ECSS-E-ST-50-12C Rev.1
15 May 2019

Space engineering

SpaceWire - Links, nodes, routers and networks

ECSS Secretariat
ESA-ESTEC
Requirements & Standards Division
Noordwijk, The Netherlands

Foreword

This Standard is one of the series of ECSS Standards intended to be applied together for the management, engineering, product assurance and sustainability in space projects and applications. ECSS is a cooperative effort of the European Space Agency, national space agencies and European industry associations for the purpose of developing and maintaining common standards. Requirements in this Standard are defined in terms of what shall be accomplished, rather than in terms of how to organize and perform the necessary work. This allows existing organizational structures and methods to be applied where they are effective, and for the structures and methods to evolve as necessary without rewriting the standards.

This Standard has been prepared by the ECSS-E-ST-50-12C-Rev1 Working Group, reviewed by the ECSS Executive Secretariat and approved by the ECSS Technical Authority.

Disclaimer

Published by: ESA Requirements and Standards Division
ESTEC, P.O. Box 299,
2200 AG Noordwijk
The Netherlands

Change log

ECSS-E-50-12A	First issue.
ECSS-E-50-12B	Never issued.
ECSS-E-ST-50-12C 31 July 2008	Second issue. Editorial changes to conform with the new ECSS template and reorganization of information in clause 10.
ECSS-E-ST-50-12C Rev.1 15 May 2019	Third issue. Complete rewriting of the standard that aims to be compliant to the previous issues, while making essential improvements and required additions. • General improvement of terms and clauses in line with ECSS rules. • Removal of errors and ambiguities throughout document. • Clarifications of terms and inclusion of more terms in the list of terms. • Improvement to the protocol layering and inclusion of a protocol stack diagram. • Improvement to SpaceWire port architecture diagram. • Improvements to physical layer to allow other cable and connector solutions. • Addition of recovery state diagram making the recovery operation clear. • Detailing of multicast operation. • Addition of distributed interrupts. • Addition of UML diagrams and detailed descriptions of node, router, network, broadcast code and device. • Addition of service interface specifications for all layers. Annex A gives a brief overview of the technical changes introduced in this revision of the standard.

Table of contents

Figures

Tables

1
Scope

SpaceWire technology has grown from the needs of spacecraft on-board data handling applications. This Standard provides a formal basis for the exploitation of SpaceWire in a wide range of future on-board processing systems.

One of the principal aims of SpaceWire is the support of equipment compatibility and reuse at both the component and subsystem 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.

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

This standard may be tailored for the specific characteristics and constraints of a space project in conformance with ECSS-S-ST-00.

2
Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this ECSS Standard. For dated references, subsequent amendments to, or revision of any of these publications do not apply. However, parties to agreements based on this ECSS Standard are encouraged to investigate the possibility of applying the more recent editions of the normative documents indicated below. For undated references, the latest edition of the publication referred to applies.

ECSS-S-ST-00-01	ECSS system - Glossary of terms
ECSS-Q-ST-70-08	Space product assurance - Manual soldering of high-reliability electrical connections
ECSS-Q-ST-70-26	Space product assurance - Crimping of high-reliability electrical connections
ESCC 3401	Connectors, electrical, non-filtered circular and rectangular, ESCC Generic Specification no. 3401, Issue 5, March 2018
ESCC 3401/029:2017	Connectors, Electrical, Rectangular, Microminiature, based on type MDMA, ECSS Detail Specification no. 3401/029, Issue 15, November 2017.
ESCC 3401/077:2016	Connectors, Electrical, Rectangular, Microminiature, Removable Crimp Contacts, based on type MDMA, ECSS Detail Specification no. 3401/077, Issue 7, April 2016.
ESCC 3902/003:2014	Cable, “SpaceWire”, Round, Quad using Symmetric Cables, Flexible, –200 to +180 °C, Detail Specification no. 3902/003, Issue 4, November 2014.
ESCC 3902/004:2014	Cable, Low Mass, “SpaceWire”, Round, Quad using Symmetric Cables, Flexible, –100 to +150 °C, Detail Specification no. 3902/004, Issue 1, October 2014.
MIL-DTL-17J:2014	Military Specification: Cables, Radio, Frequency, Flexible and Semirigid, General Specification for, 10th February 2014.
TIA-644-A:2012	TIA-644-A, Electrical Characteristics of Low Voltage Differential Signalling (LVDS) Interface Circuits, Revision A, Reaffirmed 12/07/12, Telecommunications Industry Association, 2012.

3
Terms, definitions and abbreviated terms

3.1 Terms from other standards

For the purpose of this Standard, the terms and definitions from ECSS-S-ST-00-01 apply.

3.2 Terms specific to the present standard

The UML diagrams of Figure 5-22, Figure 5-26, Figure 5-27 and Figure 5-28 in clause 5.6 illustrate the relationships between various terms used within this standard.

