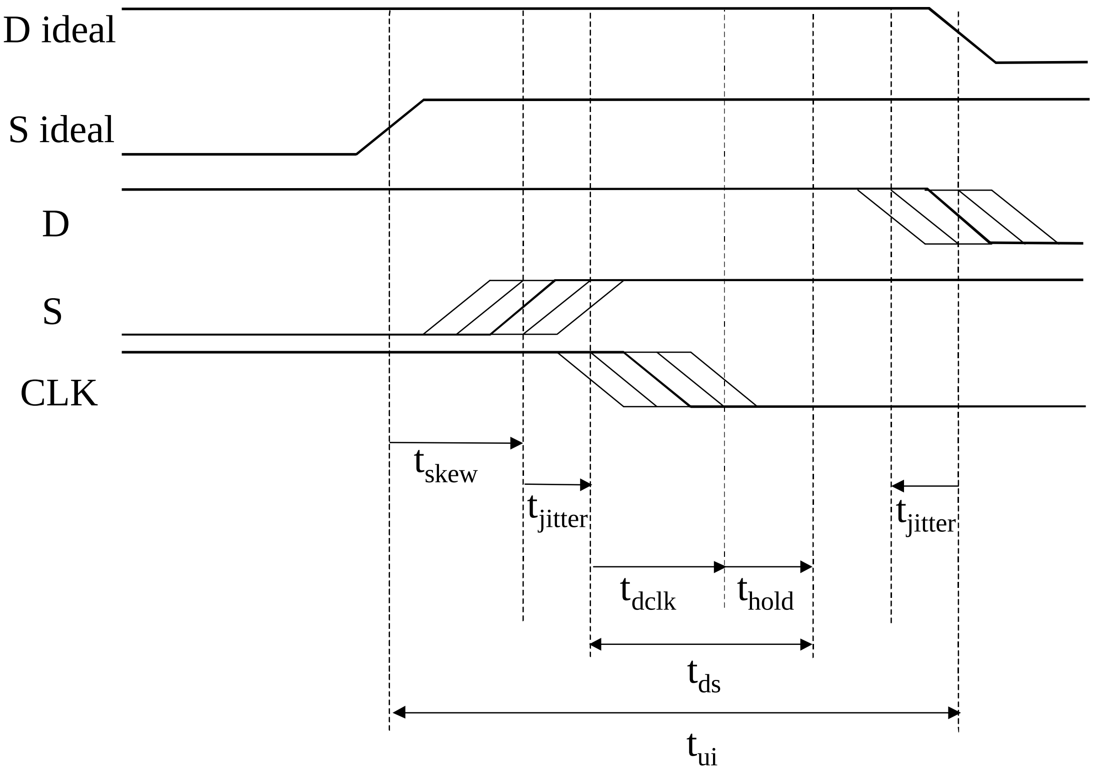


Figure 5-3 Skew and Jitter

t_{skew} is the skew between the data and strobe signals.

t_{jitter} is the jitter on the data or strobe signal. t_{jitter}data = t_{jitter}strobe since they follow identical signal paths (as close as possible).

t_{dclk} is the delay in the receiver from the edge of the data or strobe signal, through the XOR operation which produces the clock signal, to the clocking in of the data in the input flip-flop. This may be regarded as the set-up time for the data input flip-flop from the edge of the data or strobe signal.

t_{hold} is the hold time required for the data signal after the clocking of the data into the input flip-flop.

t_{ui} is the unit interval or bit period. $t_{ui} = 1/F_{op}$ where F_{op} is the link operating frequency.

The t_{dclk} and t_{hold} parameters may be combined into a minimum specification for the separation of consecutive edges on the data and strobe signals at the input to the decoder, $t_{ds} = t_{dclk} + t_{hold.}$

t_{margin} is the available margin. $t_{margin} = t_{ui} – (t_{skew} + 2*t_{jitter} + t_{dclk} + t_{hold})$.

For reliable operation the margin shall be greater than zero.

Figure 5-4 illustrates the contributors to skew and jitter in a typical system.

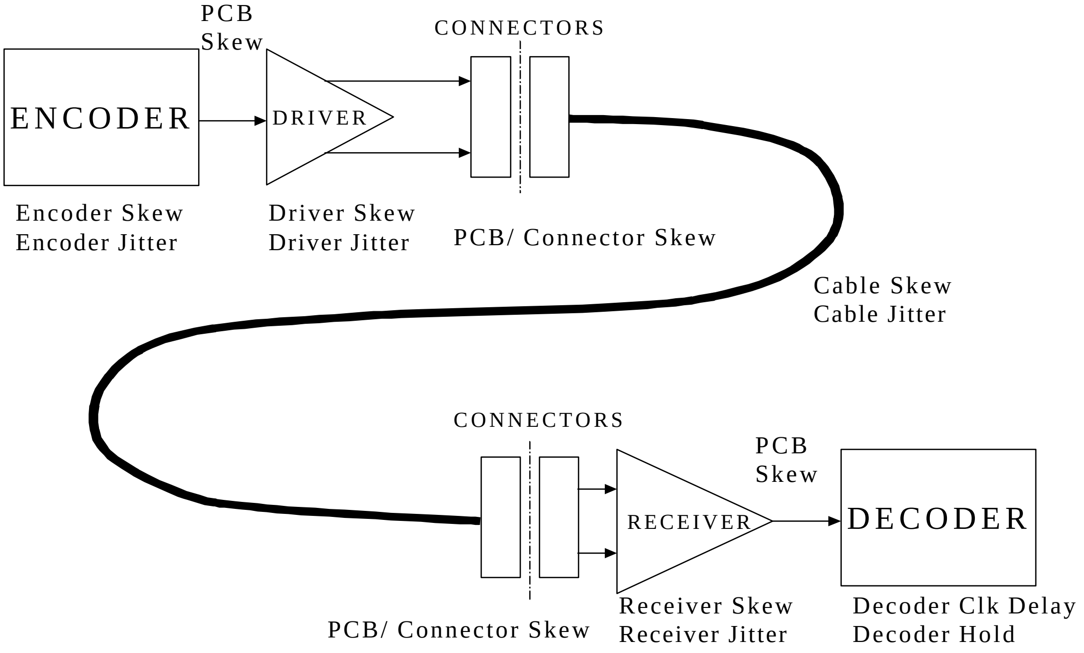


Figure 5-4 Contributors to Skew and Jitter

The following three tables (Table 5-1, Table 5-2 and Table 5-3) provide the jitter and skew budgets at three different operating frequencies used as examples (100 Mbps, 200 Mbps and 400 Mbps). The table for 400 Mbps operation contains suggested jitter and skew budgets for future systems and is provided for information only.

FIGURES IN THE FOLLOWING TABLES THAT ARE ITALIC AND UNDERLINED ARE TBC.

Table 5-1 Example Jitter and Skew Budget at 100 Mbps (tui = 10 ns)

	Data Jitter (ns)	Strobe Jitter (ns)	Skew (ns)	Min edge separation (ns)	Total (ns)
	t jitter	t jitter	t skew	t ds	
Encoder Skew			0.50		
Encoder Jitter	*0.50*	*0.50*			
PCB Skew			0.05		
Driver Skew			1.00		
Driver Jitter	*0.50*	*0.50*			
PCB/ Connector Skew			0.10		
Total Transmitter	**1.00**	**1.00**	**1.65**		**3.65**
Cable Jitter	*0.50*	*0.50*			
Cable Skew			1.00		
Total Cable	**0.50**	**0.50**	**1.00**		**2.00**
PCB/ Connector Skew			0.10		
Receiver Skew			1.50		
Receiver Jitter	*0.50*	*0.50*			
PCB Skew			0.05		
Decoder Clock Delay and Hold				1.00	
Total Receiver	**0.50**	**0.50**	**1.65**	**1.00**	**3.65**
Total System	**2.00**	**2.00**	**4.30**		**8.30**
Margin					1.70

Table 5-2 Example Jitter and Skew Budget at 200 Mbps (tui = 5 ns)

	Data Jitter (ns)	Strobe Jitter (ns)	Skew (ns)	Min edge separation (ns)	Total (ns)
	t jitter	t jitter	t skew	t ds	
Encoder Skew			0.50		
Encoder Jitter	*0.10*	*0.10*			
PCB Skew			0.05		
Driver Skew			0.07		
Driver Jitter	*0.20*	*0.20*			
PCB/ Connector Skew			0.10		
Total Transmitter	**0.30**	**0.30**	**0.72**		**1.32**
Cable Jitter	*0.50*	*0.50*			
Cable Skew			1.00		
Total Cable	**0.50**	**0.50**	**1.00**		**2.00**
PCB/ Connector Skew			0.10		
Receiver Skew			0.12		
Receiver Jitter	*0.20*	*0.20*			
PCB Skew			0.05		
Decoder Clock Delay and Hold				1.00	
Total Receiver	**0.20**	**0.20**	**0.27**	**1.00**	**1.67**
Total System	**1.00**	**1.00**	**1.99**	**1.00**	**4.99**
Margin					0.01

Table 5-3 Example Jitter and Skew Budgets for 400 Mbps Operation (tui = 2.5 ns)

	Data Jitter (ns)	Strobe Jitter (ns)	Skew (ns)	Min Edge Separation (ns)	Total (ns)
	t jitter	**t jitter**	**t skew**	**t ds**	
Encoder Skew			*0.20*		
Encoder Jitter	*0.10*	*0.10*			
PCB/ Connector Skew			0.05		
Total Transmitter	**0.10**	**0.10**	**0.25**		**0.45**
Cable Jitter	*0.35*	*0.35*			
Cable Skew (5m max. length)			0.50		
Total Cable	**0.35**	**0.35**	**0.50**		**1.20**
PCB/ Connector Skew			0.05		
Receiver Jitter	*0.10*	*0.10*			
Decoder Clock Delay and Hold				*0.50*	
Total Receiver	**0.10**	**0.10**	**0.05**	**0.50**	**0.75**
Total System	**0.55**	**0.55**	**0.80**	**0.50**	**2.40**
Margin					0.10

NOTE: The jitter and skew figures for 400 Mbps operation (Table 5-3) assume that the LVDS driver / receiver are integrated in the same package as the encoder / decoder.

The maximum data signalling rate for a SpaceWire link must be set so that the timing margin (t_{margin}) is greater than zero.

5.6.5 Initial Operating Data Signalling Rate

After reset the SpaceWire link transmitter shall initially commence operating at a data signalling rate of 10 ± 2 Mbps. It shall operate at this rate until commanded to operate at a different data signalling rate. This requirement is intended to provide all systems with a common, slow, initial data signalling rate so that system operation can be validated before switching to higher and possibly widely different data signalling rates. All SpaceWire systems shall be capable of this initial slow data signalling rate but may not necessarily be capable of higher data signalling rates. This minimum data signalling rate constrains the speed at which a link may reconnect either initially or after a disconnect (see section 7).

5.6.6 Altering Data signalling rate

The transmitter operating rate shall only be changed after the link connection has been made fully (i.e. the exchange-level state machine is in the *Run* state – see section 7).

6. CHARACTER LEVEL

The character level protocol follows the DS-SE and DS-DE character level encoding given in IEEE Std 1355-1995 [RD1].

6.1 DATA

A data character shall contain a parity-bit, a data-control flag and eight-bits of data. The data-control flag shall be set to zero to indicate that the current character is a data character. The eight-bit data value shall be transmitted least-significant bit first. This is illustrated in Figure 6-1.

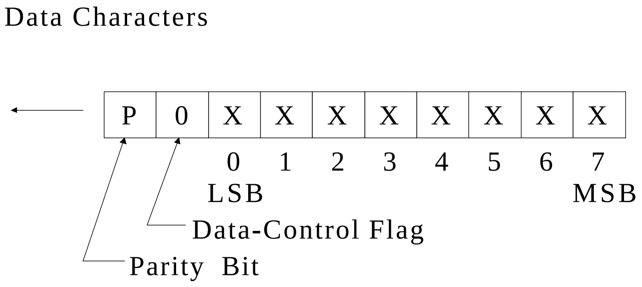


Figure 6-1 SpaceWire Data Characters

6.2 CONTROL

A control character shall be formed from a parity-bit, a data-control flag and a two-bit control code. The data-control flag shall be set to one to indicate that the current character is a control character. The different control characters are illustrated in Figure 6-2.

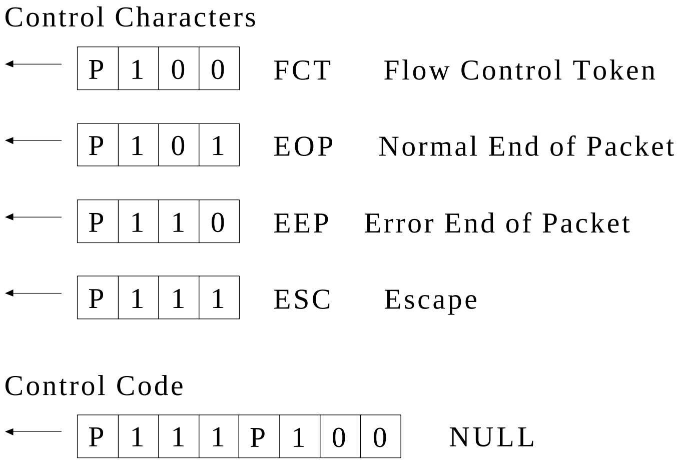


Figure 6-2 SpaceWire Control Characters and NULL Control Code

One of the four possible control characters is the escape code (ESC).

The NULL control code shall be formed from ESC followed by the flow control token (FCT). NULL shall be transmitted whenever a link is not sending data or control tokens, to keep the link active and to support link disconnect detection (see section 7).

The ESC character shall only be used to form the NULL control code. Any other use of ESC is invalid. An escape character (ESC) followed by any character other than the flow control token (FCT) is an invalid sequence and shall be noted as a receive error.

6.3 PARITY FOR ERROR DETECTION

A parity bit shall be assigned to each data or control character to support the detection of transmission errors.

The parity-bit shall cover the previous eight bits of a data character or two bits of a control character, and the current data-control flag. It shall be set to produce odd parity so that the total number of 1's in the field covered is an odd number. The coverage of the parity bit is illustrated in Figure 6-3.

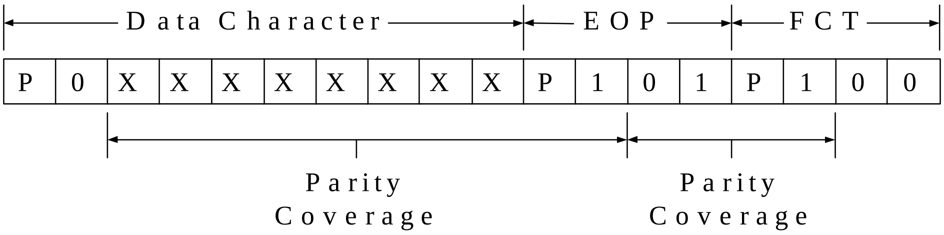


Figure 6-3 Parity Coverage

After reset or link error (while in the ErrorReset state – see section 7) the data and strobe signals shall be set to zero. When the transmitter is enabled after reset the first bit to be sent shall be a parity bit, this bit shall be set to zero so that the first transition shall be on the strobe line. This results in the patterns shown in figure 6-4 appearing when a link is started.

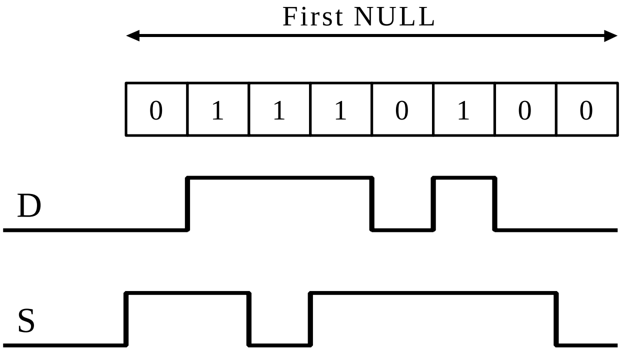


Figure 6-4 Data and Strobe Signals on Link Start

6.4 HOST INTERFACE TO TRANSMITTER AND RECEIVER

When the transmit and receive data interfaces to the host comprise 8 data-bits and 1 control-flag, the coding given in table 6-1 shall be used.

Table 6-1: Transmit and Receiver Host Data Interface Coding.

Control Flag	**Data Bits (MSB … LSB)**	**Meaning**
0	xxxxxxxx	8-bit data
1	xxxxxxx0 (use 00000000)	EOP
1	xxxxxxx1 (use 00000001)	EEP

For the two control codes only the significant bit is decoded. When writing to the transmit interface the remaining data bits should be set to zero. The receiver should set the seven most significant data bits to zero when the control bit is set.

7. EXCHANGE LEVEL

The exchange level protocol is an improved version of the DS-SE and DS-DE exchange level protocol given in section 5.7 of the IEEE Standard 1355-1995 [RD1]. The Exchange level state machine has been modified to eliminate problems with the *ResetLinkCommand* and several ambiguities within the IEEE 1355-1995 standard have been resolved. See Annex C for details of the differences between SpaceWire and IEEE 1355 and the reasons for those differences.

The exchange level is responsible for making a connection across a link and for managing the flow of data across the link.

7.1 LINK-CHARACTERS AND NORMAL-CHARACTERS (NORMATIVE)

At the exchange level, data and control characters are separated into two types L-Char and N-Char.

Link-characters (L-char) are those that are used in the exchange level and which are not passed on to the packet level. The flow control token (FCT) character and escape (ESC) character are Link-characters.

Normal-characters (N-char) are all the other characters: data characters and end-of-packet markers (EOP and EEP). Normal characters are all passed up to the packet level.

Only normal-characters shall be passed from the host interface to the link for transmission. The link interleaves link-characters and normal-characters during transmission but passes only normal-characters on to the host interface on the receiving side.

Normal-characters from one packet shall NOT be interleaved with Normal-characters from another packet.

A received character shall not be acted upon until its parity has been checked.

7.2 FLOW CONTROL (NORMATIVE)

The flow of data across a link must be controlled to avoid host receive buffer overflow and subsequent loss of data. Data flow shall be controlled using flow control tokens sent by the receiver to the transmitter. The flow control token (FCT) is defined in section 6.2.

Each flow control token sent out by a link interface shall indicate that there is space for eight more Normal-characters in the host receive buffer. A host system with a large buffer should ask the link interface to send out multiple FCTs according to the amount of space in the buffer. For each FCT sent the host system shall reserve room in its receive buffer to accommodate eight Normal-characters.