3.2.1 allocated output port

output port that a **packet** is routed through

3.2.2 after 6,4 µs

delay of 6,4 µs (nominal) measured from when a state is entered

3.2.3 after 12,8 µs

delay of 12,8 µs (nominal) measured from when a state is entered

3.2.4 AutoStart

management parameter set by hardware or software which when asserted allows an enabled **SpaceWire port** to start the SpaceWire **link** only when a **Null** is received

3.2.5 bit error rate

ratio of the number of bits received in error to the total number of bits sent across a link

3.2.6 broadcast code

time-code or **distributed interrupt code**

3.2.7 broadcast code identifier

two-bit code that identifies the type of **broadcast code**: 0b00 identifies a **time-code** and 0b10 identifies a **distributed interrupt code**

3.2.8 byte

eight bits

3.2.9 cargo

some information for transfer from a **source** to a **destination** which is encapsulated in a **packet**

3.2.10 character

control character or **data character**

3.2.11 coding

act of translating a set of bits into another set of bits which are more appropriate for transmitting across a medium

3.2.12 configuration port

port in a **routing switch** or **node** that gives access to a **configuration node**

3.2.13 configuration node

type of **node** whose purpose is to configure the **routing switch** or **node** that it is part of

3.2.14 control character

ESC, FCT, EOP or EEP **character** that is used to pass control information across a **link**

3.2.15 control code

sequence of an ESC followed by an FCT forming a **Null** which is used to keep a **link** active, or a sequence of an **ESC character** followed by a **data character** forming a **broadcast code** which is used to broadcast **time-codes** and **distributed interrupt codes** over a **SpaceWire network**

3.2.16 control symbol

control character encoded in 4-bits for transfer across a **link**

3.2.17 data character

character that is used to pass 8 bits of data across a **link**

3.2.18 data link layer

protocol layer which is responsible for the initialisation of a SpaceWire **link**, for transferring **packets** and **broadcast codes** over the **link** and for recovery from errors on the **link**

3.2.19 data rate

rate at which the application data is transferred across a **link**

3.2.20 data signalling rate

rate at which the bits constituting **control** and **data symbols** are transferred across a **link**

3.2.21 data-strobe

sequence of data bits and bit clock encoded into two signals, one containing the data bit sequence (data) and the other changing state whenever the data bit sequence does not (strobe)

3.2.22 data symbol

data character encoded in 10 bits for transfer across a **link**

3.2.23 decoding

act of translating an encoded set of bits back into the original set of bits prior to encoding

3.2.24 de-serialisation

transformation of a serial bit stream into a sequence of **control** or **data symbols**

3.2.25 destination

end-point that a **packet** is being sent to

3.2.26 destination address

route, described by a **path address**, to be taken by a **packet** in moving from **source** to **destination** or an identifier, in the form of a **logical address**, specifying the **destination**

3.2.27 destination node

node that is the **destination** of one or more **SpaceWire packets**

3.2.28 disconnect

indication from a **receiver** that there has been no edge on the data or strobe signals for the past 850 ns (nominal) indicating that the **link** is disconnected

3.2.29 distributed interrupt

interrupt that is distributed to all or many **nodes** on the **network**

3.2.30 distributed interrupt code

broadcast code used to distribute interrupts over a **SpaceWire network** which is either an **interrupt code** or an **interrupt acknowledgement code**

3.2.31 driver

electronic circuit that transmits signals across a particular transmission medium

3.2.32 driver ground

ground at the 0V pin or pins of the **driver** differential outputs

NOTE If the driver has one 0V for signals such as TTL or CMOS, and a separate 0V for the differential outputs, the driver ground is the 0V for the differential outputs. The driver ground is a ground plane or planes which provide

controlled impedance for the driver's differential pairs.

3.2.33 encoding layer

protocol layer which is responsible for the encoding of **characters** into **symbols**, **symbol** serialisation, data-strobe encoding of the serial bit stream, data-strobe decoding, **symbol** de-serialisation and decoding of **symbols** into **characters**

3.2.34 end of packet marker

control character which indicates the end of a **packet**

3.2.35 end-point

interface between the **network** and a **host system** providing a single **port** into the **network**

3.2.36 error recovery scheme

method for handling errors detected within a SpaceWire **link**

3.2.37 ESC

control character which is followed by another **control character** or **data character** to form a **control code**

3.2.38 ESC error

invalid **ESC** sequence, formed from an **ESC** followed immediately by an EOP, EEP, or ESC

3.2.39 FIFO port

port that has a FIFO interface rather than a **SpaceWire interface**

3.2.40 fall time

time during which a falling signal voltage is within the range of 80 % to 20 % of the difference between the two steady state values

3.2.41 first Null

initial **Null** received without a parity error when the link state machine is not in the ErrorReset state

3.2.42 flow control token

control character used to manage the flow of data across a **link**, and being exchanged for eight **N-Chars**

3.2.43 gotBC

sometime after the first **Null** was received, a **broadcast code** has been received without a parity error

3.2.44 gotFCT

sometime after the first **Null** was received, an FCT has been received without a parity error

3.2.45 gotNchar

sometime after the first **Null** was received, an **N-Char** has been received without a parity error

3.2.46 gotNull

first **Null**, after the **receiver** has been enabled, has been received without a parity error

3.2.47 group adaptive routing

assignment of a set of several **ports** to a **logical** or **path address** so that when a **packet** arrives with that address it is switched to one of the set of several **ports** that is currently available to accept the **packet** or that becomes available first

3.2.48 host interface

interface to a **host system**

3.2.49 host system

system that is connected to a **SpaceWire network** via an **end-point** and which uses the services of that **SpaceWire network**

3.2.50 input port

receive side of a **port**

3.2.51 interrupt acknowledgement code

code used to confirm that a **distributed interrupt** has reached the appropriate **interrupt handler**

3.2.52 interrupt code

code used to distribute an interrupt over a **SpaceWire network**

3.2.53 interrupt destination

node that receives and handles a **distributed interrupt**

3.2.54 interrupt handler

node that is responsible for handling an **interrupt code** with a specific value of **interrupt identifier**

3.2.55 interrupt identifier

5-bit value representing one of 32 possible interrupts, which is held in the value field of an **interrupt code** and which identifies the particular interrupt being carried by the **interrupt code** or being acknowledged by an **interrupt acknowledgement code**

3.2.56 interrupt relay register

register in a **routing switch** that holds the current state of the **distributed interrupt** where the i^{th} bit of the register relates to the **interrupt identifier** of value i, and is used to prevent repeated circular propagation of **distributed interrupt codes**

3.2.57 interrupt source

node that generates a **distributed interrupt**

3.2.58 jitter

random errors in the timing of a signal

3.2.59 L-Char

link character

3.2.60 leading data character

very first **data character** sent over a link after initialisation or the first **data character** following the EOP or EEP that terminated the previous packet

3.2.61 line driver

electronic circuit that drives signals across a particular transmission medium

3.2.62 line receiver

electronic circuit that receives signals sent across a particular transmission medium

3.2.63 link

bi-directional connection between two **ports** used to transfer **packets** and **broadcast codes** between the two **ports**

3.2.64 link character

control character or **control code** which appears on the **link** only and is not passed between the data link and network layers

NOTE The **FCT** and **Null** are the only link characters

3.2.65 LinkDisabled

management parameter which when asserted prevents a **SpaceWire port** from operating

3.2.66 link error

disconnect error, parity error, **ESC** error or credit error

3.2.67 link receiver

receiver at an end of a SpaceWire **link**

3.2.68 LinkStart

management parameter set by hardware or software which when asserted causes an enabled **SpaceWire port** to attempt to start the SpaceWire **link** by sending **Nulls**

3.2.69 link transmitter

transmitter at an end of a SpaceWire **link**

3.2.70 logical address

data character which identifies the **destination** for the **packet**

3.2.71 low voltage differential signalling

particular form of differential signalling using low voltage signals

3.2.72 management parameter

configuration parameter, control variable or status variable of a **SpaceWire node** or **routing switch** used to manage its operation

3.2.73 multicast

sending of the same **packet** through two or more output **ports** of a **routing switch** concurrently

3.2.74 multicast set

set of **output ports** assigned to a **logical address** through which a **packet** with that **logical address** is forwarded concurrently

3.2.75 N-Char

normal-character

3.2.76 network level

protocol level responsible for transferring **packets** across a **SpaceWire network** from **source node** to **destination node** via **links** and **routers**

3.2.77 node

source or **destination** of SpaceWire **packets** and **broadcast codes** comprising one or more **end-points** each providing an interface between a **port** and a **host system**

3.2.78 normal-character

data character, EOP or EEP

3.2.79 Null

control code made up of an **ESC** followed by an **FCT** which is sent to keep the data **link** active when there are no **data** or **control characters** to send, thus preventing a disconnect

3.2.80 output port

transmit side of a **port**

3.2.81 packet

sequence of **normal-characters** comprising a **destination address**, **cargo** and an **end of packet marker**

3.2.82 path address

series of one or more **data characters** at the start of a **packet** which define the route to be taken across a **SpaceWire network** from **source** to **destination**

3.2.83 physical layer

protocol layer which specifies the cables, connectors, cable assemblies, **line drivers** and **line receivers**

3.2.84 port

SpaceWire interface or FIFO interface comprising an **input port** and an **output port**

3.2.85 port reset

reset signal that resets a single **SpaceWire port**

3.2.86 receive error

error detected when receiving a **symbol**

NOTE This is a parity error or invalid code error

3.2.87 receive FIFO

FIFO memory which stores received **N-Chars** until they can be read by the application via the **SpaceWire port** interface

3.2.88 receiver

circuit that receives signals from the **line receiver** and decodes those signals into a stream of **characters**

3.2.89 receiver ground

ground at the 0V pin or pins of the **receiver** differential inputs

NOTE If the driver has one 0V for signals such as TTL or CMOS, and a separate 0V for the differential inputs, the receiver ground is the 0V for the differential inputs. The receiver ground is a plane or planes which provide controlled impedance for the receiver's differential pairs.

3.2.90 reset

power on reset, other hardware reset or software commanded reset

3.2.91 rise time

time during which a rising signal voltage is within the range of 20 % to 80 % of the difference between the two steady state values

3.2.92 router

routing switch

NOTE The term “router” is used because of the heritage of SpaceWire which is based on IEEE1355-1995 which is in turn based on earlier work on Transputer technology. This work predated the Internet and the use of the term “router” to mean a device that determines the

route to a destination as opposed to a "switch" that switches a packet to an output port based on an address. In SpaceWire the term "router" is used to mean a routing switch. This is widely understood terminology in the SpaceWire community.

3.2.93 routing switch

switch matrix connected to one or more **SpaceWire ports**, a **configuration port** and a local **broadcast code** register, which optionally broadcasts **broadcast codes** and which switches **packets** from one **port** to one or more other **ports** where the **destination address** of each **packet** is used by the **routing switch** to determine which **port** the **packet** is forwarded through

3.2.94 routing table

table in a **routing switch** which is indexed by the leading **data character** of the **packet** to look-up the **output port** for forwarding a **packet** through

3.2.95 serialisation

transformation of a sequence of **control** or **data symbols** into a serial bit stream

3.2.96 signal

measurable quantity that varies with time and propagates along a transmission medium to transfer information across that medium

3.2.97 skew

difference in time between the expected position of the rising or falling edge of a signal and the actual position of that signal

3.2.98 source

originator of a **packet**, signal or other form of information

3.2.99 source node

node that is the **source** of one or more SpaceWire **packets**

3.2.100 SpaceWire interface

interface comprising a transmitter for sending information across a SpaceWire **link**, and a receiver for receiving information from that SpaceWire **link**

3.2.101 SpaceWire network

two or more **nodes** connected together via one or more **links** and zero or more **routing switches**

NOTE A point-to-point link between two nodes is therefore regarded as a network.