The transmitter shall not transmit any normal-characters until it has received one or more FCTs to indicate that the receiver is ready. The transmitter shall keep a credit count of the number of Normal-characters that it has been authorised to send. Each time a link interface receives an FCT its transmitter shall increment the credit count by eight. Whenever the transmitter sends an N-Char it shall decrement the credit count by one. If the credit count reaches zero the transmitter shall cease sending Normal-characters until it receives another FCT increasing the credit count again to eight. When the credit count is zero the transmitter shall continue to send Link-characters (NULLs or FCTs).

After a reset the initial value of the credit count shall be zero.

If the credit count is at its maximum value when an FCT is received that FCT shall be ignored.

If an FCT is received when the credit count is close to its maximum value (i.e. within eight of the maximum value) then the credit count shall not be incremented and the FCT shall be ignored.

The credit count maximum shall be at least 56.

Consider the following situation. A link interface (end A) has a credit count of a limited size (e.g. max. credit count is 8x7 = 56, allowing for seven FCTs). The host system at the other end of the link (end B) has a larger buffer than can be handled by the limited credit count size (e.g. host receive buffer greater than 8*7 = 56). In this case any additional FCTs that are sent by end B to end A will be ignored. Thus if an FCT is received when the credit count is at its maximum value the credit count will not be incremented. The system will operate as if the host receive buffer size is eight times the maximum credit count (e.g. 8*7 =56).

If transmission of an FCT is requested then it shall be sent immediately, as soon as the transmitter has finished sending the current character. When no FCT is requested and normal-characters are available from the host interface and the flow control credit count is above zero, the transmitter shall send normal-characters. If no FCTs or normal characters have to be sent then the transmitter shall send NULLs to indicate that the link is still active (and to prevent the disconnect detection mechanism from being triggered at the other end of the link).

The order of priority for transmission of characters shall be as follows:-

1. FCTs – highest priority
2. Normal characters
3. NULLs – lowest priority

7.3 ENCODER/DECODER BLOCK DIAGRAM (INFORMATIVE)

An example block diagram of a SpaceWire Encoder/Decoder is illustrated in Figure 7-1 below.

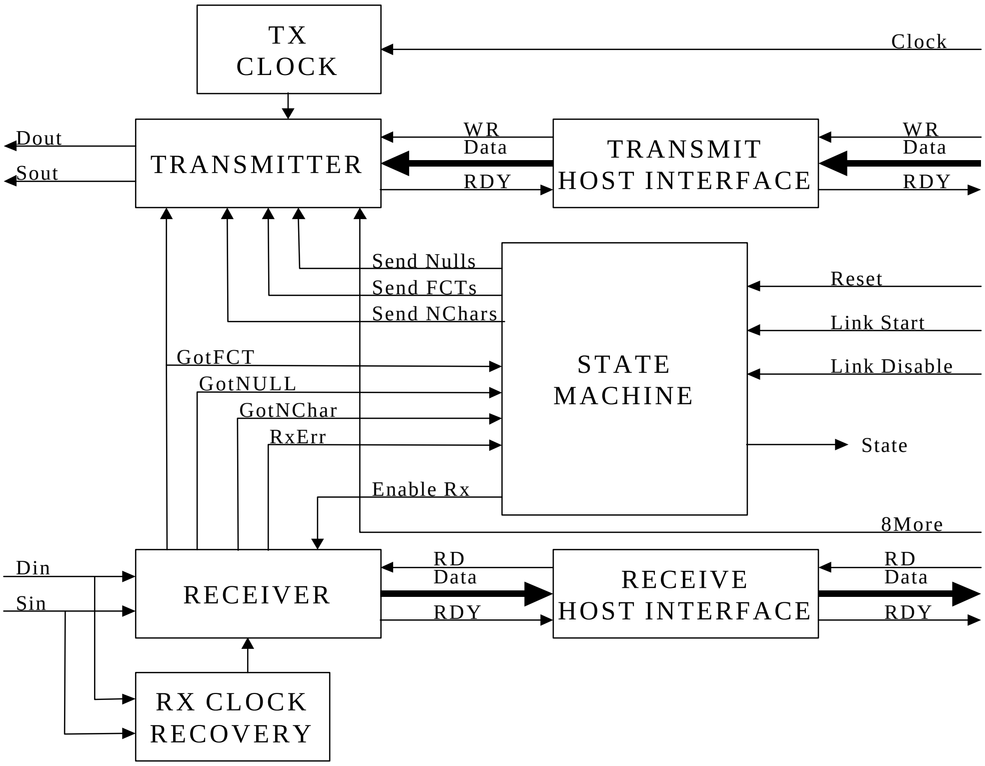


Figure 7-1 Example SpaceWire Link Interface Block Diagram

7.3.1 Transmitter

The Transmitter is responsible for encoding data and transmitting it using the DS encoding technique. It receives its data from the Transmit Host Interface. If there is no data to transmit the Transmitter will send NULLs. The Transmitter is only allowed to send data if the host system at the other end of the link (end B) has room in its host receive buffer. This is indicated by the link interface at end B sending an FCT, indicating that it is ready to accept another 8 data characters. The Transmitter is responsible for keeping track of the FCTs received and the number of data characters sent to avoid input buffer overflow at the other end of the link. To do this the Transmitter holds a credit count of the number of characters it has been given permission to send. The transmitter is also responsible for sending FCTs whenever the local Receiver has space for eight more data characters.

7.3.2 Transmit Clock

The Transmit Clock is responsible for producing the clock signals needed by the transmitter. The transmit-clock signals are typically derived from the local system clock or from a special transmit clock circuit.

7.3.3 Transmit Host Interface

The Transmit Host Interface provides the interface between the Transmitter and the local system source of data. The local system writes data into the Transmit Host Interface at any time provided the host interface is ready to receive data (see section 6.4).

7.3.4 Receiver

The Receiver is responsible for decoding the DS signals (Din and Sin) to produce a sequence of data characters that are passed on to the host system via the Receive Host Interface. It also receives NULLs, FCTs and other control characters (EOP, EEP). NULLs represent an active link. They are flagged to the exchange-level state machine (see section 7.4) but are ignored otherwise. When an FCT is received the Receiver must inform the Transmitter so that it can update its credit count accordingly. All other control characters received are flagged to the host system. The receiver will ignore any N-Chars, L-Chars, parity errors or escape errors until the first NULL has been received. The disconnection detection mechanism with the receiver will be enabled as soon as the first bit arrives (i.e. first transition detected on D or S inputs to receiver).

7.3.5 Receive Clock Recovery

The receive-clock is recovered by simply XORing the received data and strobe signals together. The Receive Clock Recovery circuit provides all the clock signals needed by the receiver.

7.3.6 Receive Host Interface

The Receive Host Interface provides the interface between Receiver and the local host system. As data is received by the Receiver it is written into the Receive Host Interface and passed on to the local host system. The local host system is responsible for informing the link interface whenever it is ready to receive eight more data characters from the Receive Host Interface so that the Transmitter can send an FCT to the interface at the other end of the link (see section 6.4).

7.3.7 State Machine

The state machine controls the overall operation of the link interface. It provides link initialisation, normal operation and error recovery services. The operation of the state machine is discussed in detail in the remainder of section 7.

7.3.8 Receive Buffer Data Management

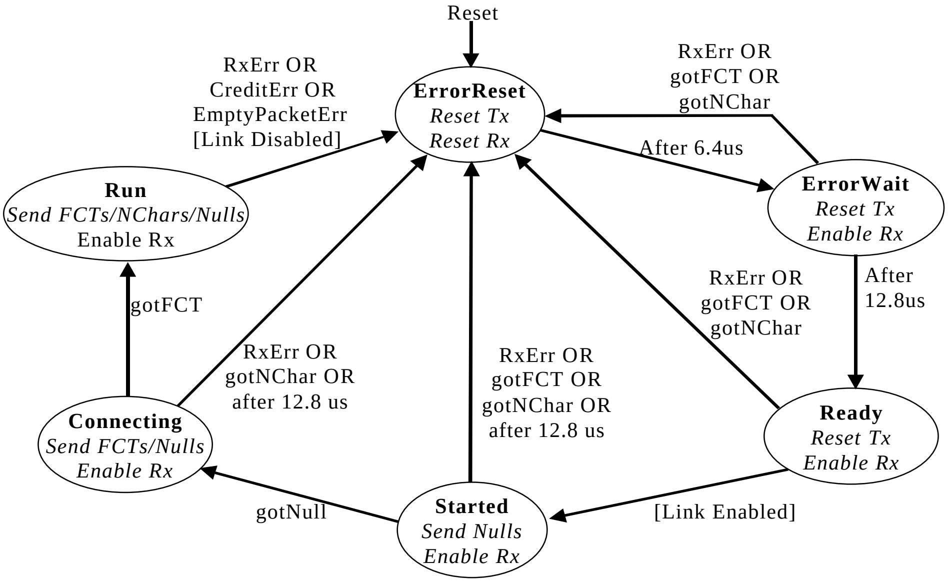
The host system is responsible for data buffer management. This makes the SpaceWire interface more versatile and eases partitioning of the error recovery mechanism (see section 10) across the various levels of the SpaceWire standard. Several different types of host receiver buffering may be implemented:-

- FIFO buffering – where the size of the FIFO buffer depends upon the particular application.
- Memory buffering – where direct memory access (DMA) is used to transfer data to host system memory. As soon as the DMA channel has been set up, several FCTs can be requested immediately to allow the data to be transferred as fast as possible.
- No buffering – where the host system is able to accept data at the highest rate that the link interface can provide it. In this case several FCTs can be sent initially, followed by one more every time eight normal characters are received.

7.4 STATE MACHINE (NORMATIVE)

The complete state transition diagram for the SpaceWire link interface is illustrated in Figure 7-2 below.

The state diagram notation is explained in section 1.7.3.



RxErr = Disconnect Error OR Parity Error OR Escape Error (ESC not followed by FCT)
Note: Disconnect Error only enabled after first bit received.
Parity Error, Escape Error,, gotFCT, gotNChar only enabled after first Null received (i.e. gotNull asserted).

Figure 7-2 State Diagram for SpaceWire Link Interface

7.4.1 Definition of States

In this section the states represented in figure 7-2 are described.

7.4.1.1 ErrorReset

The *ErrorReset* state shall be entered after a system reset, after link operation has been terminated for any reason or if there is an error during link initialisation. In the *ErrorReset* state the Transmitter, Receiver, Transmit Host Interface and Receive Host Interface shall all be reset. When the reset signal is de-asserted the *ErrorReset* state shall be left unconditionally after a delay of 6.4 µs (nominal) and the state machine shall move to the *ErrorWait* state. Whenever the reset signal is asserted the state machine shall move immediately to the ErrorReset state and remain there until the reset signal is de-asserted.

7.4.1.2 ErrorWait

The *ErrorWait* state shall be entered only from the *ErrorReset* state. In the *ErrorWait* state the Receiver shall be enabled and the Transmitter shall be held reset. This allows the Receiver to start the disconnection detection mechanism (after registering a transition on the D or S line) and to begin looking for the arrival of a NULL. If a NULL is received then the gotNULL condition shall be set. This condition will be acted upon in the *Started* state. The *ErrorWait* state shall be left unconditionally after a delay of 12.8 µs (nominal) and the state machine shall move to the *Ready* state. If, while in the *ErrorWait* state, a disconnection error is detected, or if after the first NULL has been received, a parity error or escape error occurs, or any character other than a NULL is received, then the state machine shall move back to the *ErrorReset* state.

The ErrorReset and ErrorWait state with their 6.4 µs and 12.8 µs delays ensure that the receivers at both ends of a link are enabled before either end begins transmission.

7.4.1.3 Ready

The *Ready* state shall be entered only from the *ErrorWait* state. In the *Ready* state the link interface is ready to initialise as soon as it is allowed to do so. The Receiver shall be enabled and the Transmitter shall be held reset. If a NULL is received then the gotNULL condition shall be set. This condition will be acted upon in the *Started* state. The state machine shall wait in the *Ready* state until the [Link Enabled] guard becomes true (see section 7.5) and then it shall move on into the *Started* state. If, while in the *Ready* state, a disconnection error is detected, or if after the first NULL has been received, a parity error or escape error occurs, or any character other than a NULL is received, then the state machine shall move to the *ErrorReset* state.

In the Ready state the two receivers are enabled and the state machine is waiting for the local host system to command the link to start.

7.4.1.4 Started

The *Started* state shall be entered from the *Ready* state when the link interface is enabled. In the *Started* state the state machine begins making a connection with the link interface at the other end of the link by sending NULLs. When the Started state is entered a 12.8 µs (nominal) timeout timer shall be started. In *Started* state the Receiver shall be enabled and the Transmitter shall send NULLs. If a NULL is received then the gotNULL condition shall be set. The state machine shall move to the *Connecting* state if the gotNULL condition is set. The NULL that set the gotNULL condition may have been received in the *ErrorWait*, *Ready* or *Started* states. In the *Started* state a least one NULL must be requested to be sent from the transmitter before moving to the *Connecting* state. If, while in the *Started* state, a disconnection error is detected, or if after the first NULL has been received, a parity error or escape error occurs, or any other character other than a NULL is received, then the state machine shall move to the *ErrorReset* state. If the 12.8 µs timeout timer expires (i.e. no NULL received since leaving the *ErrorReset* state) then the state machine shall move to the *ErrorReset* state.

In the *Started* state the attempt to make a connection across the link is started. NULLs are transmitted and the receiver is waiting to receive a NULL.

7.4.1.5 Connecting

The *Connecting* state shall be entered from the *Started* state after a NULL has been received (gotNULL condition set). On entering the *Connecting* state a 12.8 µs timeout timer shall be started. In the *Connecting* state the Receiver shall be enabled and the Transmitter shall be enabled to send FCTs and NULLs. If an FCT is received the state machine shall move to the *Run* state. If a disconnect error, parity error or escape error is detected, or if an N-Char is received while in the *Connecting* state then the state machine shall move to the *ErrorReset* state. If the 12.8 µs timeout occurs then the state machine shall move to the *ErrorReset* state.

The *Connecting* state is entered when the link interface (end A) has received a NULL. It now has to wait for an FCT to be received indicating that the other end of the link (end B) has also received a NULL. When the link interface has received a NULL and an FCT it means that communication is established in both directions. If an FCT fails to arrive within 12.8 µs then something is wrong with the link connection so the link interface is reset once more (*ErrorReset* state) and connection is attempted once again.

7.4.1.6 Run

The *Run* state shall be entered from the *Connecting* state. In the *Run* state the Receiver is enabled and the Transmitter is enabled to send N-Chars, FCTs and NULLs. If the link interface is disabled, or if a disconnect error, parity error, escape error, credit error or empty packet error is detected (see section 7.4.2), while in the *Run* state then the state machine shall move to the *ErrorReset* state.

The *Run* state is the state for normal operation. Link connection has been made. L-Chars and N-Chars can flow freely in both directions across the link. The link remains in the *Run* state until an error occurs or until the link is disabled.

7.4.2 Definition of Transitions

In this section the transitions represented in figure 7-2 are described.

7.4.2.1 Reset

Reset represents power on reset, other hardware reset or software commanded reset.

7.4.2.2 After T µs

After 6.4 µs or after 12.8 µs represents a delay of the specified time measured from when the current state is entered. The actual time intervals are nominal delays (see section 7.10).

7.4.2.3 [Link Enabled]

[Link Enabled] is a condition that must be met for the transition to occur (i.e. a guard). [Link Enabled] can be set true by software or hardware (see section 7.5).

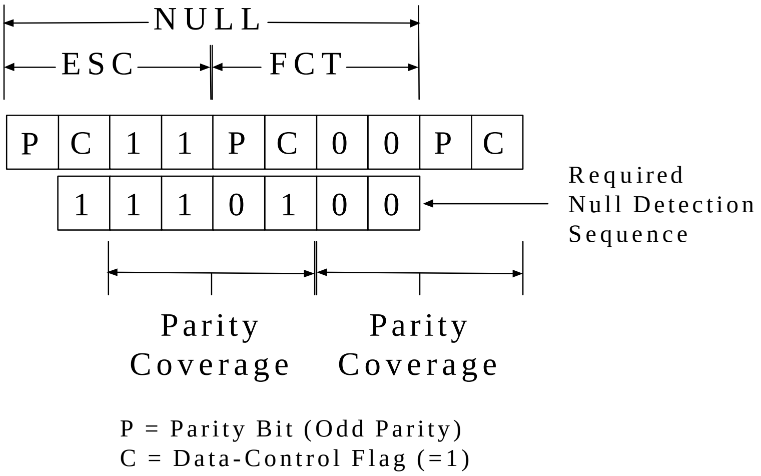
7.4.2.4 gotNull

gotNull means that a NULL has been received. NULL detection is enabled whenever the Receiver is enabled.

Any sequence of bits encountered prior to the first NULL being received shall be ignored.

NULL detection shall include the parity bit within the NULL i.e. the parity bit that covers the ESC character within the NULL control code. The second parity bit associated with the NULL, that covers the FCT character shall not be included in the NULL detection. Hence the NULL shall be detected and gotNull asserted, when the 1110100 sequence of bits is received as illustrated in figure 7-3.

If a parity error occurs with the first parity bit (for the ESC character) then the NULL will not be detected. If a parity error occurs with the second parity bit, then this error will be picked up immediately since parity error detection is enabled within the receiver after a NULL has been received.



NULL
ESC
FCT
P C 1 1 P C 0 0 P C
1 1 1 0 1 0 0
Required Null Detection Sequence
Parity Coverage
Parity Coverage
P = Parity Bit (Odd Parity)
C = Data-Control Flag (=1)

Figure 7-3 NULL Detection Sequence

7.4.2.5 gotFCT

gotFCT means that an FCT has been received. FCTs are only valid when received in the *Connecting* and *Run* states. If received in any other state they represent an error.

7.4.2.6 gotNChar

gotNChar means that an N-Char has been received. An N-Char received when the exchange-level state machine is not in the *Run* state is an error.

7.4.2.7 [Link Disabled]

[Link Disabled] is a condition set by external hardware or software in order to disable and stop the link interface (see section 7.5).

7.4.2.8 RxErr

RxErr or Receiver Error is shorthand for Disconnect Error, Parity Error or Escape Error.

7.4.2.9 Disconnect Error

Disconnect Error is an error condition asserted when the length of time since the last transition on the D or S lines was longer ago than 850 ns nominal (see section 7.10). The disconnect detection mechanism is activated after leaving the *ErrorReset* state as soon as the first edge is detected on the D or S line.

7.4.2.10 Parity Error

The parity error event occurs if a parity error is detected (see section 6.3). Parity detection is enabled whenever the receiver is enabled after the first NULL has been received.

7.4.2.11 Escape Error

The escape error event occurs if an ESC character is followed by any character other than an FCT (ESC followed by FCT is a NULL, see section 6.2). Escape error detection is enabled whenever the receiver is enabled after the first NULL has been received.