3.2.102 SpaceWire node

node

3.2.103 SpaceWire port

port which has a **SpaceWire interface**

3.2.104 switch matrix

non-blocking, worm-hole **routing switch** that switches a **packet** arriving at an **input port** of a **routing switch** to the appropriate **output port**, or to several **output ports** in case of multicasting

3.2.105 symbol

encoded **data character**, encoded **control character** or encoded **control code**

3.2.106 time-code

code used to distribute synchronisation information over a **SpaceWire network**

3.2.107 time-code master

host application on a **node** that is responsible for periodically sending out **time-codes** which are broadcast across the **SpaceWire network**

3.2.108 time-code master node

node on which the **time-code master** resides and which when requested by a **time-code master** sends **time-codes** out of one or more **end-points**

3.2.109 time-code register

register in a **node** or **routing switch** that holds the value of the last **time-code** received

3.2.110 time-code value

six bits of information contained in a **time-code**

3.2.111 transmission medium

medium over which data is transferred e.g. screened twisted-pair wires

3.2.112 transmit FIFO

FIFO memory which stores **N-Chars** until they can be sent across the **link**

3.2.113 transmitter

circuit that encodes the **characters** being sent across a **link**, serialises them, data-strobe encodes them and passes the resulting data and strobe signals to **line drivers**

3.2.114 unit

entity, like an instrument, processor or mass memory, which contains zero or more **nodes** and zero or more **routing switches**, and which contains at least one **node** or one **routing switch**

3.3 Abbreviated terms

The following abbreviations are defined and used within this standard:

Abbreviation	Meaning
AWG	American Wire Gauge
BC	broadcast code
BER	bit error rate
DC	direct current
DS	Data-Strobe
ECSS	European Cooperation for Space Standardization
EEP	error end of packet
EOP	end of packet
ESA	European Space Agency
ESC	escape character
ESCC	European Space Components Coordination
FCT	flow control token
FIFO	first in first out
ID	identifier
IID	Interrupt identifier
LS	least-significant
LSB	least significant bit
LVDS	low voltage differential signalling
LVTTL	low voltage transistor-transistor logic
Mbps	Megabits per second
MS	most-significant
MSB	most-significant bit
PCB	printed circuit board
PSELFEXT	Power Sum Equal Level Far-End Cross-Talk
PSNEXT	Power Sum Near End Cross-Talk
RX	receive
SpW	SpaceWire
TDR	time domain reflectometer
TX	transmit
UML	Unified Modelling Language

3.4 Conventions

3.4.1 Numbers

In this document hexadecimal numbers are written with the prefix 0x, for example 0x34 and 0xDF15.

Binary numbers are written with the prefix 0b, for example 0b01001100 and 0b01.

Decimal numbers have no prefix.

3.4.2 Differential signals

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 are designated Dout+, Dout- and Sout+, Sout- for Data and Strobe, respectively. Similarly, the signals at the input end of a SpaceWire link are Din+, Din- and Sin+, Sin-.

3.4.3 Order of sending bits in symbols

The first bit of a data symbol or control symbol to be sent is the parity bit.

The data bits of a data symbol are sent least significant bit first.

An arrow is added to relevant diagrams to illustrate the direction that the data moves, and which bit is sent first as shown in Figure 3-1

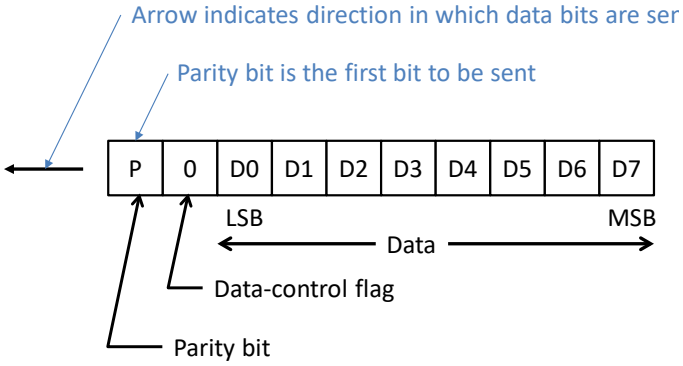


Figure 3-1: Convention for first bit to be sent

3.4.4 Graphical representation of packets

Packets and their component fields are represented graphically as shown in Figure 3-1. The arrow indicates the direction that the packet is travelling, so the destination field is sent first.

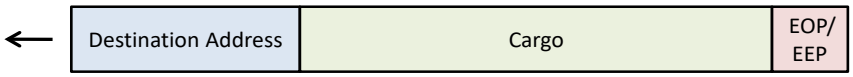


Figure 3-2: Graphical packet notation

3.4.5 State diagram notation

State diagrams are used to represent the behaviour of several parts of the SpaceWire protocol. All state diagrams in this Standard use the style shown in Figure 3-3. States are represented by ellipses with the state name written inside the ellipse in bold. Actions to take 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.

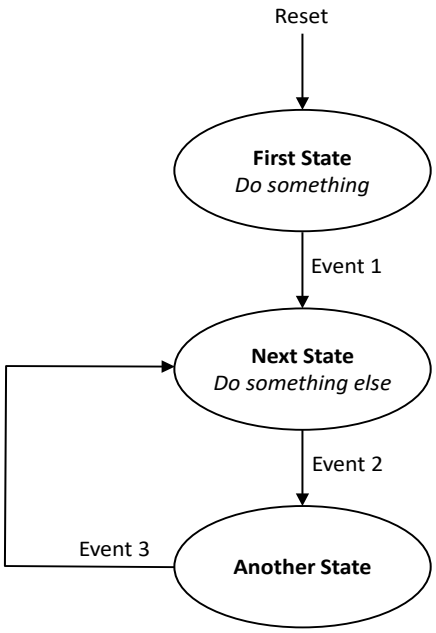


Figure 3-3: State diagram style

3.4.6 UML diagram notation

UML diagrams in this document use the notation illustrated in Figure 3-4.