7.4.2.12 Character Sequence Error

Any characters received before a NULL has been received are ignored. Once a NULL has been received an FCT received before a NULL has been sent indicates an error (i.e. FCT received in *ErrorWait*, *Ready* or *Started* state). An N-Char should only be received after both a NULL and an FCT have been received otherwise an error has occurred (i.e. N-Char can only be received in the *Run* state).

Note: In the state diagram of figure 7-4, the invalid gotFCT or gotNChar events are shown explicitly, rather than as a general character sequence error event.

7.4.2.13 Credit Error

Credit error occurs if data is received when the host system is not expecting any more data, i.e. when all the N-Chars expected, according to the requested “8 more” N-Chars and subsequent transmitted FCTs, have been received. A credit error ought never to occur and indicates that some undetected error has occurred on the link affecting the transfer of FCTs.

7.4.2.14 Empty Packet Error

Empty packets are not permitted (see section 8). If the next N-Char received after an EOP or EEP is another EOP or EEP then an empty packet error has occurred. An empty packet error ought never to occur and indicated that some undetected error has occurred on the link producing a spurious EOP or EEP.

7.5 AUTOSTART (NORMATIVE)

A link interface should be able to be commanded to start automatically on receipt of a NULL. In this case the Link Enabled condition in the state machine should be set as follows.

[Link Enabled] = (NOT [Link Disabled]) AND ([LinkStart] OR ([AutoStart] AND gotNull))

LinkDisabled is the flag set by software or hardware to indicate that the link is disabled. This corresponds to the Link Disabled condition in the state diagram.

LinkStart is a flag set by software or hardware to start a link i.e. to cause the transition from the *Ready* state to the *Started* state.

AutoStart is a flag set by software or hardware to indicate that the link should start automatically on receipt of a NULL.

gotNull is the flag indicating that the link interface has received a NULL.

LinkStart and AutoStart are only acted upon when the link interface is not disabled i.e. [LinkDisabled] = False.

The AutoStart facility enables a system to be set up where one end (end A) of the link is held waiting for the other end (end B) to attempt connection. As soon as end B tries to connect end A will respond immediately. This allows connection of a link to be controlled from one end of the link only.

7.6 LINK INITIALISATION (INFORMATIVE)

This section explains how the state diagram given in section 7.4 handles link initialisation, going from the reset of one end of a link through to the link operating normally sending data in both directions. The basic state diagram with the receiver error conditions removed is illustrated in Figure 7-4 below.

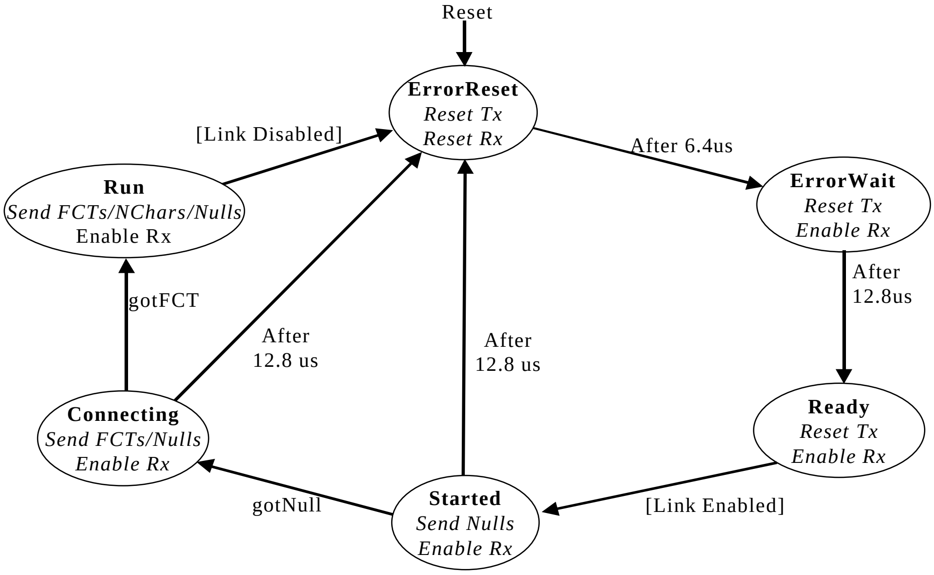


Figure 7-4 Basic State Diagram for SpaceWire Link Interface

After a link interface (one end of a link) has been reset, it enters the *ErrorReset* state where the transmitter and receiver are reset. The transmitter reset is a controlled reset, resulting first in the transmitter stopping transmission followed by resetting of the strobe signal and then the data signal. This sequence avoids the simultaneous transition of both data and strobe signals.

The link interface will remain in the *ErrorReset* state for approximately 6.4 µs and then move to the *ErrorWait* state. In the *ErrorWait* state the transmitter remains disabled, but the receiver is enabled so that it can begin searching for NULLs.

The link interface remains in the *ErrorWait* state for 12.8 µs and then moves into the *Ready* state. The 6.4 µs from *ErrorReset* to *ErrorWait* and the 12.8 µs delay from *ErrorWait* to *Ready* make sure that the receivers at both ends of a link are ready to receive characters before either end starts transmission.

The link interface may be enabled in many possible ways, for example, by software command, automatically when the receiver detects a NULL, or the link may be permanently enabled (see section 7-5). When a link interface is enabled the [LinkEnabled] condition becomes true. The link interface will move from the *Ready* state to the *Started* state as soon as the link is enabled.

In the *Started* state the link interface instructs the transmitter to start sending NULLs. It will remain in this state until the receiver detects that a NULL has been received over the link or until a connection timeout has expired. The connection timeout is set to a nominal 12.8 µs since this period has to be generated for the *ErrorReset* state timeout. If a NULL is received then the link interface will move to the *Connecting* state. If no NULL is received within 12.8 µs it will move to the *ErrorReset* state. In the latter case the link interface will go through the reset sequence (*ErrorReset, ErrorWait, Ready*) and attempt to make a connection again a short time later.

In the *Connecting* state the link interface will send some FCTs (and NULLs) and will wait for an FCT to be received. If an FCT is received the link interface will move on to the *Run* state. If an FCT has not been received within 12.8 µs then link connection has not been made properly, so the link interface moves back to the *ErrorReset* state. The link interface will then go through the reset sequence (*ErrorReset, ErrorWait, Ready*) and attempt to make a connection again a short time later.

When the link enters the *Run* state it starts normal operation, sending and receiving data and control characters. It remains in the *Run* state until the link is disabled. The link interface then moves through the reset sequence (*ErrorReset, ErrorWait, Ready*) and stays in the ready state until the link is enabled once more.

Table 7-1 and figure 7-5 illustrate a typical initialisation sequence. Link interface A is at one end of the link and link interface B is at the other end.

Table 7-1 Example: Typical Initialisation Sequence.

Link Interface End A	**Link Interface End B**	**Event/Condition Causing Transition**
ErrorReset	*ErrorReset*	End A times out after 6.4 µs and moves to the *ErrorWait* state.
ErrorWait	*ErrorReset*	End B times out after 6.4 µs and moves to the *ErrorWait* state.
ErrorWait	*ErrorWait*	End A times out after 12.8 µs and moves to the *Ready* state.
Ready	*ErrorWait*	End A is link enabled so moves to the *Started* state.
Started Sending Nulls	*ErrorWait*	End B detects NULL sent from end A. This is registered as gotNull by end B. There is no state change.
Started Sending Nulls	*ErrorWait*	End B times out after 12.8 µs and moves to the *Ready* state.
Started Sending Nulls	*Ready*	End B is link enabled so moves straight to the *Started* state.
Started Sending Nulls	*Started* Sending Nulls	End B sends a NULL. It has already detected a NULL (Got Null) so can now move to the *Connecting* state.
Started Sending Nulls	*Connecting*	End A detects NULL sent from end A (gotNull) and can move to the *Connecting* state. End B sends out FCTs (and NULLs).
Connecting	*Connecting*	End A sends out FCTs (and NULLs). End B sends out FCTs (and NULLs). End A receives an FCT and moves to the *Run* state.
Run	*Connecting*	End A sends out FCTs, N-Chars and NULLs. End B receives an FCT and moves to the *Run* state.
Run	*Run*	Both ends are in the *Run* state and begin normal operation sending and receiving N-Chars, FCTs and NULLs.

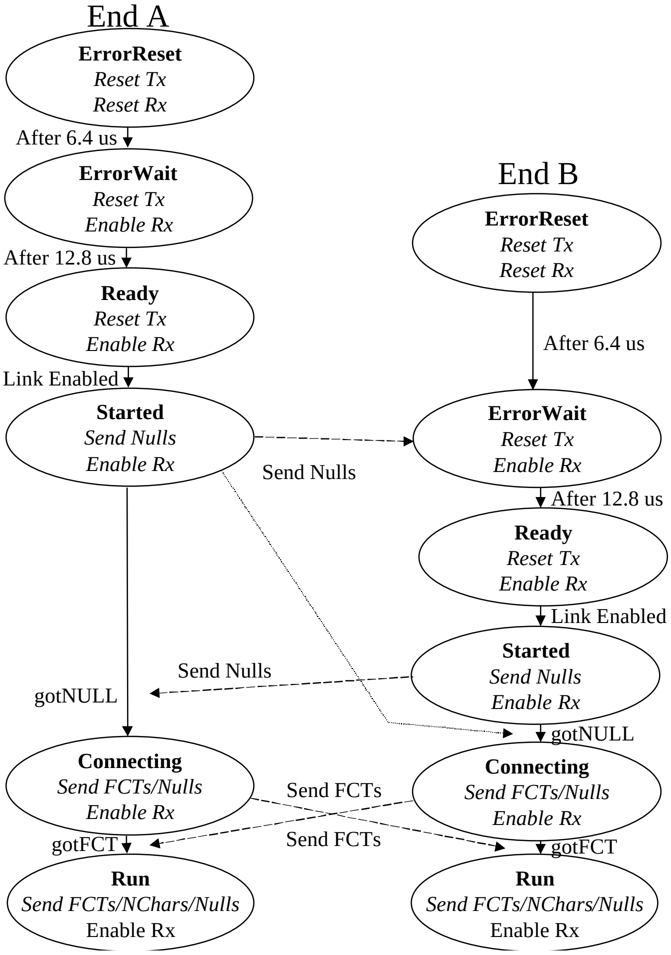


Figure 7-5 Example: Typical Initialisation Sequence

A link can only send FCTs once it has received a NULL. So, when a link has received an FCT it knows that the link is connected in both directions. NULL correlation in the *ErrorWait*, *Ready* and *Started* states ensures proper character synchronisation. The NULL/FCT handshake sequence ensures that the link is connected in both directions before normal link operation begins.

The time taken from a link being enabled in the *Starting* state to normal operation in the *Run* state can be as little as the time taken to transfer two NULLs and an FCT. End A is enabled and sends a NULL. End B is autostart enabled when it receives the NULL from end A and sends a NULL followed by an FCT. End A receives the NULL from end B and sends an FCT. Both ends receive FCTs and move to the *Run* state. At a link data signalling rate of 10 Mbps this could take just 2 µs.

7.7 NORMAL OPERATION (INFORMATIVE)

In normal operation both ends of the link are in the *Run* state and will be sending and receiving N-Chars, FCTs and NULLs.

Consider a host system with buffer space sufficient to hold 16 normal-characters. This host system at one end of a link (end A) will indicate that it is ready to receive normal-characters by twice flagging that it has room for 8 more characters to the link interface. The link interface will send two FCTs to the other end of the link (end B) which will increment its credit count accordingly (from zero to 16). The link interface at end B indicates to its host system that it is ready to transmit data (normal-characters) When the host system at end B has data to transfer, it will pass it to the link interface, which will send it across the link to end A. As each character is transmitted by the link interface (end B) it will decrement its credit count until it reaches zero, at which point the link interface (end B) will indicate to its host system that it is not ready to transfer any more data. The data received at end A will be passed on to its host system which will place it in its 16 character buffer. As the host system uses the data out of this buffer it makes space for more data to be received. As soon as there is space for another 8 more characters it flags this to the link interface, which will then send out another FCT informing end B that 8 more normal characters may be sent.

7.8 Error Detection (Normative)

There are six forms of receiver error that can be detected and acted upon at the exchange level – disconnect errors, parity errors, escape errors, credit errors, character sequence errors and empty packet errors. Whenever one of these errors occur both character synchronisation and flow-control status cease to be valid. Both ends of the link must be reset and re-initialised to recover character synchronisation and flow control status.

An error can occur in the transmitter if it is given an invalid character to transmit. In this event the transmitter shall ignore the invalid character, cease N-Char transmission and report the error to the network level.

7.8.1 Disconnect Error

An operational link interface sends normal-characters, FCTs or NULLs continuously, thus the data and/or strobe signals are always changing. The receiver shall detect a disconnection when the time interval from the last transition on either the data or strobe signal exceeds the disconnect-detection time. The disconnect-detection time shall be 850 nsec nominal (see section 7.10.2 for the disconnect timing specification). Before being able to detect a disconnect error the receiver must have received at least one bit.

A disconnect error can either be caused when one end of the link is disabled or when the link is physically disconnected (intentionally or unintentionally). If a physical disconnection is the cause of the disconnect error then both ends of the link will try repeatedly to make a connection until the link is reconnected or until the link interfaces are disabled.

If a disconnect error is detected then the link interface shall follow the exchange of silence error recovery procedure described in section 7.8.7. If the disconnect error occurs in the *Run* state then the disconnect error shall be flagged up to the network level as a link error (see section 7.8.8).

7.8.2 Parity Error

When a parity bit is received it shall be checked (see section 6.3). If a parity error occurs after the first NULL has been received, then the link interface shall follow the error recovery procedure described in section 7.8.7. If the parity error occurs in the *Run* state then the parity error shall be flagged up to the network level as a link error (see section 7.8.8).

7.8.3 Escape Error

An ESC character shall only be used to form the NULL (ESC followed by FCT, see section 6.2). If a ESC character is received followed by any character other than an FCT then the link interface shall follow the error recovery procedure described in section 7.8.7. If the escape error occurs in the *Run* state then the escape error shall be flagged up to the network level as a link error (see section 7.8.8).

7.8.4 Credit Error

In the *Run* state if a normal character is received when the host system is not expecting any N-Chars then a credit error has occurred. A credit error may be caused if an error occurs undetected by the parity bit (e.g. two bits in error) which results in one or more spurious FCTs. In the event of a credit error the link interface shall follow the error recovery procedure described in section 7.8.7. If the credit error occurs in the *Run* state then the credit error shall be flagged up to the network level as a link error (see section 7.8.8).

7.8.5 Character Sequence Error

During initialisation it is possible for a link interface to receive FCTs or normal-characters when they are not expected. Any unexpected characters are caught by the exchange-level state machine resulting in the link being reset and re-initialised (see figure 7-2). A character sequence error shall not be flagged up to the network level as a link error because it can only occur during link initialisation (see section 7.8.8).

7.8.6 Empty Packet Error

An EOP or EEP followed immediately by another EOP or EEP represents an empty packet, which is not permitted. In the *Run* state, if the next N-Char received after an EOP or EEP has been received is another EOP or EEP, then there has been an error on the link. In this event, the link interface shall follow the error recovery procedure described in section 7.8.7. If the empty packet error occurs in the *Run* state then the empty packet error shall be flagged up to the network level as a link error (see section 7.8.8).

7.8.7 Exchange of Silence Error Recovery Procedure

When one end of the link (end A) is disabled or detects an error, it will cease transmission. This will cause a disconnect error at the other end of the link (end B). End B will then cease transmission resulting in a disconnect error at end A. This procedure is known as an “exchange of silence”. Both ends of the link will cycle through the reset sequence (*ErrorReset, ErrorWait*, *Ready*) ending up in the *Ready* state ready to begin operation once enabled.

If both ends are enabled then they will move to the *Started* state and re-initialise.

If one end (end A) is disabled and the other end (end B) is enabled then end B will move from the *Ready* state to the *Started* state and will send NULLs for 12.8 µs. Since end A is disabled it cannot respond. End A will, however, have started its disconnect timer and will also have registered that a NULL has been received. When end B completes the 12.8 µs timeout it will move to the *ErrorReset* state and disconnect (stop its output). End A is able to detect the disconnection so will also move to the *ErrorReset* state. Both ends will once again move through the reset sequence. This series of events will continue until either end A is enabled or end B is disabled.

7.8.8 Link Error

During initialisation, receiver errors (disconnect error, parity error, escape sequence error, character sequence error, credit error and empty packet error) are likely to occur and are part of the natural initialisation sequence. These errors shall not be reported to the network level when they occur during link initialisation (*ErrorReady*, *ErrorWait*, *Ready*, *Started* and *Connecting* states).

Once a link connection has been established (*Run* state) then a receiver error represents a failure of the link connection and must be reported to the network level so that appropriate action for error recovery and/or reporting can be taken. A link error is reported to the network level whenever any of the following errors occur while a link interface is in the *Run* state: disconnect error, parity error, escape sequence error, credit error, empty packet error. Note the exclusion of character sequence error from this list. A character sequence error is only possible during initialisation.

7.9 EXCEPTION CONDITIONS (INFORMATIVE)

Several exception conditions have been identified where things, for one reason or another, do not follow the usual sequence of events. These exceptions are considered in this section.

7.9.1 Disconnect error while waiting to start

“Waiting to start” means that a link interface is in either the *ErrorReset*, *ErrorWait*, *Ready* or possibly the *Started* state. For a disconnect error to be detected while waiting to start, the other end of the link (end B say) must have sent at least one bit, so that the disconnect detect mechanism at end A can be activated. End B must have then given up waiting for end A to send a NULL and moved to the *ErrorReset* state and stopped its transmitter – thus causing the disconnect. An alternative possibility is that the link became physically disconnected. The following tables illustrate the various sequences of events starting from when end B has just moved to the *ErrorReset* state.

Table 7-2 End A in ErrorReset state

Link Interface End A	Link Interface End B	Event/Condition Causing Transition
ErrorReset	*Started*	End B times out while waiting to receive a NULL from end A or end B detects a disconnect. End B moves to the *ErrorReset* state.
ErrorReset	*ErrorReset*	End A and end B are both in the *ErrorReset* state they will step through the reset sequence and will start again when both ends are enabled.

Table 7-3 End A in ErrorWait state

Link Interface End A	Link Interface End B	Event/Condition Causing Transition
ErrorWait	*Started*	End B times out while waiting to receive a NULL from end A or end B detects a disconnect. End B moves to the *ErrorReset* state.
ErrorWait	*ErrorReset*	End B stops transmission. This is detected at end A as a disconnect error. End A moves to the *ErrorReset* state.
ErrorReset	*ErrorReset*	Both ends step through the reset sequence and will start again when both ends are enabled.