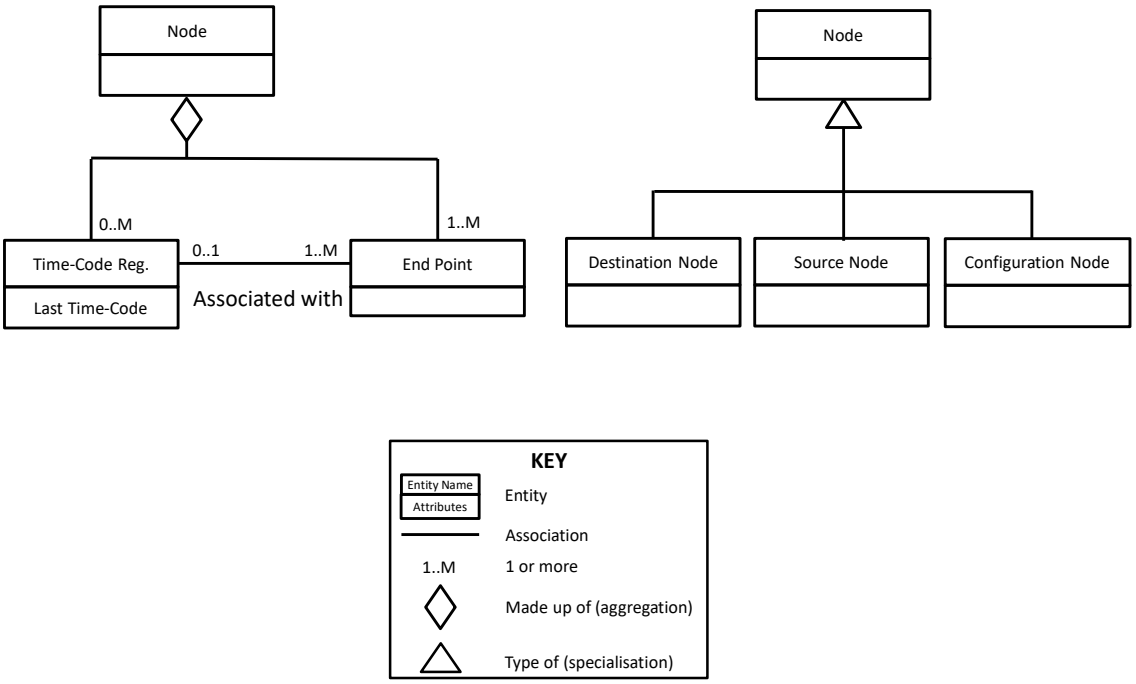


Figure 3-4: UML notation

Each entity (or object) is shown as a rectangle divided in two by a horizontal line. The name of the entity is at the top above the line and a list of key attributes is below the line. Only those attributes that are particularly relevant to understanding a diagram are shown, for example "Last Time-Code" in the Time-Code Register.

A diamond is used to represent aggregation and is translated as "is made up of", hence in Figure 3-4 a node is made up of one or more (1..M) end-points and zero or more (0..M) time-code registers. The number at the end of the line shows the cardinality of the relationship and is either a number (e.g. 1) or a range (e.g. 1..M) with M representing more or many.

A triangle represents specialisation in one direction (reading in the direction the triangle is pointing) and generalisation in the other. So, in Figure 3-4, a destination node is a special type of node. Reading in the other direction, a node is a generalisation of destination node, source node and configuration node.

Other associations between entities are shown as lines connecting the two objects. A name capturing the intent of the association is written alongside the line. If the association is general, no name needs to be given to it. Hence, in Figure 3-4, each end-point is associated with zero or more time-code registers and each time-code register is associated with 1 or more end-points.

3.5 Nomenclature

The following nomenclature applies throughout this document:

a. The word “shall” is used in this Standard to express requirements. All the requirements are expressed with the word “shall”.

b. The word “should” is used in this Standard to express recommendations. All the recommendations are expressed with the word “should”.

NOTE It is expected that, during tailoring, recommendations in this document are either converted into requirements or tailored out.

c. The words “may” and “need not” are used in this Standard to express positive and negative permissions, respectively. All the positive permissions are expressed with the word “may”. All the negative permissions are expressed with the words “need not”.

d. The word “can” is used in this Standard to express capabilities or possibilities, and therefore, if not accompanied by one of the previous words, it implies descriptive text.

NOTE In ECSS “may” and “can” have completely different meanings: “may” is normative (permission), and “can” is descriptive.

e. The present and past tenses are used in this Standard to express statements of fact, and therefore they imply descriptive text.

4
Overview of SpaceWire

4.1 Introduction

SpaceWire is a data-handling network for use on-board spacecraft, which connects together instruments, mass-memory, processors, downlink telemetry, and other on-board sub-systems. SpaceWire provides 2 Mbps to 200 Mbps, bi-directional, full-duplex data-links, which connect together SpaceWire-enabled equipment. Data rates of over 400 Mbps are possible when matched impedance connectors are used. It delivers high-speed, low-power, simplicity, relatively low implementation cost, and its architectural flexibility makes it ideal for many space missions. Data-handling networks can be built to suit particular applications using point-to-point data-links and routing switches [ESA Bulletin Vol. 145].