Table 7-4 End A in Ready state

Link Interface End A	Link Interface End B	Event/Condition Causing Transition
Ready	*Started*	End B times out while waiting to receive a NULL from end A or end B detects a disconnect. End B moves to the *ErrorReset* state.
Ready	*ErrorReset*	End B stops transmission. This is detected at end A as a disconnect error. End A moves to the *ErrorReset* state.
ErrorReset	*ErrorReset*	Both ends step through the reset sequence and will start again when both ends are enabled.

Table 7-5 End A in Started state

Link Interface End A	Link Interface End B	Event/Condition Causing Transition
Started	*Started*	End B times out while waiting to receive a NULL from end A or end B detects a disconnect. End B moves to the *ErrorReset* state.
Started	*ErrorReset*	End B stops transmission. This is detected at end A as a disconnect error. End A moves to the *ErrorReset* state.
ErrorReset	*ErrorReset*	Both ends step through the reset sequence and will start again when both ends are enabled.

If a physical disconnection has occurred then both ends of the link will continue to try to make a connection, cycling around the reset sequence, until they are disabled or until the connection is re-established.

7.9.2 Link connected in one direction but not in the other

A link may be connected in one direction and not in the other while a link is in the process of being plugged in (contact bounce time may be significantly larger than tens of µs) or if there is a break in the link cable.

In this case the sequence of events listed in the table below will be followed. Consider for convenience that both links are in the *started* state and that end A is connected to end B, but end B is not connected to end A.

Table 7-6 Link connected in one direction (A to B) but not in other

Link Interface End A	Link Interface End B	Event/Condition Causing Transition
Started	*Started*	End A is sending NULLs to end B and these are received, starting the disconnect timer of end B and registering as GotNull at end B. End B will therefore move to the *Connecting* state. End B is also sending NULLs to end A but these are not received at end A because the link is not connected in this direction.
Started	*Connecting*	End A is in the *Started* state waiting for NULLs to arrive. After waiting for 12.8 µs end A times out and moves to the ErrorReset state. End B sends NULLs and FCTs to end A but these are not received at end A. End B is able to receive FCTs but none are sent by end A because end A has not received a NULL
ErrorReset	*Connecting*	End B continues to send NULLs and FCTs and is able to accept FCTs.
ErrorReset	*Connecting*	End A ceases transmission and this is detected at end B as a disconnect. End B moves to the *ErrorReset* state.
ErrorReset	*ErrorReset*	Both ends are in the *ErrorReset* state and will now cycle through the reset sequence (*ErrorReset*, *ErrorWait*, *Ready*, etc) until the link is properly connected or until one or both link interfaces are disabled.

7.9.3 Parity error while waiting to start

Parity errors are only recognised after a NULL has been received. If a parity error occurs during link initialisation the effect is identical to a disconnect error.

7.9.4 One end starts as other end disconnects

One end (end A) arrives at the *Start* state 12.8 µs before the other end (end B) arrives at the *Start* state. The sequence of event is illustrated the following table.

Table 7-7 One end starts as other end disconnects

Link Interface End A	Link Interface End B	Event/Condition Causing Transition
Started	*Ready*	The timeout timer at end A expires (after 12.8 µs) so end A moves to the *ErrorReset* state. End B, in the *Ready* state, has just been enabled and now moves to the *Started* state.
ErrorReset	*Started*	End B has already received a NULL from end A (gotNULL is TRUE) so moves straight on into the *Connecting* state.
ErrorReset	*Connecting*	End A stops transmitting and causes end B to detect a disconnect. End B then moves on into the *ErrorReset* state.
ErrorReset	*ErrorReset*	Both ends step through the reset sequence and will start properly next time round.

7.9.5 D connected, S disconnected

If D is connected and S disconnected then the clock generated in the receiver will follow the data signal i.e. there will be a clock edge every time the data signal changes. This results in a continuous sequence of 0101010101.

During initialisation this sequence will be ignored as it does not produce a NULL so initialisation will fail until the strobe signal is properly connected. In this case the link interface will cycle round *ErrorReset*, *ErrorWait*, *Ready*, *Started* until full link connection is achieved or until the link is disabled.

If S becomes disconnected after a NULL has been received, this sequence will quickly produce a parity error because the parity is even for both control characters (4-bits) and data characters (10-bits) extracted from the 01010101… sequence, whereas the required parity is odd (see section 6.3). The parity error will cause the link interface to cycle through *ErrorReset*, *ErrorWait*, *Ready* and *Started* until S is connected or the link is disabled.

7.9.6 S connected, D disconnected

If S is connected and D disconnected then the clock generated in the receiver will follow the strobe signal i.e. there will be a clock edge every time the strobe signal changes. This results in a continuous sequence of 1111111111 since the data signal input will go to 1 when the data line is disconnected. If the data line is shorted to ground then the continuous sequence 0000000000 will be received.

During initialisation either sequence will be ignored as it does not produce a NULL so initialisation will fail until the data signal is properly connected. In this case the link interface will cycle round *ErrorReset*, *ErrorWait*, *Ready*, *Started* until full link connection is achieved or until the link is disabled.

If S becomes disconnected after a NULL has been received this sequence will quickly produce a parity error because the parity is even for both control characters (4-bits) and data characters (10-bits) extracted from the 01010101… sequence, whereas the required parity is odd (see section 6.3). The parity error will cause the link interface to cycle through *ErrorReset*, *ErrorWait*, *Ready* and *Started* until S is connected or the link is disabled.

7.9.7 One side of differential pair disconnected

The effect of disconnecting one side of the differential pair will depend upon the values of the internal bias resistors at the interface, on the length of cable and on the grounding arrangements. There are three possibilities.

1. The data and strobe signals are still received correctly but with a much reduced noise margin. The link will continue to operate with a significant increase in the number of detected parity errors.
2. The stobe signal sits at logic 0 or logic 1. This is similar in effect to the D connected, S disconnected case of section 7.9.5 above.
3. The data signal sits at logic 0 or logic 1. This is similar in effect to the S connected, D disconnected case of section 7.9.6 above.

REMINDER: Further testing needs to be done to confirm the above.

7.10 LINK TIMING (NORMATIVE)

7.10.1 D and S Reset Timing

The delay between the reset of the strobe signal and the data signal shall be between 625 ns (period of slowest permitted transmit clock, 2MHz – 20%) and the shortest clock cycle time for the transmitter (period of maximum clock, dependent upon implementation). TBC

7.10.2 Disconnect Timing

The disconnect timeout of 850 ns nominal shall be between 740 ns (8 cycles of 10MHz clock – 20%) and 1080 ns (9 cycles of 10MHz clock + 20%). TBC

7.10.3 Exchange Timeout Periods

The 6.4 µs (nominal) timeout period shall be between 5.12 µs (64 cycles of 10MHz clock – 20%) and 7.78 µs (65 cycles of 10MHz clock + 20%). TBC

The 12.8 µs (nominal) timeout period shall be between 10.24 µs (128 cycles of 10MHz clock – 20%) and 15.48 µs (129 cycles of 10MHz clock + 20%). TBC

7.10.4 Other Timing

Further timing constraints may have to be defined for the exchange level state machine e.g. after condition X the link interface shall move from state A to state B in less than Y µs.

8. PACKET LEVEL

The packet level protocol follows the principles of the DS-SE and DS-DE character level encoding given in section 9, “Common Packet Level”, of the IEEE Standard 1355-1995 [RD1]. Some ambiguities in the IEEE 1355 specification have been resolved in the SpaceWire standard. See Annex C for details of the differences between SpaceWire and IEEE 1355 and the reasons for those differences.

8.1 PACKETS

A packet shall comprise a destination address, a cargo and an end_of_packet marker.

<destination address> <cargo> <EOP>

8.1.1 Destination

The destination address shall consist of a list of zero or more destination identifiers (dest_id).

<destination address> = <dest_id1> <dest_id2> … <dest_idN>

A destination identifier shall comprise one or more data characters.

The destination list shall not be delimited.

The case of zero destination identifiers in the destination list (i.e. the destination list is empty) is intended to support a network which is simply a single point-to-point link from source to destination.

The case of one or more destination identifiers in the destination list is intended to support routing of a packet across a network.

8.1.2 Cargo

The cargo shall comprise the data characters that are to be transferred from the source to the destination.

The cargo shall contain one or more characters.

8.1.3 End_of_Packet Markers

There are two possible end of packet markers: EOP and EEP.

The EOP and EEP control character formats are described in section 6.2 of this standard.

EOP (end of packet) shall be used as the normal end of packet marker indicating the end of a packet.

EEP (error end of packet) shall be used to indicate the end of a packet in which an error has occurred. The data in this packet will be valid, but the packet will be prematurely terminated at the point that the error occurred.

The first data character following either end of packet marker shall be taken as the first character of the next packet.

9. NETWORK LEVEL

This section describes the network level. First the basic concepts are described and then the SpaceWire routing switches, nodes and networks are defined. Network level errors are described and the network level error recovery protocol is defined.

9.1 NETWORK AND ROUTING CONCEPTS (INFORMATIVE)

In this section the basic concepts of packet routing networks that apply to SpaceWire networks are introduced.

9.1.1 Networks

A network is made up of a number of links, nodes and routing switches. The nodes are the sources and destinations of packets. For example, a processor is a type of network node. Links provide the means for passing packets from one node to another. Nodes can be either directly connected by links or they may be connected via routing switches. Usually a node can only support a few links (e.g. six links) and so can only be directly connected to a limited number of other nodes (e.g. six other nodes). Routing switches connect together many nodes and provide a means of routing packets from one node to one of many other possible nodes. A typical network comprising several nodes and routing switches is illustrated in figure 9-1. Packets may be transferred from one node to another through one or more routing switches, or directly from node to node where direct connections exist.

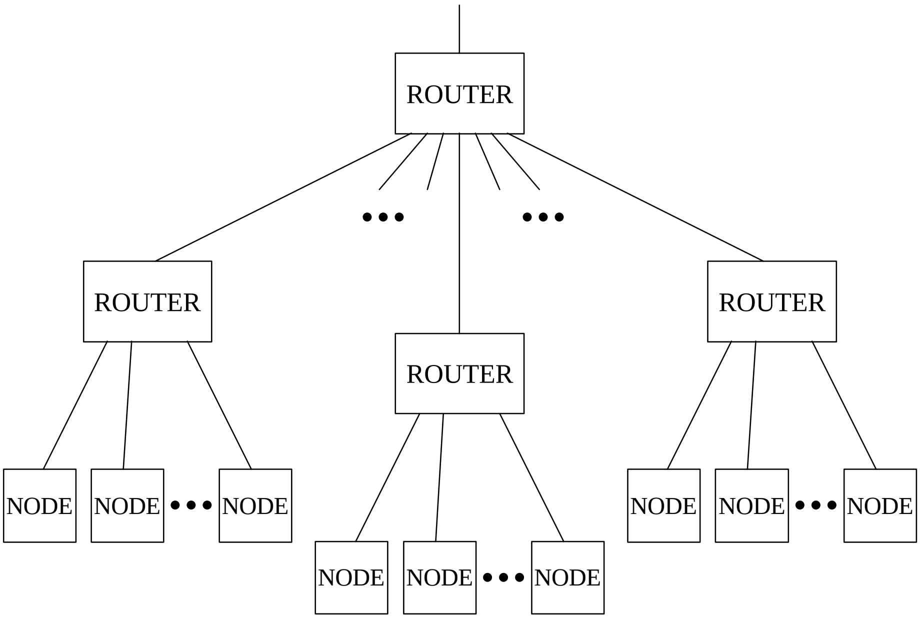


Figure 9-1 A Typical Network

There are two types of routing switch: static and dynamic. A static routing switch sets up connections between nodes and does not change them very often. Dynamic switches change the routing frequently, usually on a packet by packet basis and are consequently also known as packet routing switches. SpaceWire routing switches are dynamic, packet routing switches. Packets may be interleaved across data links and routing switches to provide many virtual communication channels across a few physical data links.

9.1.2 Packets

Data is split into packets so that it can be transported in manageable chunks across a network. Packets of data are the smallest elements of data that are handled at the network level. Packets are regarded as indivisible by the network level and are transported whole across a network.

SpaceWire packets have a simple packet structure (see section 8):-

- Destination –the address of the destination node or task.
- Cargo – the data to be delivered.
- End of Packet Marker – a special character which indicates the end of a packet.

9.1.3 Flow Control

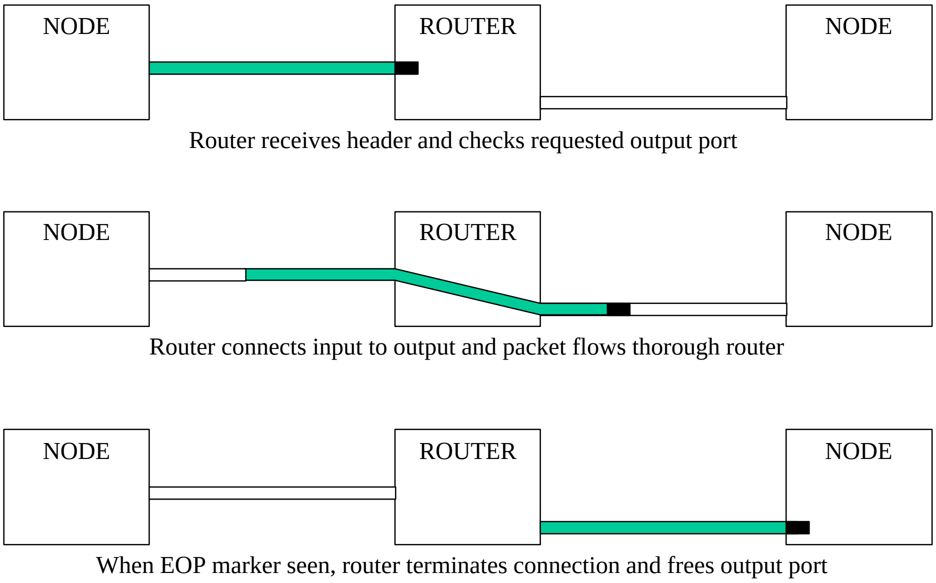
Flow control is necessary to manage the movement of packets across a network from one node to another. For a node or router to receive some data there must be buffer space for that data in the receiving node or router. When the receive-buffer becomes full the receiver must stop the transmitting node from sending any more data.

SpaceWire uses flow control tokens to manage the flow of data across a data link connecting one node to the next (see section 7.2).

9.1.4 Wormhole Routing

Wormhole routing is a particular form of packet routing. Each packet contains a header which holds the destination node address. As soon as the header for a packet is received the switch determines which output port the packet must be routed to by checking the destination address. If the required output port is free then the packet is routed immediately to that output port. That output port is now marked as busy until the last byte of the packet has passed through the switch – indicated by the end of packet marker being detected by the switch. Wormhole routing cuts down on the amount of buffering needed within each switch, compared to a store and forward technique where an entire packet must first be received and stored before it is sent out of the switch.

Wormhole routing is illustrated in figure 9-2 which shows a packet being sent from one node through a routing switch (router) to another node. The header of the packet is marked as black and the rest of the packet as grey. As soon as the router receives the header it checks the required output port. If the output port is free then the router makes a connection between the input port and the output port. The packet then flows through the router. When the end of packet (EOP) marker is received by the switch the router terminates the connection and frees the output port ready for its next packet which can come from any input port.



NODE
ROUTER
NODE
Router receives header and checks requested output port
NODE
ROUTER
NODE
Router connects input to output and packet flows thorough router
NODE
ROUTER
NODE
When EOP marker seen, router terminates connection and frees output port

Figure 9-2 Wormhole Routing

If a required output port is busy then the input port must halt the incoming packet until the output port becomes free. This is achieved by the input port ceasing to send flow control tokens to the source node. The link connecting the source node to the routing switch is then blocked until the routing switch output finishes transferring its current packet and becomes free to transmit the new packet.

9.1.5 Cascading

Direct connection between nodes can accommodate up to five to seven nodes (typically). A single routing switch can usually connect many more nodes depending on the size of the switch (possibly up to 32 nodes, for example). When larger networks are required several switches can be cascaded to form larger networks (see figure 9-1). To arrive at its required destination a packet may then have to travel through several switches.

9.1.6 Header Deletion

Header deletion is a simple and effective technique designed to manage the transfer of packets across an arbitrary sized network. Header deletion is illustrated in figure 9-3 below. The first byte of a packet (the complete destination address, or a part of it) is used to specify the router output port address. When a packet is received at a routing switch its first byte is checked to determine which output port the packet is to be routed through. The first byte of the header is then deleted and the packet passes through the switch without this first byte. The second byte of the original header (now the first byte) is responsible for any subsequent routing.

Figure 9-4 shows a packet passing through two routing switches. The destination address of the packet comprises three bytes. At the first router the first address byte is used to determine the required output port. This byte is then stripped off. At the second router the second address byte is used and stripped off. Finally the packet arrives at the destination node with the third address byte at the front of the packet. This is used to determine where the packet is to go within the destination node (see section 9.1.7).

At each stage as the packet passes though the network, it can be regarded as a packet comprising a single byte address header, a cargo and an end of packet marker. When, at the first router this single byte header is stripped off, the first byte of the cargo becomes the new header. All stages treat the packet in the same way, using the first byte to determine the destination (output port) and then deleting that first byte before forwarding the packet.

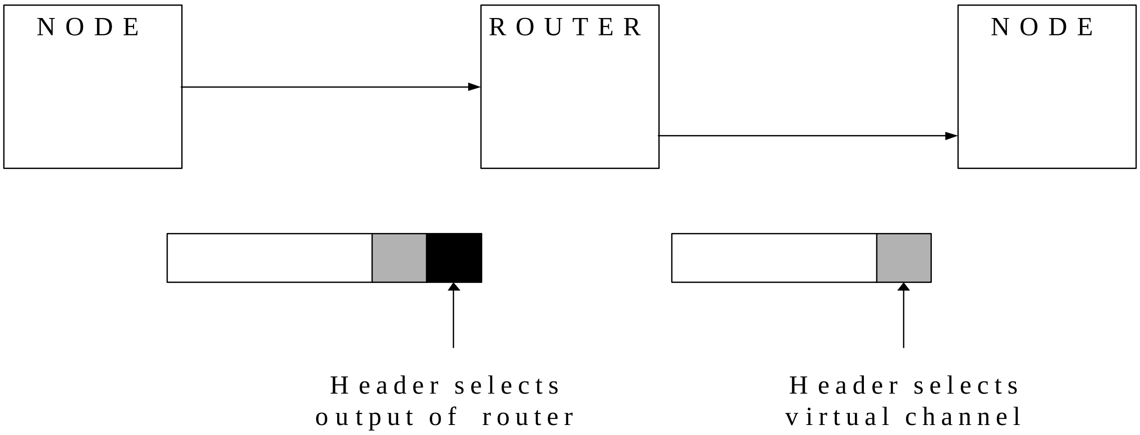

NODE
ROUTER
NODE
Header selects output of router
Header selects virtual channel

Figure 9-3 Header Deletion

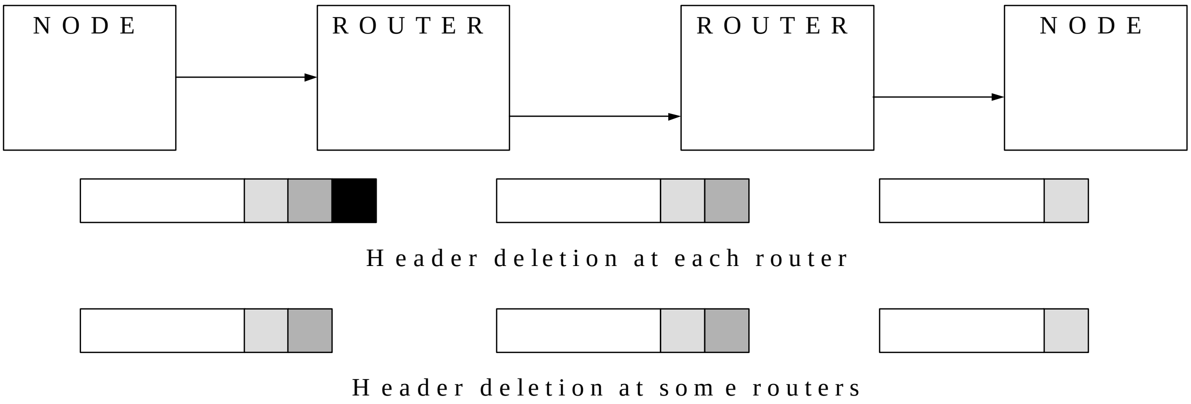

NODE
ROUTER
ROUTER
NODE
Header deletion at each router
Header deletion at some routers

Figure 9-4 Header Deletion Across Multiple Switches

Header deletion can be implemented in all routers in a network or at some selected routing devices only. In the latter case each router must be told beforehand which node addresses require header deletion and which ones do not.

9.1.7 Virtual Channels

Many packets from different sources and going to different destinations may be routed though a single data link. Each source-destination pair forms a virtual channel which is mapped on to the physical network comprising links and routing switches.

This concept can be extended within the source and destination nodes. Consider a processing device attached to a network. There may be several tasks running on the processor. These tasks may need to send or receive information to or from other tasks running on other processors within the network. When a packet arrives at a destination node its header (first byte) can be inspected to see what task it is intended for. The header is stripped off and the packet put in a buffer which may be accessed by the destination task. At any one time the destination task must only expect packets from a single source.

9.1.8 Packet Addressing

In this section several different packet-addressing schemes are considered and compared.

9.1.8.1 Hardware Addressing

With hardware addressing the destination address is specified as a sequence of router output port numbers needed to guide the packet across the network.

Hardware addressing is simple and needs comparatively few gates for implementation. Its drawback is that the destination address can become relatively large if several routing switches have to be traversed. Also the length of the destination address can vary depending on where the destination is located on the network relative to the source. The complexity of packet addressing is handled by the source node and the routing switches are comparatively simple.

9.1.8.2 Logical Addressing

In logical addressing each destination has a unique number or logical address associated with it. These numbers can be assigned arbitrarily to nodes provided that no two nodes have the same logical address. When a source node has to transfer a message to a destination node it simply addresses the packet with the logical address. To support logical addressing each routing switch must contain a routing table. This tells the router what output port a packet is to be transferred to, for each possible logical address. Consider the routing table illustrated in figure 9-5.

Routing Table	
Logical Destination	Physical Output Port
1	8
2	1
3	3
4	1
…	…

Figure 9-5 Example Routing Table

In this example, when a packet is received with a logical address of 1 it is routed to output port 8 of the router. A packet with logical address of either 2 or 4 is routed to output port 1 and a packet with logical address 3 is routed to output port 3.

Logical addressing requires each routing switch to have a routing table. It also means that the routing table will be fairly large (for a reasonable sized network). It is necessary to initialise the routing table in some way, possibly using separate control/configuration links. With logical addressing the complexity of packet addressing is handled by the routing switches rather than by the source node, as is the case with hardware addressing.

9.1.8.3 Regional Logical Addressing

Logical addressing can be used in conjunction with header deletion. In this case the routing table must also hold information about whether the header is to be deleted or not, for each logical address. This facilitates multiple level logical addressing schemes. For local addresses a single logical address is used whereas to send a packet to more remote locations a double logical address is used. In the latter case the first logical address byte represents the destination region and the second byte represents the local address within that region. When the packet arrives at the destination region the first routing switch encountered in the destination region strips off the first address byte making the local address visible for subsequent local routing.

Regional logical addressing can reduce the size of the routing switches needed for logical addressing at the expense of a slightly longer address (two or more data characters), when packets are to be sent to logical addresses in remote regions.

9.1.8.4 Interval Labelling

Interval labelling is based on logical labelling. Destinations are grouped together in contiguous intervals e.g. 1-3, 4-9, 10-32. Each interval is assigned to an output port of the routing switch so that, following the example, destinations 1-3 would all be reached via one particular output port. Interval labelling was designed to reduce the size of the routing tables and to speed up the time taken to decode the destination address within a routing switch.

Interval labelling is more complex than logic labelling but requires smaller routing tables.

9.2 SPACEWIRE ROUTING SWITCHES (NORMATIVE)

This section defines SpaceWire routing switches.

9.2.1 Routing Switch

A SpaceWire routing switch shall comprise a number of SpaceWire link interfaces and a routing matrix. The routing matrix enables packets arriving at one link interface to be transferred to and sent out of another link interface on the routing switch. Each link interface may be considered as comprising an input port (the link interface receiver) and an output port (the link interface transmitter).

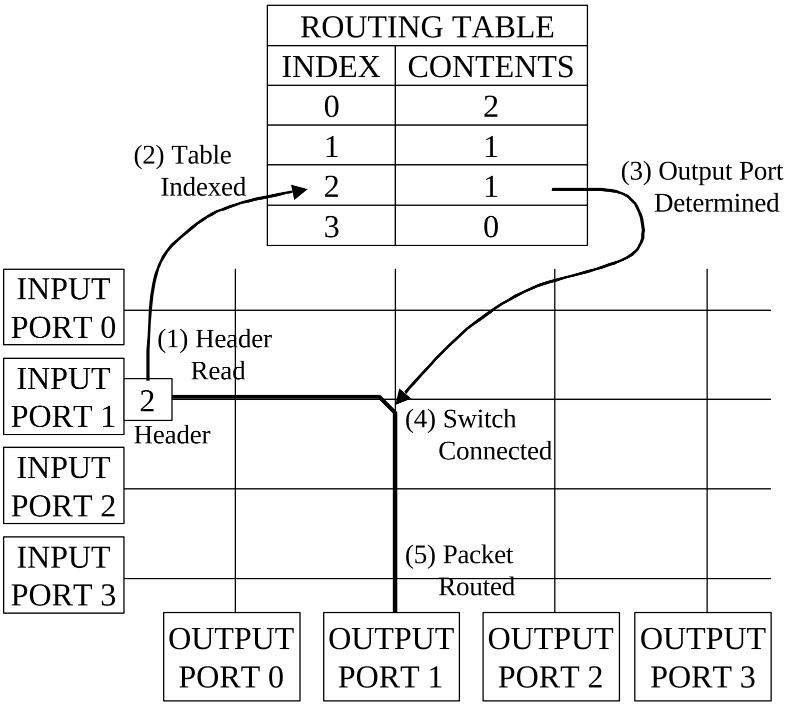
A SpaceWire routing switch shall transfer packets from the input port of the switch where the packet arrives, to a particular output port determined by the packet destination address. The destination address may comprise several data characters (see section 8.1.1). A routing switch shall only use the leading data character of a packet to determine the output port of the routing switch that the packet is to be routed to. The leading data character is the first data character following the EOP or EEP that terminated the previous packet.

9.2.2 Routing Scheme

SpaceWire routing switches shall either support hardware addressing, logical addressing, regional logical addressing or all of these addressing schemes.

If hardware routing is supported then the first data character of a packet shall directly address the output port that the packet is to be routed to. For example, if the packet is to be routed to output port two then the first data character of the packet must be set to two. The first character of the packet is the address of the required output port.

If logical addressing is supported, then the first data character shall indirectly address the output port via a routing table. The value of the first data character may be used as an index into the routing table. The value held in the routing table at the location indexed by the first data character is the address of the output port that the packet is to be routed to. For example, if the routing table contains the values shown in figure 9-6 and the first data character of a packet has the value two, then that packet will be routed to output port one. Alternatively the value of the first data character may be searched for in the contents of a Content Addressable Memory (CAM). The index of the context addressable memory location that matches the first data character, is the address of the output port that the packet is to be routed through.



ROUTING TABLE	
INDEX	CONTENTS
0	2
1	1
2	1
3	0

Figure 9-6 Example: Indirect Output Port Addressing via a Routing table.

9.2.3 Header Deletion

SpaceWire routing switches may implement header deletion. If header deletion is implemented in a routing switch then the leading data character (byte) of each packet shall be deleted. One and only one data character shall be deleted by each routing switch that has header deletion enabled, that is traversed by the packet.

If a routing table is being used within a switch then header deletion may be enabled by a field in the routing table, so that deletion of a header can depend upon the value of that header. This allows regional logical addressing to be realised.

9.2.4 Wormhole Routing

SpaceWire routing switches shall implement wormhole routing. When a packet arrives at a routing switch the required output port is determined. If the output port is free (i.e. not currently transmitting a packet) then the output port shall be allocated to transmit the newly arrived packet. As each character of the packet arrives at the input port it shall be transferred immediately to the output port for transmission. The output port shall not transmit any other packet until the packet that it is currently transmitting has been sent. If the input port has to wait for packet characters to arrive then the output port shall also wait. If the output port has to wait to transmit packet characters then the input port shall also wait.

If the required output port is busy then the newly arrived packet shall wait at the input port until the required output port is free to transmit the new packet. When the output port finishes transmission of its current packet then it shall be available to accept a packet from another input port.

It is possible that two or more input ports could all be waiting to send data out of the same output port. SpaceWire routing switches shall provide a means of arbitrating between input ports that require to send data to the same output port. Priority based, round-robin or random arbitration schemes may be implemented. When the required output port becomes free the input port selected through arbitration shall transfer one packet to the output port. Arbitration shall take place after each packet has been transmitted by an output port that has several input ports waiting to use it.

9.3 SPACEWIRE NODES (NORMATIVE)

A SpaceWire node represents an interface between a SpaceWire network and an application system using the network services. A SpaceWire node comprises one or more SpaceWire link interfaces and an interface to the host system.

A SpaceWire node accepts a stream of packets from the host system for transmission and provides a stream of packets to the host system after reception from the SpaceWire link.

A SpaceWire node may implement virtual channels.

9.3.1 SpaceWire Virtual Channels

This section applies to any SpaceWire node that implements virtual channels.

The link interface output port that a packet provided by the host system is to be sent out from, shall be determined by the first data character of the packet. The first data character shall provide the address of the output port to be used, in a similar arrangement to that adopted in SpaceWire routing switches. A packet prepared by a task running on the host system will contain the complete address of the destination node to which the packet is to be sent. This packet address includes the output port address of the SpaceWire node that the source host system is attached to. Each virtual channel shall be assigned a transmit buffer within the host system into which the application writes one or more complete packets. When there is at least one packet ready for transfer in the host transmit buffer then the host system shall inform the SpaceWire node which will then start to transmit the packet. The SpaceWire node shall inform the host system when the packet has been transferred.

When a packet arrives at an input port of a SpaceWire node the first data character of the packet shall determine the virtual channel that the packet is intended for. Each active virtual channel shall have a receive buffer assigned to it within the host system. When a packet arrives for a particular virtual channel then that packet shall be transferred to the receive buffer for that virtual channel. If the receive buffer becomes full then the input port shall remain blocked until data is read from the host receive buffer by the host system. The host system shall be informed when a complete packet has been transferred to the receive buffer. If a packet arrives addressed for a virtual channel that does not have a receive buffer assigned (i.e. is not active) then the host system shall be informed that a packet has arrived for a particular inactive virtual channel. An input port receiving data for an inactive virtual channel shall remain blocked until that virtual channel is activated by the host system and the complete packet has been transferred to the virtual channel (see section 9.5.3.2).

The SpaceWire node shall perform header deletion when both sending and receiving a packet.

It is possible for a virtual channel to receive data from many different source nodes. It shall be the responsibility of the application level protocols to ensure that a virtual channel receives appropriate data at any particular time.

9.4 SPACEWIRE NETWORK (NORMATIVE)

A SpaceWire network shall comprise two or more SpaceWire nodes and zero or more SpaceWire routing switches. SpaceWire nodes and SpaceWire routing switches shall be interconnected with SpaceWire links. Packets shall be transferred from one SpaceWire node to another across SpaceWire links and through SpaceWire routing switches.

9.5 NETWORK LEVEL ERRORS (NORMATIVE)

Several types of error can occur at the packet level:

- ❑ Link Error,
- ❑ EEP Received,
- ❑ Invalid Destination Address.

9.5.1 Link Error Recovery

When a link error is detected within a link interface the following actions shall be taken at the network level to recover from the error.

1. If an error is detected in a link interface at a source/destination node then the error shall be flagged up to the application level. If an error is detected in a link interface within a routing switch, then the error may be flagged to a pin on the routing switch device or to an internal status register within the routing switch.
2. The current packet being received shall stop being received and the part of the packet that has been received already shall be terminated by an EEP. If a complete packet has just been received when the error occurs (i.e. last character received before error was an EOP or EEP) then no action needs to be taken at the network level other than to flag the error up to the application level.
3. The current packet being transmitted shall stop being transmitted, i.e. the part of the packet that has not yet been transmitted shall not be transmitted i.e. N-Chars up to and including the next EOP or EEP, shall be discarded by the link interface without being transmitted. This is known as spilling the packet. If a complete packet has just been transmitted (i.e. the last character passed to the transmitter was an EOP or EEP) then no action needs to be taken other than to flag the error up to the application level.
4. Packets terminated by an EEP will flow through routing switches within a network (see section 9). When an EEP is received at a destination node the application level shall be informed that an error has occurred.

Note: The application level at a source node may re-send the packet which was being sent when the link error was reported. The application level at a destination node may discard a packet terminated by an EEP, or may decide to use it.

9.5.2 EEP received

An EEP at the end of a packet means that an error occurred within the transmitted packet and that the packet was consequently terminated prematurely. The data in the packet is valid but the complete packet has not been transferred successfully.

If an EEP is received the action taken will depend on whether the link interface is within a source/destination node or within a network routing switch.

9.5.2.1 Routing Switch

An EEP received by a routing switch shall be transferred through the routing switch in the same way as an EOP. EEP is shall be treated in the same way as an EOP within routing switches.

9.5.2.2 Node

An EEP received by a link interface within a source/destination node shall be reported to the application level.

9.5.3 Destination address error

It is possible that a destination address is not recognised by a routing switch or node. For example, a hardware address of 9 may be specified in a routing switch with only eight output ports. The action taken when a packet has an invalid destination address will depend on whether the link interface is within a source/destination node or within a network routing switch.

9.5.3.1 Routing Switch

If a packet arriving at a routing switch has an invalid destination address then that packet shall be spilt i.e. the N-Chars arriving at the input port where the invalid destination address was detected shall be discarded until and including the next EOP or EEP. The invalid destination address error may be flagged to external status pins on the routing switch device or to a status register within the routing switch.

9.5.3.2 Destination Node

A packet arriving at a destination node which supports virtual channels may have an invalid destination address if the address does not match the address of any of the available virtual channels. In this case the destination node shall flag this error to the application layer which should arrange to receive the packet either for later use or so that it may be discarded.

9.5.3.3 Source Node

A packet to be transmitted by a source node, which supports virtual channels, could have an invalid destination address if the packet destination address does not specify a valid output port of the source node. In this case the source node should spill the packet and report the error to the application level.

9.6 EXAMPLE NETWORKS (INFORMATIVE)

Several examples are provided in this section to clarify the operation of packet routing and header deletion.

A SpaceWire network is illustrated in figure 9-7. The SpaceWire nodes are numbered N1 to N9 and each have just one SpaceWire link interface. The SpaceWire routing switches are numbered R1 to R4 and each has four link interfaces.

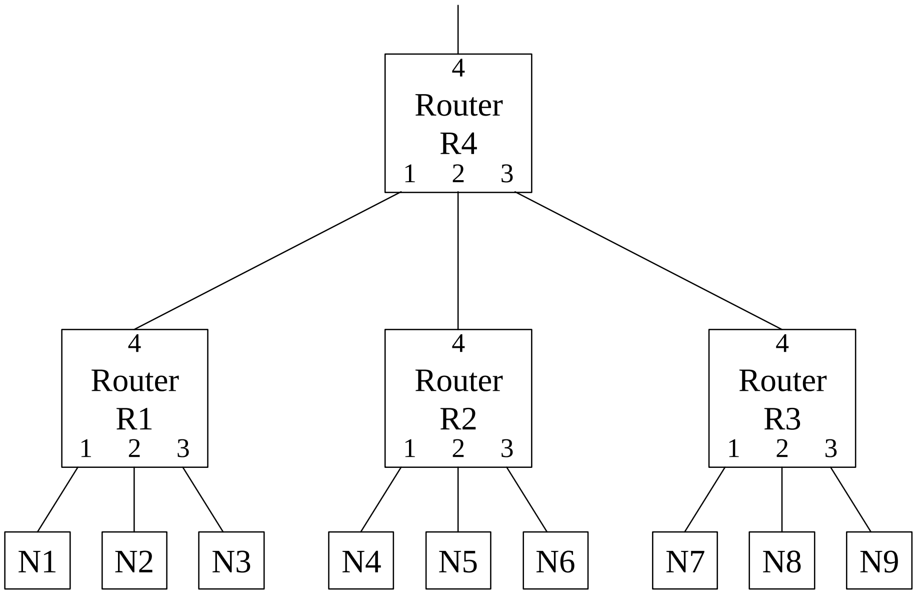


Figure 9-7 Example SpaceWire Network

9.6.1 Hardware Addressing with Header Deletion

In this section the SpaceWire network shown in figure 9-7 is assumed to be configured for hardware addressing with header deletion at each routing switch. One data character is used as the address for each routing switch traversed by a packet. An unlimited number of nodes can be supported.