Since the SpaceWire standard was published by the European Cooperation for Space Standardization in January 2003, it has been adopted by ESA, NASA, JAXA and Roscosmos for many missions and is being widely used on scientific, Earth observation, commercial and other spacecraft. High-profile missions using SpaceWire include: Gaia, ExoMars, Bepi-Colombo, James Webb Space Telescope, GOES-R, Lunar Reconnaissance Orbiter and Astro-H.

SpaceWire has been developed in an extremely successful collaboration between ESA, academia and space industry, involving spacecraft engineers from across the world. This revision of the SpaceWire standard aims to correct errors, remove ambiguities and make clarifications to the previous version. It adjusts the SpaceWire protocol stack to make the layer interfaces clear and provides abstract service interface descriptions for each layer. Distributed interrupts are added to SpaceWire and time-codes are generalised into broadcast codes which comprise time-codes and distributed interrupt codes.

4.2 SpaceWire Spacecraft Data-Handling Network

4.2.1 The Rationale for SpaceWire

Before SpaceWire became a standard, many spacecraft primes and equipment manufacturers had to use ad hoc or their own proprietary interfaces for inter-unit communications, e.g. connecting high data-rate instruments to mass-memory units. This resulted in several different types of communication links being used on a spacecraft, increasing the cost and extending the time required

for spacecraft integration and test. There was a clear need for a standard on-board communication link to simplifies spacecraft development.

SpaceWire aims to:

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

Use of the SpaceWire standard ensures that equipment is compatible at both the component and sub-system levels. Instruments, processing units, mass-memory devices and down-link telemetry systems using SpaceWire interfaces that were developed for one mission can be readily used on another mission. This reduces the cost of development (**Cheaper**) and reduces development timescales (**Faster**) while improving reliability (**Better**) and maximising the amount of science and data return that can be achieved within a given budget (**More**).

4.2.2 Example SpaceWire Application

SpaceWire is able to support many different payload processing architectures using point-to-point links and SpaceWire routing switches. The data-handling architecture can be constructed to suit the requirements of a specific mission, rather than having to force the application onto a constrained bus or network with restricted topology.

An example SpaceWire architecture is shown in Figure 4-1. It uses two SpaceWire routing switches to provide the interconnectivity between instruments, memory and processing modules.

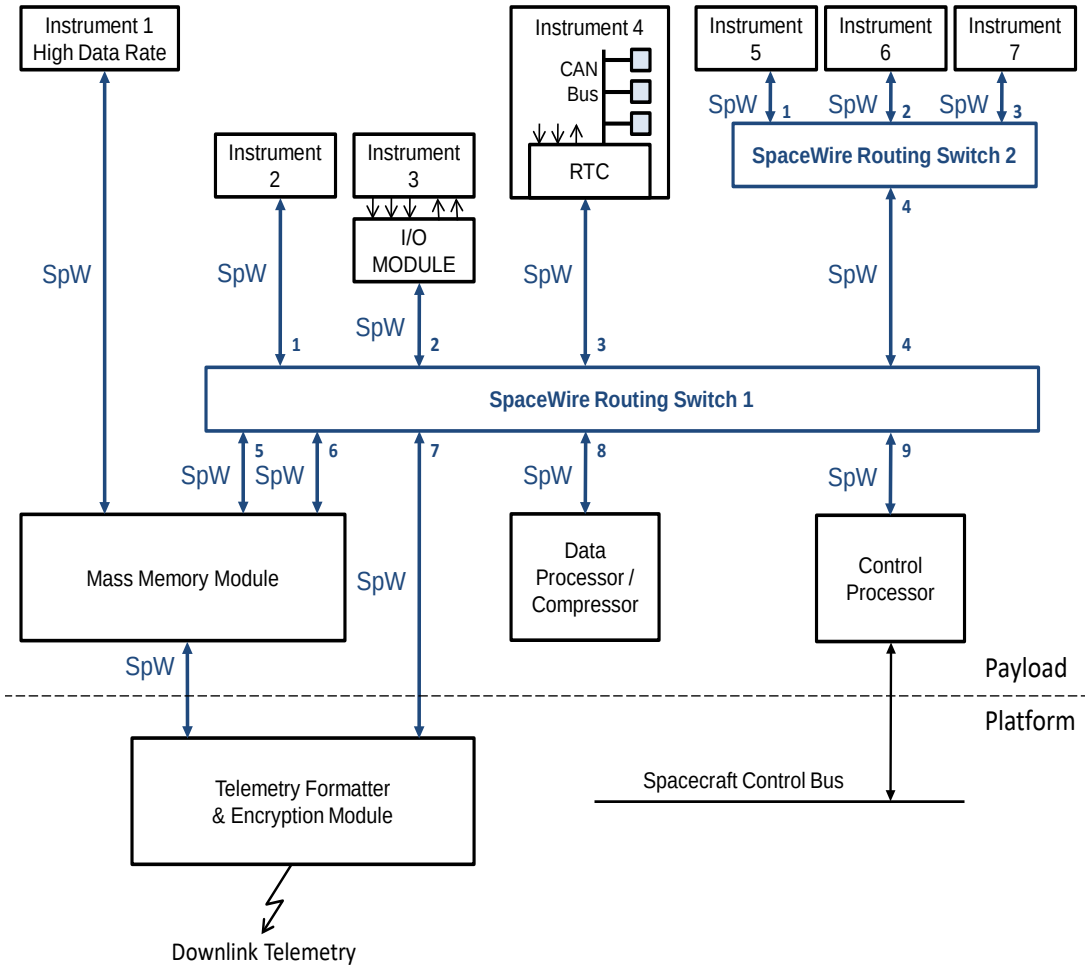


Figure 4-1: Example SpaceWire Architecture without redundancy

Instrument 1 in the top left-hand corner is a high data-rate instrument (e.g. producing data at a rate of 150 Mbps). A SpaceWire point-to-point link is used to stream data from this instrument directly into the Mass Memory Module. If a single SpaceWire link is insufficient to handle the data-rate from this instrument, two or more links can be used in parallel.

Instrument 2 is of lower average data-rate than instrument 1. Its data is passed through SpaceWire Routing Switch 1 to the Mass Memory Module.

Instrument 3 does not have a SpaceWire interface so an input/output (I/O) module is used to connect the instrument to the SpaceWire router. Its data is then be sent over the SpaceWire network to the Mass Memory Module.

Instrument 4 is a complex instrument containing a number of sub-modules which are interconnected using the CAN bus. A Remote Terminal Controller (RTC) is used to bridge between the CAN bus and SpaceWire. Other signals from the instrument are also connected to the RTC, which contains a processor for performing the bridging and local instrument control functions.

Instruments 5, 6 and 7 are located in a remote part of the spacecraft. To avoid having three SpaceWire cables running to this remote location a second routing switch (SpaceWire Routing Switch 2) is used to concentrate the information

from these three instruments and send it over a single SpaceWire link to Routing Switch 1 and then on to the Mass Memory Module.

This Mass Memory Module can receive data from any of the instruments either directly, as is the case for Instrument 1, or indirectly via Routing Switch 1. Data stored in the Mass Memory Module can be sent to the Telemetry Formatter/Encryption Module for sending to Earth, or it is first be sent to a Data Processing or Data Compression Unit. This unit returns the processed/compressed data to the Mass Memory Module or send it straight to the Telemetry Module via Routing Switch 1.

The Control Processor is responsible for controlling all the Instruments, the Mass Memory Module and the Telemetry unit. The SpaceWire Network provides access to all these modules so that the Control Processor can configure, control and read housekeeping and status information from them. The Control Processor is also attached to the spacecraft control bus over which it can receive telecommands and forward housekeeping information.

With several instruments and the Data Processor/Compressor sending data to the Mass Memory Module via Routing Switch 1, a single link from that Routing Switch to the Mass Memory Module can be insufficient to handle all the data, so a second link has been added to provide more bandwidth. In a SpaceWire network links can be added to provide additional bandwidth or to add fault tolerance to the system. In Figure 4-1 no redundancy has been included for clarity. In a spaceflight application, an additional pair of routing switches is included with duplicate links to the modules to provide redundancy. It is straightforward to support traditional cross-strapped, redundant modules using SpaceWire.

4.2.3 How SpaceWire Works

4.2.3.1 Overview

Having looked at the way in which SpaceWire can be used to provide a spacecraft data-handling system, this section examines SpaceWire in more detail to provide an understanding of how SpaceWire works. The normative clauses are presented in clause 5.

4.2.3.2 SpaceWire Links

SpaceWire links are point-to-point data links that connect a SpaceWire node (e.g. instrument, processor, mass-memory unit) to another node or to a router. Information can be transferred in both directions of the link at the same time. Each full-duplex, bi-directional, serial data link can operate at data-rates between 2 Mbps and 200 Mbps. When matched impedance connectors are used, SpaceWire is able to run from 2 Mbps to rates above 400 Mbps. It sends information as a serial bit stream using two signals in each direction (data and strobe). These signals are normally driven across a link using Low Voltage Differential Signalling (LVDS), which requires two wires for each signal, resulting in a SpaceWire cable containing four screened twisted pairs.

Bit synchronisation in SpaceWire is achieved by sending a clock signal along with the serial data. To reduce the maximum clock to data skew requirements

the clock signal is encoded into a strobe signal in such a way that XORing the data and strobe signal recovers the clock signal. Data is transferred as a stream of elementary data and control characters. Character synchronisation is only performed once, when the link is started. If character synchronisation is ever lost, leading to parity errors, this is detected and the link restarted to recover initial character synchronisation.

SpaceWire provides a simple mechanism for starting a link, keeping the link running, sending data over the link, ensuring that data is not sent if the receiver at the other end of the link is not ready for it, and for recovering from any errors on the link. All this is handled by state-machines in the SpaceWire interface and is transparent to the user application.

4.2.3.3 SpaceWire Packets

Information is transferred across a SpaceWire link in distinct packets. Packets can be sent in both directions of the link, provided that there is room in the receiver for more data.

A packet is formatted as illustrated in Figure 4-2.

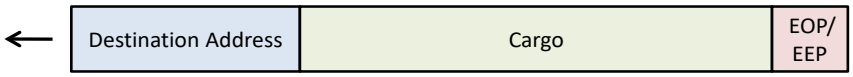


Figure 4-2: SpaceWire Packet Format

The "Destination Address" at the beginning of the packet contains either the identity of the destination node or the path the packet has to take through a SpaceWire network to reach to the destination node. In the case of a point-to-point link connection directly between two nodes (no routing switches in between), the destination address is not necessary.

The "Cargo" is the data to be transferred from source to destination. An arbitrary number of data bytes can be transferred in the cargo of a SpaceWire packet.

The End of Packet marker (EOP or EEP) is the last character in the packet. The data character following an End of Packet is the start of the next packet. There is no limit on the size of a SpaceWire packet.

As can be seen the SpaceWire packet format is very simple. It is, however, also very powerful, allowing SpaceWire to be used to carry a wide range of application protocols, with minimal overhead.

4.2.3.4 SpaceWire Networks

4.2.3.4.1 General

SpaceWire networks are constructed using SpaceWire point-to-point links and routing switches.