To transfer a cargo from node 1 to node 3 the following packet must be sent:-

<3><cargo><EOP>

To transfer a cargo from node 1 to node 8 the following packet must be sent:-

<4><3><2><cargo><EOP>

9.6.2 Routing Table without Header Deletion

In this section the SpaceWire network shown in figure 9-7 is assumed to be configured for logical addressing i.e. routing table addressing without header deletion at any routing switch. A single data character holds the logical address of the destination node. Up to 256 nodes can be supported.

To transfer a cargo from node 1 to node 3 the following packet must be sent:-

<3><cargo><EOP>

To transfer a cargo from node 1 to node 8 the following packet must be sent:-

<8><cargo><EOP>

The routing switch contents are shown in figure 9-8.

Routing Table Switch 1	
1	1
2	2
3	3
4	4
5	4
6	4
7	4
8	4
9	4

Routing Table Switch 2	
1	4
2	4
3	4
4	1
5	2
6	3
7	4
8	4
9	4

Routing Table Switch 3	
1	4
2	4
3	4
4	4
5	4
6	4
7	1
8	2
9	3

Routing Table Switch 4	
1	1
2	1
3	1
4	2
5	2
6	2
7	3
8	3
9	3

Figure 9-8 Example SpaceWire Network Routing Table Contents

9.6.3 Routing Table with Partial Header Deletion

In this section the SpaceWire network shown in figure 9-7 is altered slightly to illustrate regional logical addressing i.e. routing table addressing with header deletion at some routing switches. The altered SpaceWire network is shown in figure 9-9. The network is split into two regions. The nodes in region 2 have been renumbered to support the example. The structure of the network is identical to that in figure 9-7. A single data character is used to define the logical address within a region. Almost 256 nodes can be addressed within a single region. Some logical addresses within a region must be used as the bridge to other regions. An unlimited number of regions can be supported.

To transfer a cargo from node 1 to node 3 (both in region 1) the following packet must be sent:-

<3><cargo><EOP>

To transfer a cargo from node 1 to node 5 (both in region 1) the following packet must be sent:-

<5><cargo><EOP>

To transfer a cargo from node 1 (region 1) to node 3 (region 2) the following packet must be sent:-

<7><3><cargo><EOP>

<7> is the logical address given to the link between region 1 and region 2 (R4 port 3 to R3 port 4). R4 deletes the header character on any packet leaving via port 3 or port 4. R3 deletes the header character on any packet leaving via port 4.

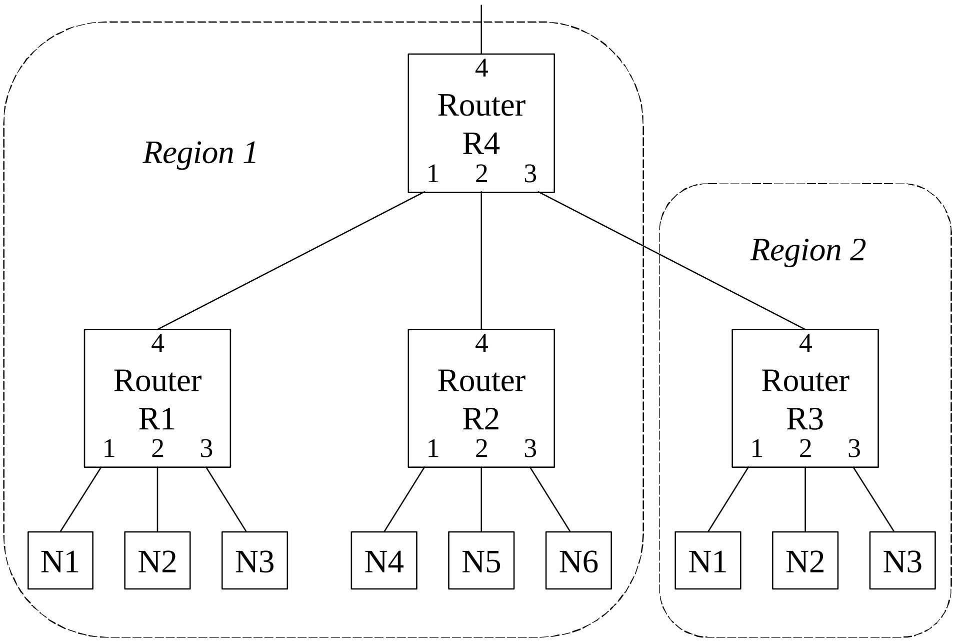


Figure 9-9 Example SpaceWire Network with Local Logical Address Regions

10. ERROR RECOVERY SCHEME (INFORMATIVE)

The error recovery scheme is split across the exchange level, packet level and network level. This section aims to describe the error recovery scheme as a whole, to aid comprehension.

10.1 EXCHANGE LEVEL ERRORS

The following errors can be reported at the exchange level:-

- ❑ Disconnect Error
- ❑ Parity Error
- ❑ Escape Sequence Error
- ❑ Character Sequence Error
- ❑ Credit Error
- ❑ Empty Packet Error

The response to any of these errors is the same:-

1. Detect error
2. Disconnect link
3. Report error to network level
4. Attempt to reconnect link if link interface still enabled.

10.2 NETWORK LEVEL ERRORS

The following errors can be reported at the network level:-

- ❑ Link error (Exchange level error)
- ❑ EEP received
- ❑ Destination address error

10.2.1 Link Error

If a link error (i.e. error detected in exchange level) is reported to the network level then the network level will make the following response.

1. Error reported to network level
2. Terminate current receive packet with EEP
3. Spill current transmit packet until and including next EOP (or EEP)
4. If the error occurs in a link interface within a source/destination node then flag the error to the application layer.
5. If the error occurs in a link interface within a routing switch, then the error may be flagged to external status pins on the routing switch device or to a status register within the device.

This sequence of events is illustrated and described in more detail in section 10.3.

10.2.2 EEP Received

If an EEP is received the action taken will depend on whether the link interface is within a destination node or within a network routing switch. In a destination node, the reception of an EEP is flagged to the application level. The packet terminated by the EEP is otherwise transferred as normal to the application level. In a routing switch no special action is taken when an EEP is received. The EEP is treated in exactly the same way as an EOP.

10.2.3 Destination address error

A packet that arrives at a routing switch with an invalid address (i.e. an address that is not recognised by the routing switch) must be discarded.

When a destination node supports virtual channels and a packet arrives with an invalid (unrecognised) virtual channel address, then this could be the result of an error somewhere. Alternatively, it could be that the application level has not yet set up the virtual channel that is to receive the packet. In either case the error must be flagged to the application level so that the application can receive the packet and decide what to do with it – either to discard it or to use it later.

It is possible for a source node supporting virtual channels to receive a packet from the application level with an invalid address (i.e. one that does not match one of the available output port addresses). The erroneous packet must be spilt and the error reported back to the application level.

10.3 LINK ERROR RECOVERY

If any form of error is detected within the link interface then the following sequence of events shall occur to recover from the error (see figure 10-1):-

1. Detect Error (Parity, Disconnect, Escape sequence, Character sequence, Credit)
2. Disconnect link
3. If previous character was NOT EOP then add EEP (Error End of Packet) to the receiver buffer (i.e. the receive FIFO).
4. Delete data in the transmitter buffer (i.e. transmit FIFO) until the next EOP (End of Packet).
5. Reconnect.
6. Send next packet.

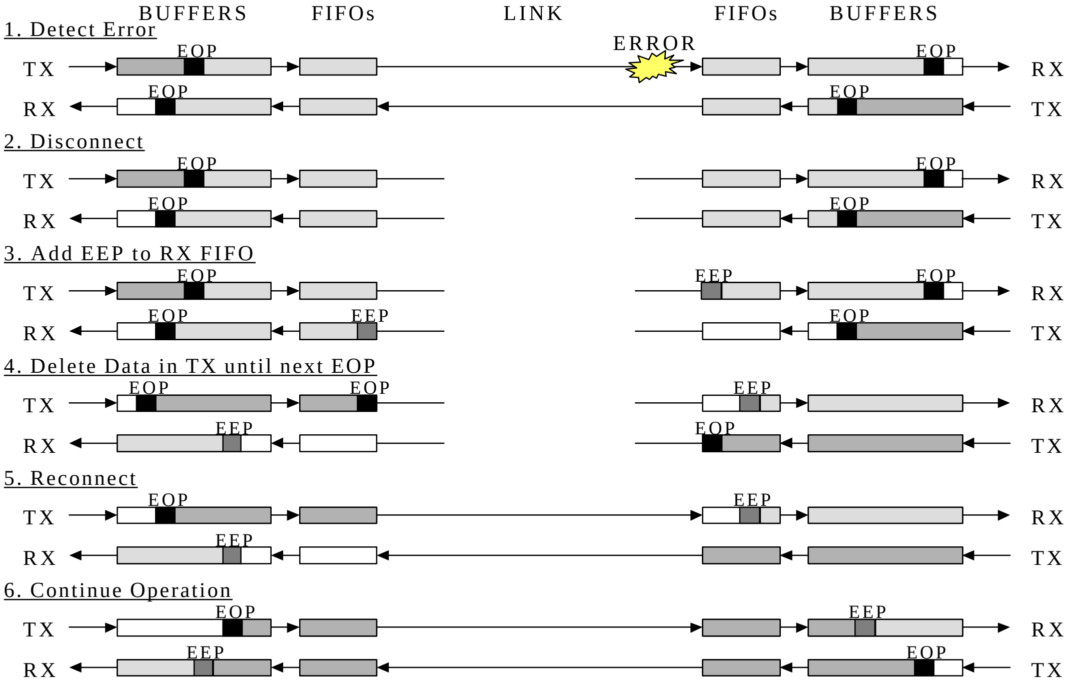


Figure 10-1 Link Interface Error Detection and Recovery Operation

The decision about what to do with the packet that terminates with the EEP is left up to the higher-level application layer. For example if the packet is a line from an image it may be appropriate to use the part of the packet that was received correctly or alternatively to throw the packet away and carry on. If the packet was part of some important control information (e.g. program code) it is important that the packet is resent.

The above protocol works across networks comprised of a number of routing switches. Only the link on which the error occurs is reset (disconnect/reconnect). All the other links continue operation. Only the packet in which the error occurred is partly lost. All other packets remain valid.

If the header byte (i.e. first byte after an EOP or EEP) is corrupted then the entire packet is lost and the data cannot be propagated across a network. The routing switch must simply dispose of the packet. This can cause a problem at the destination of the packet because the fact that a packet is missing will not be reported. It is the responsibility of the application level to ensure that the proper sequence of packets is received. In the event of a receiver at a destination node detecting an error in a header byte it may report this fact to the application level. If virtual channels are being used then the destination virtual channel will be unknown.

If the error occurs in a EOP (or EEP) then two packets will be affected – the one before the EOP where all the data will be sent but no EOP will be received, and the following one because the link receiver will “spill” the packet until the next EOP (or EEP).

If the error occurs in a NULL or FCT inserted in the data stream for a packet then the packet being sent will have to be discarded from that point on. This is because it is not known what the character was before it was corrupted.

If the error occurs in the FCT following the ESC character as part of a NULL code, then the entire NULL code (ESC and FCT) must be replaced by the EEP. If only the FCT is replaced then an ESC followed by EEP will result which is an invalid sequence (escape error).

10.4 APPLICATION LEVEL ERROR HANDLING

The application interface is not defined in the SpaceWire standard. However, a typical application interface will comprise the following services:-

- ❑ Open Link – Starts a link interface and attempts to establish a connection with the link interface at the other end of the link.
- ❑ Close Link – Stops a link and breaks the connection.
- ❑ Write Packet – Sends a packet out of the link interface.
- ❑ Read Packet – Reads a packet from the link interface.
- ❑ Status and Configuration – Reads the current status of the link interface and sets the link configuration.

Several error checks may be implemented at the application level. These are described below.

10.4.1 Link Initialisation Timeout Error

When an application attempts to open a link, it may set a timeout period for link connection. If the link connection has not been established within the specified timeout period then the link may be considered unusable and an alternative link used for the required communication.

10.4.2 Packet Transmit Timeout Error

When an application tries to write a packet to a link interface, it may set a time-out period for transmission of the packet. If the complete packet has not been transmitted when the timeout period expires then the transmitter is assumed blocked. The link interface should be disabled to cause a disconnect error and reset of the link. The link interface should then be enabled again to allow the link to start and reconnect.

10.4.3 Packer Receive Timeout Error

When an application tries to read a packet from a link interface, it may set a timeout period for reception of the packet. If the complete packet has not been received when the time-out period expires then the received is assumed blocked. The link interface should be disabled to cause a disconnect error and reset of the link. The link interface should then be enabled again to allow the link to start and reconnect.

11. CONFORMANCE CRITERIA

11.1 CONFORMANCE STATEMENTS

Several SpaceWire compatible products may be identified each of which implement only a part of the SpaceWire standard:-

- ❑ SpaceWire Cable
- ❑ SpaceWire Connector
- ❑ SpaceWire Cable Assembly
- ❑ SpaceWire Interface
- ❑ SpaceWire Encoder/Decoder
- ❑ SpaceWire LVDS Encoder/Decoder
- ❑ SpaceWire Routing Switch
- ❑ SpaceWire LVDS Routing Switch
- ❑ SpaceWire Routing Switch Unit
- ❑ SpaceWire Network

Corresponding subsets of the SpaceWire standard are defined to which implementations may claim conformance. The conformance statement used should take the form given in the appropriate subset definition.

11.2 DEFINITION OF SUBSETS

11.2.1 SpaceWire Cable

An implementation of SpaceWire cable shall conform to all of the normative specifications given in all of the sections listed in Table 11-1. A cable meeting this specification may use the following conformance statement:-

This cable conforms to the SpaceWire cable specification of the ESA SpaceWire Standard.

Table 11-1 SpaceWire Cable Conformance

Relevant Sections	Title
4.1	Cables

11.2.2 SpaceWire Connector

An implementation of a SpaceWire connector shall conform to all of the normative specifications given in all of the sections listed in table 11-1. A connector meeting this specification may use the following conformance statement:-

This connector conforms to the SpaceWire connector specification of the ESA SpaceWire Standard.

Table 11-2 SpaceWire Connector Conformance

Relevant Sections	Title
4.2	Connectors

11.2.3 SpaceWire Cable Assembly

An implementation of a SpaceWire cable assembly shall conform to all of the normative specifications given in all of the sections listed in Table 11-3. A cable assembly meeting this specification may use the following conformance statement:-

> This cable assembly conforms to the SpaceWire cable assembly specification of the ESA SpaceWire Standard.

Table 11-3 SpaceWire Cable Assembly Conformance

Relevant Sections	Title
4.3	Cable Assembly

11.2.4 SpaceWire Interface

An implementation of a SpaceWire interface shall conform to all of the normative specifications given in all of the sections listed in Table 11-4. A product fitted with an interface meeting this specification may use the following conformance statement:-

This product conforms to the SpaceWire interface specification of the ESA SpaceWire Standard.

Table 11-4 SpaceWire Interface Conformance

Relevant Sections	Title
4.2	Connectors
4.4	PCB Tracks
5	Signal Level
6	Character Level
7	Exchange Level
8	Packet Level
9.3	SpaceWire Nodes
9.5	Network Level Errors

Together with the above conformance statement the following parameters shall be specified for the interface.

1. Total transmitter data-strobe skew (worst case[3]) measured at the interface connector (Dout to Sout skew).
2. Total transmitter data jitter (worst case) measured at the interface connector (Dout jitter).

[3] Worst case over process, temperature, voltage range and irradiation.

3. Total transmitter strobe skew (worst case) measured at the interface connector (Sout jitter).
4. Total receiver minimum separation between data and strobe (worst case) measured at the interface connector (Din to Sin minimum separation[4]).

Typical figures for the above parameters may also be provided, in which case the conditions applying for the typical figure measurements must be clearly stated (e.g. temperature, operating voltage).

A detailed explanation of the above parameters is provided in section 5.

11.2.5 SpaceWire Encoder/Decoder

An implementation of a SpaceWire encoder/decoder shall conform to all of the normative specifications given in all of the sections listed in Table 11-5. A product fitted with an interface meeting this specification may use the following conformance statement:-

This product conforms to the SpaceWire encoder/decoder specification of the ESA SpaceWire Standard.

Table 11-5 SpaceWire Encoder/Decoder Conformance

Relevant Sections	Title
5.3	Signal Coding
5.5	SpaceWire Link
5.6	Data Signalling Rate
6	Character Level
7	Exchange Level
8	Packet Level

Together with the above conformance statement the following parameters shall be specified for the interface.

1. Encoder data-strobe skew (worst case[5]) measured at the output of the encoder device.
2. Encoder data jitter (worst case) measured at the output of the encoder device.
3. Encoder strobe skew (worst case) measured at the output of the encoder device.
4. Decoder minimum separation between data and strobe (worst case) measured at the input of the decoder device.

Typical figures for the above parameters may also be provided, in which case the conditions applying for the typical figure measurements must be clearly stated (e.g. temperature, operating voltage).

A detailed explanation of the above parameters is provided in section 5.

11.2.6 SpaceWire LVDS Encoder/Decoder

An implementation of a SpaceWire encoder/decoder which include the LVDS drivers and receivers shall conform to all of the normative specifications given in all of the sections listed in Table 11-6. A product fitted with an interface meeting this specification may use the following conformance statement:-

[4] This must include all D-S skew, D jitter and S jitter between the interface connector and the decoder device.

[5] Worst case over process, temperature, voltage range and irradiation.

This product conforms to the SpaceWire LVDS encoder/decoder specification of the ESA SpaceWire Standard.

Table 11-6 SpaceWire LVDS Encoder/Decoder Conformance

Relevant Sections	Title
5	Signal Level
6	Character Level
7	Exchange Level
8	Packet Level

Together with the above conformance statement the following parameters shall be specified for the interface.

1. Encoder data-strobe skew (worst case[6]) measured at the output of the encoder device.
2. Encoder data jitter (worst case) measured at the output of the encoder device.
3. Encoder strobe skew (worst case) measured at the output of the encoder device.
4. Decoder minimum separation between data and strobe (worst case) measured at the input of the decoder device.

Typical figures for the above parameters may also be provided, in which case the conditions applying for the typical figure measurements must be clearly stated (e.g. temperature, operating voltage).

A detailed explanation of the above parameters is provided in section 5.

11.2.7 SpaceWire Routing Switch

An implementation of a SpaceWire routing switch shall conform to all of the normative specifications given in all of the sections listed in Table 11-7. A routing switch meeting this specification may use the following conformance statement:-

This product conforms to the SpaceWire routing switch specification of the ESA SpaceWire Standard.

Table 11-7 SpaceWire Routing Switch Conformance

Relevant Sections	Title
5.3	Signal Coding
5.5	SpaceWire Link
5.6	Data Signalling Rate
6	Character Level
7	Exchange Level
8	Packet Level
9.2	SpaceWire Routing Switches
9.5	Network Level Errors

[6] Worst case over process, temperature, voltage range and irradiation.

Together with the above conformance statement the following parameters shall be specified for the routing switch.

1. SpaceWire link encoder data-strobe skew (worst case[7]) measured at an output of the routing switch device.
2. SpaceWire link encoder data jitter (worst case) measured at an output of the routing switch device.
3. SpaceWire link encoder strobe skew (worst case) measured at an output of the routing switch device.
4. SpaceWire link decoder minimum separation between data and strobe (worst case) measured at an input of the routing switch device.

Typical figures for the above parameters may also be provided, in which case the conditions applying for the typical figure measurements must be clearly stated (e.g. temperature, operating voltage).

A detailed explanation of the above parameters is provided in section 5.

11.2.8 SpaceWire LVDS Routing Switch

An implementation of a SpaceWire routing switch which includes the LVDS drivers and receivers shall conform to all of the normative specifications given in all of the sections listed in Table 11-8. A routing switch meeting this specification may use the following conformance statement:-

This product conforms to the SpaceWire LVDS routing switch specification of the ESA SpaceWire Standard.

Table 11-8 SpaceWire LVDS Routing Switch Conformance

Relevant Sections	Title
5	Signal Level
6	Character Level
7	Exchange Level
8	Packet Level
9.2	SpaceWire Routing Switches
9.5	Network Level Errors

Together with the above conformance statement the following parameters shall be specified for the routing switch.

1. SpaceWire link encoder data-strobe skew (worst case[8]) measured at an output of the routing switch device.
2. SpaceWire link encoder data jitter (worst case) measured at an output of the routing switch device.
3. SpaceWire link encoder strobe skew (worst case) measured at an output of the routing switch device.
4. SpaceWire link decoder minimum separation between data and strobe (worst case) measured at an input of the routing switch device.

[7] Worst case over process, temperature, voltage range and irradiation.

[8] Worst case over process, temperature, voltage range and irradiation.

Typical figures for the above parameters may also be provided, in which case the conditions applying for the typical figure measurements must be clearly stated (e.g. temperature, operating voltage).

A detailed explanation of the above parameters is provided in section 5.

11.2.9 SpaceWire Routing Switch Unit

An implementation of a SpaceWire routing switch unit shall conform to all of the normative specifications given in all of the sections listed in Table 11-9. A routing switch unit meeting this specification may use the following conformance statement:-

This product conforms to the SpaceWire routing switch unit specification of the ESA SpaceWire Standard.

Table 11-9 SpaceWire LVDS Routing Switch Conformance

Relevant Sections	Title
4.2	Connectors
4.4	PCB Tracks
5	Signal Level
6	Character Level
7	Exchange Level
8	Packet Level
9.2	SpaceWire Routing Switches
9.5	Network Level Errors

Together with the above conformance statement the following parameters shall be specified for the routing switch.

1. SpaceWire link encoder data-strobe skew (worst case[9]) measured at an output of the routing switch device.
2. SpaceWire link encoder data jitter (worst case) measured at an output of the routing switch device.
3. SpaceWire link encoder strobe skew (worst case) measured at an output of the routing switch device.
4. SpaceWire link decoder minimum separation between data and strobe (worst case) measured at an input of the routing switch device.

Typical figures for the above parameters may also be provided, in which case the conditions applying for the typical figure measurements must be clearly stated (e.g. temperature, operating voltage).

A detailed explanation of the above parameters is provided in section 5.

11.2.10 SpaceWire Network

An implementation of a SpaceWire network shall conform to all of the normative specifications given in all of the sections listed in Table 11-9. A network meeting this specification may use the following conformance statement:-

[9] Worst case over process, temperature, voltage range and irradiation.

This network conforms to the SpaceWire network specification of the ESA SpaceWire Standard.

Table 11-9 SpaceWire Network Conformance

Relevant Sections	Title
4	Physical Level
5	Signal Level
6	Character Level
7	Exchange Level
8	Packet Level
9	Network Level

Together with the above conformance statement the following parameters shall be specified for the routing switch.

1. SpaceWire link encoder data-strobe skew (worst case[10]) measured at an output of the routing switch device.
2. SpaceWire link encoder data jitter (worst case) measured at an output of the routing switch device.
3. SpaceWire link encoder strobe skew (worst case) measured at an output of the routing switch device.
4. SpaceWire link decoder minimum separation between data and strobe (worst case) measured at an input of the routing switch device.

Typical figures for the above parameters may also be provided, in which case the conditions applying for the typical figure measurements must be clearly stated (e.g. temperature, operating voltage).

A detailed explanation of the above parameters is provided in section 5.

11.2.11

[10] Worst case over process, temperature, voltage range and irradiation.

Annex A (Informative) SpaceWire Requirement s

The initial requirements for the physical and signalling layers of the SpaceWire data link as defined during the DICE study [RD11] are given in this section.

A.1 DATA SIGNALLING RATE AND CABLE LENGTH

A.1.1 Maximum Signalling Rate

A maximum signalling rate of at least 100 Mbits per second (100 Mbaud) shall be provided over a single differential pair.

Rationale: Many instruments already have data rates approaching 100Mbits per second or higher. Future instruments are likely to require data rates of up to 1Gbit per second. To be useful in future as a high-speed data link at least 100Mbaud should be supported. Higher data rates may be implemented by using several data links in parallel. The overhead of a serial link protocol will reduce the data rate supported by a single link compared to its baud rate.

Need: Essential

A.1.2 Minimum Signalling Rate

A minimum signalling rate of less that one tenth of the maximum signalling rate shall be supported.

Rationale: The minimum signalling rate will depend upon the signalling method used (e.g. Data/Strobe) and the specific interface components. Depending upon the particular drivers/receivers used, reducing the signalling rate when no data is being sent may reduce system power consumption.

Need: Desirable

A.1.3 Maximum Cable Length

The maximum signalling rate (R1.1) shall apply over a maximum cable length of at least 10m.

Rationale: A maximum distance of 10m should be adequate for most spacecraft applications, given the size of typical spacecraft.

Need: Essential

A.2 EMC

These requirements have been derived from the Rosetta and Envisat EMC requirements specifications.

A.2.1 Conducted Emission

The data link shall not emit common mode conductive interference appearing on adjacent signal lines.

TBC

A.2.2 Conducted Susceptibility

The data link receiver circuits shall have immunity to common mode interference levels of

± 360mV (1kHz square wave injected into signal lines).

Rationale: This requirement is taken from the Rosetta EMC Requirements Specification (R5.1.2.3-1).

Need: Essential

A.2.3 Radiated Emission, Electric Field

A single data link shall not radiate electric fields, measured at 1m distance, in excess of the levels given in Table A-1 below:-

Table A-1: Radiated Electric Field Emission Requirements

Reference	Frequency Range	Field Strength	Bandwidth
A.2.3-1	10kHz – 1GHz	40 dBuV/m (rms)	Narrow band
A.2.3-2	1GHz – 10GHz	60 dBuV/m (rms)	Narrow band
A.2.3-3	10kHz – 1GHz	70 dBuV/m (rms)	Broad band

Rationale: These requirements are in line with the radiated emission requirement for Rosetta (Rosetta EMC Requirements Specification R5.2.1.1-1, which gives a minimum requirement of 50dBuV/m) and the sub-system requirements for ENVISAT (ENVISAT EMC Requirements Specification section 4.4.1, which give a requirement of 44dBuV/m narrowband and 74dBuV/m broadband, 10kHz – 1GHz).

Need: Essential

A.2.4 Radiated Susceptibility, Electric Field

The data link shall not exhibit any malfunction, degradation of performance or deviation from specified parameters when irradiated with an Electric Field of strength up to the levels given in Table A-2 below:-

Table A-2: Radiated Electric Field Susceptibility Requirements

Reference	Frequency Band	Strength	Modulation
A.2.4-1	14kHz –1GHz	1V/m (rms)	30-80%AM, 1kHz squarewave
A.2.4-2	1GHz – 40GHz	20V/m (rms)	Pulsed 1kHz PRF, 10-30% duty cycle

Rationale: These requirements are in line with those of Rosetta (Rosetta EMC Requirements Specification R5.2.2.1-1) and ENVISAT (ENVISAT EMC Requirements Specification section 4.5.1).

Need: Essential

A.2.5 Radiated Emission, Magnetic Field

The single data link shall not radiate magnetic fields, measured at 1m distance, in excess of the limits given in Table A-3 below:-

Table A-3: Radiated Magnetic Field Emission Requirements

Reference	Frequency Range	Field Strength	Bandwidth
A.2.5-1	AC (30Hz – 50kHz)	≤ 50dBpT	Narrow band
A.2.5-2	DC (Static)	≤ 200nT	Static

Rationale: These requirements are in line with the magnetic radiated emission requirements for Rosetta (Rosetta EMC Requirements Specification R5.2.1.2-1) and ENVISAT (ENVISAT EMC Requirements Specification section 4.4.2).

Need: Essential

A.2.6 Radiated Susceptibility, Magnetic Field

The data link shall not exhibit any malfunction, degradation of performance or deviation from specified parameters when irradiated with a Magnetic Field of strength up to the levels given in Table A-4 below:-

Table A-4: Radiated Magnetic Field Susceptibility Requirements

Reference	Frequency Band	Strength
A.2.6-1	AC (30Hz-500Hz)	< 136dBpT
A.2.6-2	AC (500-50kHz)	< 120dBpT
A.2.6-3	DC (static)	155dBpT (Earth Field)

Rationale: These requirements are in line with those of Rosetta (Rosetta EMC Requirements Specification R5.2.2.2-1) and ENVISAT (ENVISAT EMC Requirements Specification section 4.5.2). The DC magnetic field susceptibility requirement (R2.5-2) represents operation in the Earth's magnetic field and need not be tested explicitly.

Need: Essential

A.3 ESD

A.3.1 ESD Rating

Driver and receiver devices shall have ESD protection for greater than 3000 Volts.

Rationale: This is in-line with standard CMOS components. Normal antistatic handling procedures will be required to prevent any ESD damage to driver and receiver devices.

Need: Essential

A.4 ERROR RATE

A.4.1 Bit Error Rate (BER)

A bit error rate of less than 10^{-12} shall be achieved over a distance of 10m using the recommended cable type, connectors, and PCB trace layout.

Rationale: At a signalling rate of 100 Mbaud this corresponds to one error every 10^{4} seconds or 2.78 hours. A higher level error detection and correction protocol must be provided to reduce the error rate below this level. This is provided by IEEE-1355.

Need: Essential

A.5 RADIATION

A.5.1 Total Dose

A radiation total dose of greater than 100krad (Si) shall be tolerated.

Rationale: This would make the LVDS devices suitable for most space applications.

Need: Desirable – this is a target specification, lower radiation performance values may be acceptable (TBD).

A.5.2 Single Event Upset

The single event upset threshold (SEU LET) shall be greater 30MeV/mg/cm^2.

Rationale: This would make the LVDS devices suitable for most space applications.

Need: Desirable – this is a target specification, lower radiation performance values may be acceptable (TBD).

A.5.3 Latch-Up

The single event latch-up threshold (SEL LET) shall be greater 100MeV/mg/cm^2.

Rationale: This would make the LVDS devices suitable for most space applications.

Need: Desirable – this is a target specification, lower radiation performance values may be acceptable (TBD).

A.6 POWER CONSUMPTION

A.6.1 Link Power Consumption

A driver/receiver pair together with necessary termination and biasing components, implementing a single uni-direction link, shall have a typical power consumption of less than 0.1 Watt.

Rationale: The aim as far as power consumption is concerned is to significantly improve over the high power consumption of ECL, PECL and other high speed interface devices.

Need: Essential

A.7 FAIL SAFE / FAULT ISOLATION

A.7.1 Receiver Inputs Open

When the receiver inputs are open circuit the receiver output shall remain in a known logic state (either HIGH or LOW). The receiver output must not oscillate.

Rationale: With some logic families (e.g. CMOS) an open circuit input will float to somewhere around the logic threshold level. Any noise on the input will then cause the receiver output to oscillate. This results in high power consumption in the receiver and unpredictable behaviour of the system to which it is connected.

Need: Essential

A.7.2 Receiver Inputs Shorted

When the receiver inputs are short circuited the receiver output shall remain in a known logic state (either HIGH or LOW) and the receiver shall not be damaged.

Rationale: A short circuit must not cause damage to the receiver or unpredictable behaviour of the system to which it is connected.

Need: Essential

A.7.3 Driver Outputs Open

When the driver outputs are open circuit no damage shall result to the driver.

Rationale: It is possible that a driver may be powered and enabled before a receiver and termination resistor is connected to it. In this case the driver should not be damaged.

Need: Essential

A.7.4 Driver Outputs Shorted

When the driver outputs are shorted together or to ground no damage shall result to the driver.

Rationale: If a fault occurs which shorts the outputs of the driver together or to ground, then the driver should not be damaged or propagate the fault.

Need: Essential

A.7.5 Driver Powered, Receiver Not Powered

When the driver is powered and the receiver is not powered neither the driver nor the receiver shall be damaged.

Rationale: In a cold redundant system active signals may be connected to a unit which is powered down. In this case no damage should result to either the driver or receiver circuits. NOTE: normally the driver would be tri-stated to prevent any unnecessary power consumption.

Need: Essential

A.7.6 Driver Not Powered, Receiver Powered

When the driver is not powered and the receiver is powered neither the driver nor the receiver shall be damaged and the receiver output shall remain in a known logic state (either HIGH or LOW).

Rationale: In a cold redundant system a receiver may be powered and the driver not powered. In this case it is important that the receiver remains in a known logic state to prevent unpredictable behaviour of the system to which the receiver is connected.

Need: Essential

A.7.7 Driver Tri-State

It shall be possible to tri-state the driver.

Rationale: This is required to prevent unnecessary power consumption in cold redundant systems.

Need: Essential

A.8 MASS AND SIZE

A.8.1 Driver / Receiver Packaging

A small outline device package containing several drivers and/or receivers shall be used.

Rationale: Small outline packages with several drivers and/or receivers per package will save on size and mass.

Need: Desirable

A.8.2 Driver / Receiver Integration

Driver and receiver circuits that can be integrated with higher-level system functions are preferred.

Rationale: Integrating the drivers and receivers with the higher-level functions will eliminate the separate driver/receiver components saving mass and board area.

Need: Desirable

A.8.3 Line Termination

The line termination shall normally require no more than three components.

Rationale: A single resistor should suffice for line termination unless centre tap capacitance termination is used to filter common mode noise. This requires a pair of resistors and a capacitor.

Need: Desirable

A.8.4 Driver/Receiver Power

The driver and receiver circuits shall run off the same power supplies as the system to which they are connected, unless galvanic isolation is required.

Rationale: Extra power supplies will add to system mass, size and complexity.

Need: Desirable

A.9 MAGNETIC EMISSION

A.9.1 Non-Ferrous Materials

Ferrous materials shall NOT be used in any component forming the data link.

Rationale: Magnetic emission is an important consideration for some missions. To support these missions no magnetic materials should be used in the data link components.

Need: Desirable

A.10 GALVANIC ISOLATION

A.10.1 Galvanic Isolation

It should be possible to galvanically isolate two systems connected via the data link.

Rationale: Some space instruments or sub-systems require to be galvanically isolated. This is not generally the case but will depend upon specific mission requirements. The data link should be capable of supporting galvanic isolation through the addition of extra isolation circuitry.

Need: Desirable

A.11 REQUIREMENTS SUMMARY

A table summarising the requirements for space-link is given below:

Table A-5 Requirements summary

Ref.	Title	Requirement Summary
A.1.1	Maximum signalling Rate	> 100 Mbaud
A.1.2	Minimum Signalling Rate	< 0.1 x Maximum Signalling Rate
A.1.3	Maximum Cable Length	> 10m at 100 Mbaud
A.2.1	Conducted Emission	TBC
A.2.2	Conducted Susceptibility	Immunity to ±360mV 1kHz. Sq. wave
A.2.3-1	Radiated Emission, Electric Field	< 40dBuV/m rms (10kHz – 1GHz, narrow band)
A.2.3-2	Radiated Emission, Electric Field	< 60dBuV/m rms (1GHz – 10GHz, narrow band)
A.2.3-3	Radiated Emission, Electric Field	< 70dBuV/m rms (10kHz – 1GHz, broad band)
A.2.4-1	Radiated Susceptibility, Electric Field	immunity to 1 V/m rms (14kHz – 1 GHz, 1kHz sq.wave)
A.2.4-2	Radiated Susceptibility, Electric Field	immunity to 20V/m rms (1GHz – 40GHz, Pulsed 1kHz PRF)
A.2.5-1	Radiated Emission, Magnetic Field	< 50dBpT (30Hz – 50kHz, narrow band)
A.2.5-2	Radiated Emission, Magnetic Field	< 200nT (DC)
A.2.6-1	Radiated Susceptibility, Magnetic Field	< 136dBpT (30Hz-500Hz)
A.2.6-2	Radiated Susceptibility, Magnetic Field	< 120dBpT (500Hz-50kHz)
A.2.6-3	Radiated Susceptibility, Magnetic Field	< 155dBpT (DC, Earth Field)
A.3.1	ESD Rating	> 3000 V
A.4.1	Bit Error Rate(BER)	< 10^{-12} over 10m
A.5.1	Total Dose	> 100krad (Si)
A.5.2	Single Event Upset Threshold	> 30 MeV/mg/cm^2
A.5.3	Latch-Up Threshold	> 100 MeV/mg/cm^2
A.6.1	Link Power Consumption	< 0.1 W (uni-directional Link)
A.7.1	Receiver Inputs Open	Known output state, no oscillation
A.7.2	Receiver Inputs shorted	Known output state, no damage to receiver
A.7.3	Driver Outputs Open	No damage to driver
A.7.4	Driver Outputs Shorted	No damage to driver
A.7.5	Driver Powered, Receiver Not Powered	No damage to driver or receiver
A.7.6	Driver Not Powered, Receiver Powered	No damage to driver or receiver and known output state
A.7.7	Driver Tri-State	Possible to tri-state driver
A.8.1	Driver / Receiver Packaging	Small outline package with multiple drivers and/or receivers
A.8.2	Driver / Receiver Integration	Possible to integrate with encoder etc.
A.8.3	Line Termination	No more than three components
A.8.4	Driver/Receiver Supply Voltage	Same Vcc as system
A.9.1	Non-Ferrous Materials	No ferrous material to be used
A.10.1	Galvanic Isolation	Possible to galvanically isolate two systems connected by link

Annex B (Normative) SpaceWire EMC Performance

This section provides an evaluation of SpaceWire (in particular LVDS) against the data link requirements set out in the Annex A report. This testing was performed as part of the DICE study. Full details of the EMC test results are reported in [RD15].

The following table (Table B-1) lists the requirements and considers whether LVDS meets each requirement. A double tick ( ) in the "OK" column of the table means that tests have shown conclusively that LVDS meets a requirement. A single tick ( ) means that the performance is thought to be satisfactory although the requirement has not been met in full. TBC in the table means that the tests were inconclusive or that the requirement has not been tested during the current DICE study.

The bit error rate (BER) test has checked both LVDS and IEEE-1355 i.e. the complete SpaceWire link. The EMC testing has tested the LVDS part of the data link using where appropriate IEEE-1355 like data patterns.

Table B-1 Does LVDS Meet The Requirements

Ref.	Title	Requirement Summary	Test Result	OK?
A.1.1	Maximum signalling Rate	> 100Mbaud	LVDS eye diagram results indicated a maximum performance of 200Mbaud	√√
A.1.2	Minimum Signalling Rate	< 0.1 x Maximum Signalling Rate	Not Tested	√√
A.1.3	Maximum Cable Length	> 10m at 100 Mbaud	10m requirement achieved at 100Mbaud	√√
A.2.1	Conducted Emission	TBD – depends on particular power supply arrangement used.	Cross-coupling from signal to power supply measured at around 4mA i.e. very low.	√
A.2.2	Conducted Susceptibility	Immunity to ±360mV 1kHz. Sq. wave	AC and DC susceptibility tested	√√
A.2.3-1	Radiated Emission, Electric Field	< 40dBuV/m rms (10kHz – 1GHz, narrow band)	Problem with interference from pattern generator.	√ TBC
A.2.3-2	Radiated Emission, Electric Field	< 60dBuV/m rms (1GHz – 10Ghz, narrow band)	Requirement met.	√√
A.2.3-3	Radiated Emission, Electric Field	< 70dBuV/m rms (10kHz – 1GHz, broad band)	Not tested.	TBC
A.2.4-1	Radiated Susceptibility, Electric Field	immunity to 1 V/m rms (14kHz – 1 GHz, 1kHz sq.wave)	Requirement met	√√
A.2.4-2	Radiated Susceptibility, Electric Field	immunity to 20V/m rms (1GHz – 40GHz, Pulsed 1kHz PRF)	Tested in range 1 GHz to 20GHz with field strength of at least 4.5V/m	√
A.2.5-1	Radiated Emission, Magnetic Field	< 50dBpT (30Hz – 50kHz, narrow band)	Requirement met across frequency band.	√√
A.2.5-2	Radiated Emission, Magnetic Field	< 200nT (DC)	Not tested – DC magnetic emission is caused by the use of magnetic materials see R9.1	√
A.2.6-1	Radiated Susceptibility, Magnetic Field	< 136dBpT (30Hz-500Hz)	Requirement met.	√√
A.2.6-2	Radiated Susceptibility, Magnetic Field	< 120dBpT (500Hz-50kHz)	Requirement met.	√√
A.2.6-3	Radiated Susceptibility, Magnetic Field	< 155dBpT (DC, Earth Field)	Indirectly Tested – All tests performed in Earth's magnetic field.	√√
A.3.1	ESD Rating	> 3000 V	Specified by LVDS device manufacturer.	√
A.4.1	Bit Error Rate (BER)	< 10^{-12} over 10m	Achieved better than 10^{-13} over 21m under normal lab conditions.	√ TBC
A.5.1	Total Dose	> 100krad (Si)	50krad for receiver. Driver not tested.	TBC
A.5.2	Single Event Upset	> 30 MeV/mg/cm^2	20 MeV/mg/cm^2 for	TBC

			receiver.	
A.5.3	Latch-UP	> 100 MeV/mg/cm^2	>77.3 MeV/mg/cm^2 for receiver. Driver showed latch-up at 27.5 MeV/mg/cm^2 .	TBC
A.6.1	Link Power Consumption	< 0.1 W (uni-directional Link)	Measured as 38.5 mW at room temp., 5.5V supply operating at 100 Mbits/s	√√
A.7.1	Receiver Inputs Open	Known output state, no oscillation	Output high. Oscillation sometimes occurred – though to be due to oscilloscope probes.	√
A.7.2	Receiver Inputs shorted	Known output state, no damage to receiver	Output high. No damage.	√√
A.7.3	Driver Outputs Open	No damage to driver	No damage.	√√
A.7.4	Driver Outputs Shorted	No damage to driver	No damage.	√√
A.7.5	Driver Powered, Receiver Not Powered	No damage to driver or receiver	No damage.	√√
A.7.6	Driver Not Powered, Receiver Powered	No damage to driver or receiver and known output state	Output high. No damage.	√√
A.7.7	Driver Tri-State	Possible to tri-state driver	Facility provided on LVDS driver device.	√√
A.8.1	Driver / Receiver Packaging	Small outline package with multiple drivers and/or receivers	Not Tested – the LVDS devices used have four drivers or receivers in each small outline package	√√
A.8.2	Driver / Receiver Integration	Possible to integrate with encoder etc.	Not Tested – LVDS can be implemented in a range of different technologies.	√√
A.8.3	Line Termination	No more than three components	Indirectly Tested – single resistor termination used for all tests.	√√
A.8.4	Driver/Receiver Supply Voltage	Same Vcc as system	Indirectly Tested – LVDS devices used operate with a single supply rail at a nominal 5V.	√√
A.9.1	Non-Ferrous Materials	No ferrous material to be used	Not Tested – cables and connector must be specified to NOT have magnetic materials.	√√
A.10.1	Galvanic Isolation	Possible to galvanically isolate two systems connected by link	Not Tested – A circuit suitable for the galvanic isolation of LVDS has been identified (patent owed by TI).	√√

Annex C (Informative) Differences Between Spa ceWire and IEEE-1355

There are several differences between SpaceWire and IEEE 1355-1995 [RD1]. Improvements have been made in the SpaceWire standard to improve ruggedness, power consumption, EMC performance, and to eliminate problems and ambiguities that exist with IEEE 1355.

The differences between the two standard and the reasons for them are detailed below looking at each level of the SpaceWire standard in turn.

C.1 PHYSICAL LEVEL

- The SpaceWire cable is designed to be suitable for space applications. IEEE 1355 cable is not suitable.
- SpaceWire connectors are 9-way micro-miniature D-type connectors which are used in space applications. The IEEE 1355 connector is not rugged enough for space use.

C.2 SIGNAL LEVEL

- LVDS adopted for SpaceWire provides improved electromagnetic emission characteristics compared to the PECL signals used in IEEE 1355. LVDS supports failsafe operation, which is not the case with PECL. LVDS can be implemented in a range of semiconductor technologies. This enables integration of completed SpaceWire interfaces with other system functions.
- The DS encoding used by SpaceWire is identical to IEEE 1355 with the exception that SpaceWire interfaces must be tolerant of simultaneous transitions on data and strobe signals. This is not a requirement of IEEE 1355 and may lead to faults occurring within the interface.
- The SpaceWire timing specification is tightened up compared to that in IEEE 1355. IEEE 1355 only considers skew whereas SpaceWire considers both jitter and skew.

C.3 CHARACTER LEVEL

- The character level protocol for SpaceWire is identical to that in IEEE 1355.
- See also minor differences section 96.

C.4 EXCHANGE LEVEL

- IEEE 1355 section 5.7.2 states "Thereafter it shall send only NULLs unless and until at least one character has been received by the corresponding link input since reset. After the link output has been started and at least one character has been received by the corresponding link input since reset, the link shall begin normal operation." The state diagram in Figure 5-11 and timing sequence diagram in Figure 5-12 of IEEE 1355 show a different situation. Instead of one character being received it is specifically a NULL that must be received. This ambiguity is resolved in the SpaceWire standard – a NULL must be received.
- IEEE 1355 section 5.7.4.2 states "If a link interface detects a disconnect error before it has started, it shall start, transmit at least one character, and then halt, to ensure that a disconnection error is also detected by the other end …" The state diagram in Figure 5-11 of IEEE 1355 shows a different reaction to a disconnect error before the link has started. The link is simply halted, it is NOT started and a character is NOT sent. This ambiguity is resolved in the SpaceWire standard by modification of the state machine. If a disconnect error is detected before the link has started then the link resets immediately – it does not send a character first. This requires modification to the state machine to work reliably.
- IEEE 1355 section 5.7.2 states "After reset, a DS-SE link output shall maintain both signals at their reset level until started, i.e., instructed to begin operation (note that receipt of a character by the corresponding link input may be taken as such an instruction)." The state diagram in Figure 5-11 of IEEE 1355 requires the *LinkStart* command even when a character (specifically a NULL)

has been received. When a NULL is received in the *Ready* state the link moves to the *NullReceived* state. To move on to the *Run* state it must then receive the *LinkStart* command. The SpaceWire standard requires the *LinkEnable* (equivalent to *LinkStart)* command to be given before a link can move to the *Run* state.

- ❑ The state machine illustrated in Figure 5-11 of IEEE 1355 will hang up if the *ResetLinkCommand* is given to both ends of the link while they are both in the *Ready* state. In the *Ready* state no characters have been sent so the disconnect time-out has not started (disconnect time-out is started only after a bit has been received – IEEE 1355 section 5.7.2). When the *ResetLinkCommand* is given the state machine moves to the *WaitInStop* state where it waits for a disconnect error – that cannot occur! Further application of the *ResetLinkCommand* will not resolve this problem, the state machine remains firmly stuck in the *WaitInStop* state until a *PowerOnReset* is applied. The SpaceWire standard has a modified state machine in the exchange level that resolves this problem. The *WaitInStop* state has been removed completely and any Reset command causes a transition to the *ErrorReset* state.
- ❑ A similar problem to that described above can occur if one end of the link (end A) has started (in *Started* state) and the other end (end B) is in the *NullsReceived* state, but has not yet been commanded to start. If end A receives a *ResetLinkCommand* it will move to the *WaitInStop* state. In the *WaitInStop* state the outputs are halted so end A stops transmitting NULLs and this is detected as a disconnect error at end B. End B then moves through the *ErrorReset* and *ErrorWait* states ending up in the *Ready* state. Meanwhile end A has remained in the *WaitInStop* state unable to detect the disconnect error because it has not been sent a character. The modified state machine specified in the SpaceWire standard resolves this problem.
- ❑ IEEE-1355 allows an FCT to be received before a NULL received. SpaceWire requires a NULL to be received BEFORE an FCT is received.

C.5 PACKET LEVEL

- ❑ The IEEE Std 1355-1995 specification section 9.2.1 is unclear in the case of a destination being null. It is not clear whether this means a destination address of zero, which is a valid destination address, or whether it means a non-existent destination, i.e. the destination list is empty (contains zero destination identifiers). The latter case is assumed in the SpaceWire standard.
- ❑ The IEEE Std 1355-1995 specification section 9.2.1 is also ambiguous in its definition of a dest_id. It is defined as a fixed size field, its size being known to the (sub)network. It is not clear whether a network comprising several sub-networks can have different size dest_ids. E.g. the first sub-network encountered by a packet may expect a dest_id of 2-bytes and the next sub network encountered may expect a dest_id of 1-byte. In the SpaceWire standard it is assumed that a destination list can contain dest_ids of different lengths. Each routing switch will know how many bytes to strip off. The packet source must know the destination address across the network. Alternative paths to the destination available to the source can have different format destination lists. Alternative paths to a destination determined by an intelligent routing device must have the same format destination lists.

C.6 NETWORK LEVEL

- ❑ There is no Network level in IEEE-1355.

C.7 ERROR RECOVERY SCHEME

- ❑ There is no Error Recovery Scheme defined in IEEE-1355.

C.8 OTHER MINOR DIFFERENCES

- ❑ The name flow control token (FCT) is used consistently throughout the SpaceWire standard in place of FCT and FCC in IEEE-1355.

- □ The two EOP tokens of IEEE-1355 (EOP-1 and EOP-2) have been renamed EOP (End of Packet) and EEP (Error End of Packet). EOP of SpaceWire and EOP-1 of IEEE-1355 have the same function. The EEP of SpaceWire is used specifically for error recovery purposes and terminates a packet that has ended prematurely due to a link error.

Annex D (Informative) BIBLIOGRAPHY

D.1 REFERENCE DOCUMENTS[11]

RD1 IEEE Computer Society, "IEEE Standard for Heterogeneous Interconnect (HIC) (Low-Cost, Low-Latency Scalable Serial Interconnect for Parallel System Construction)", IEEE Standard 1355-1995, IEEE, June 1996[12].

RD2 IEEE Computer Society, "IEEE Standard for Low-Voltage Differential Signals (LVDS) for Scalable Coherent Interface (SCI)", IEEE Standard 1596.3-1996, IEEE, July 1996.

RD3 Parkes SM, Gillions AR et al, "Digital Signal Processing Technology Study Final Report", ESTEC contract no. 8741/90/NL/JG(SC), British Aerospace Space Systems Ltd TP9459, 450 pages, September 1992.

RD4 Parkes SM et al, "Multi-Processor Evaluation", Section in "High Performance 32-bit DSP Evaluation Study" ESTEC contract no. 9736/91/NL/JG, Deutsche Aerospace, Analog Devices, Bae Space Systems Ltd, MHS, TCS, June 1994.

RD5 Parkes SM and Price SD, "DSP-Link Study Final Report", ESTEC contract no. 8741/90/NL/JG(SC), Report No. ESA-CR(P)-3864, Matra Marconi Space TR 0104, October 1994.

RD6 PALASIM[13]

RD7 Parkes SM, "On-Board Processing: Cheaper, Faster, Better", 32nd ESLAB Symposium on Remote Sensing Methodology for Earth Observation and Planetary Exploration, European Space Technology Centre (ESTEC), Noordwijk, The Netherlands, 15-18 September 1998, European Space Agency (ESA) publication SP-423, ISBN 92-9092-692-9.

RD8 Parkes SM, "DSP (Demanding Space-based Processing!): the Path Behind and the Road Ahead", DSP'98, 6th International Workshop on Digital Signal Processing Techniques for Space Applications, European Space Technology Centre (ESTEC), Noordwijk, The Netherlands, 23-25 September 1998, Sponsored by IEEE Benelux Chapter on Communications and Vehicular Technology, European Space Agency (ESA) publication no. WPP-144, paper 4.1.

RD9 DIPSAP I

RD10 DIPSAP II

RD11 Parkes SM et al, "Review of Standard and Status", Digital Interface Circuit Evaluation Study WP1000 Technical Report, Document No. UoD-DICE-TN-1000, ESA Contract No. 12693/97/NL/FM, University of Dundee, July 1998.

RD12 IEEE, "The New IEEE Standard Dictionary of Electrical and Electronics Terms", IEEE Standard 100-1992, IEEE, 1992.

[11] During the DRAFT stages within the development of the SpaceWire Standard, most of the references below will be available on the LITERATURE and COMPONENTS-BOARDS section of http://www.estec.esa.nl/tech/spacewire .

[12] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 455 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (http://www.standards.ieee.org/).

[13] Further information on Solid State Mass Memories can be found at ftp://ftp.estec.esa.nl/pub/ws/wsd/ssmm/intronew.htm .

RD13 Konig W, “Rosetta EMC Requirements Specification”, RO-DSS-RS-1002, Issue 2, Daimler-Benz Aerospace, January 1998.

RD14 Gottschlag D and Aulton J, “ ENVISAT-1 EMC Requirements Specification”, PO-RS-DOR-PL-0006, Issue 2, Dornier GmbH, February 1994.

RD15 Parkes SM and Tuominen T, “EMC Test Results”, Digital Interface Circuit Evaluation Study WP3000 Technical Report, Document No. UoD-DICE-TN-3000, ESA Contract No. 12693/97/NL/FM, University of Dundee, April 1999.

RD16 Parkes SM, “High-Speed, Low-Power, Excellent EMC: LVDS for On-Board Data Handling”, Proceedings of the 6th International Workshop on Digital Signal Processing Techniques for Space Applications, ESTEC, Sept. 1998.

RD17 IEEE Computer Society, “IEEE Standard for a High Performance Serial Bus”, IEEE Standard 1394-1995, IEEE, August 1996.

RD18 May MD, Thompson PW & Welch PH, “Networks, Routers & Transputers: Function, Performance and Application”, IOS Press, ISBN 90 5199 129 0, 1993.

RD19 Rosello J, Parkes SM and Christen AZ, “ From IEEE-1355 to SpaceWire”, Proceedings DASIA’99, Lisbon, Portugal, May 1999, To be published.

RD20 ESA-PSS-07-101, Packet Utilisation Standard

RD21 ECSS-Procedure-13, Rules for the drafting and presentation of ECSS Standards[14]

RD22 ECSS-P-00, Standardisation policy

RD23 ECSS-P-001, Glossary of terms

RD24 ECSS-E-00, Space Engineering – Policy and principles

RD25 ECSS-E-10, Space Engineering – System Engineering

RD26 ECSS-E-10-02, Space Engineering – Verification

RD27 ECSS-E-10-03 Draft, Space Engineering – Testing

RD28 ECSS-E-20, Space Engineering – Electrical and Electronic

RD29 ECSS-E-50, Space Engineering – Communications

RD30 ECSS-E-70, Space Engineering – Ground Systems and operations

RD31 ECSS-M-00, Space project management – Policy and principle

RD32 ECSS-Q-00, Space product assurance – Policy and principle

RD33 ECSS-Q-60, Space product assurance – EEE components

RD34 ECSS-Q-70, Space product assurance – Materials, Mechanical Parts & Processes

RD35 ECSS-Q-80, Space product assurance – Software Product Assurance

RD36 ISO/IEC Guide 25, General requirements for the competence of testing and calibration laboratories[15]

RD37 ISO/IEC Directives, Part 3, Rules for the structure and drafting of International Standards, 3rd edition, 1997

[14] ECSS publications are available from the ESA/ECSS secretariat (http://www.estec.esa.nl/ecss/)

[15] ISO/IEC publications are available from the International Organization for Standardization, 1, rue de Varembé, Case postale 56, CH-1211 Genève 20 (http://www.iso.ch) and from the International Electro-technical Commission , 1, rue de Varembé, Case postale 56, CH-1211 Genève 20 (http://www.iec.ch) .

This page was mistakenly left blank!