4.2.3.4.2 SpaceWire Routing Switches

A SpaceWire routing switch connects together many nodes using SpaceWire links, providing a means of routing packets from one node to any of the other nodes or routing switches attached to the router. A node is simply the source or destination of a SpaceWire packet. A SpaceWire routing switch comprises a number of SpaceWire ports and a switch matrix. The switch matrix enables packets arriving at one port to be transferred to and sent out through another port on the router.

Each port can be considered as comprising an input port (the port receiver) and an output port (the port transmitter). A SpaceWire routing switch transfers packets from the input port of the routing switch where the packet arrives, to a particular output port determined by the packet destination address. A routing switch uses the leading data character of a packet (one of the destination address characters) to determine the output port of the routing switch to which the packet is routed. If there are two input ports waiting to use a particular output port when the previous packet has finished being sent, an arbitration mechanism decides which input port is served next. The packet from the other input port is queued until the output port becomes available again. This so called "wormhole" switching requires a minimum amount of buffer memory, which enables a SpaceWire routing switch to be implemented even in smaller, space-qualified chip technologies.

4.2.3.4.3 Packet Addressing

The destination address at the front of a SpaceWire packet is used to route the packet through a network from the source node to the destination. There are two forms of addressing used in SpaceWire networks: path addressing and logical addressing.

Path addressing can best be understood using the simple analogy of providing directions to someone driving a car. To reach the destination it is suggested that the driver takes exit 2 at the first road-junction, exit 1 at the next road-junction, and finally exit 3 on the third road-junction. The driver then reaches the required destination (see Figure 4-3). There is a command to follow at each road-junction (take a particular exit). Together these commands describe the path from the initial position to the required destination. There is a list of commands to follow; one for each road-junction. Once a command has been followed it is dropped from the list and the next command is followed at the next road-junction.

In a SpaceWire network the road-junctions are routing switches and the roads connecting the road-junctions are the links connecting the routing switches (see Figure 4-3). The list of commands is attached to the front of the SpaceWire packet forming the destination address. The first command is followed when the first routing switch is encountered. This command is simply a data character that specifies which port of the routing switch the packet is forwarded

through. A routing switch can have a maximum of 31 external ports (port numbers 1 to 31) and one internal configuration port (port number 0). The leading data character at the front of the packet is used to specify the port that the packet is forwarded through: for instance, if the leading data character is 3, the packet is forwarded to port 3 of the router. Once the leading data character has been used to forward a packet, it is removed as it is no longer needed. This reveals the next data character in the path address for routing the packet at the next router. Figure 4-3, shows how the path address at the start of the packet is modified as the packet goes through the network. Since a routing switch can have a maximum of 31 ports along with an internal configuration port, each data character forming a path address is in the range 0 to 31.

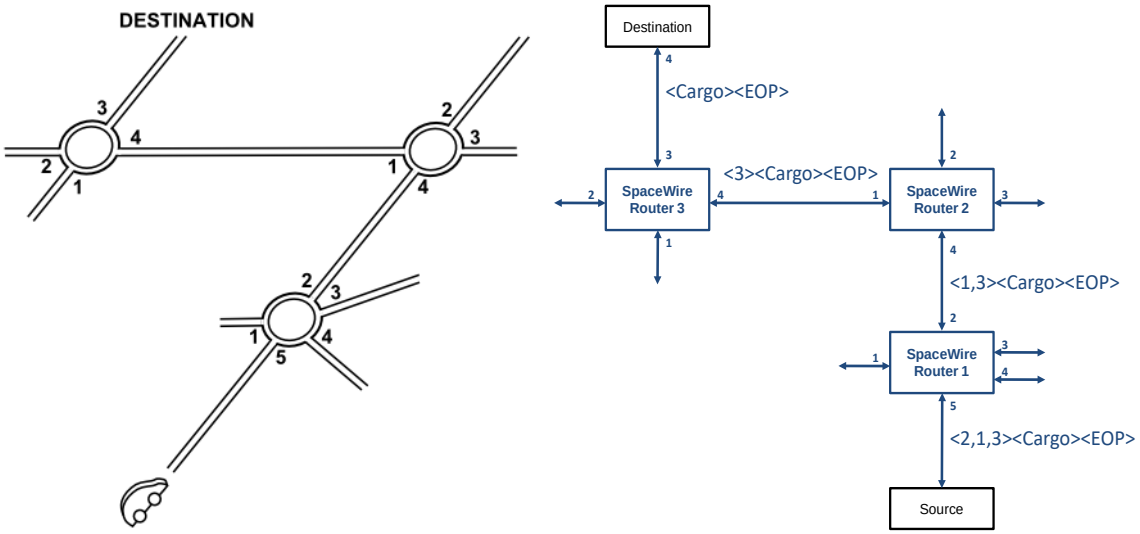


Figure 4-3: Path Addressing

Logical addressing can also be understood using the analogy of giving directions to a car driver. Logical addressing is like saying to the driver, "follow the signs to the City Centre at each of the road-junctions". This, of course, requires appropriate signs to be placed at each road-junction and each destination has to have a name or identifier so that it can be recognised on the signs. The logical address analogy is illustrated in Figure 4-4.

In a SpaceWire network using logical addressing, each destination is given an identifier, which is a number in the range 32 to 255. Each routing switch in the network has a routing table (like the sign at a road-junction) which specifies the port a packet is forwarded through for each possible destination identifier. The leading data character of a packet is set to the required destination identifier (e.g. 44 for City Centre). At each routing switch the leading data character is used to look up the appropriate directions from the routing table and the packet is forwarded accordingly. For logical addressing the leading data character is not discarded at each router, since it is needed to look up the path to follow at the next routing switch encountered. The use of logical addressing is illustrated in Figure 4-4. Logical addressing uses just a single data character to identify the destination which is in the range 32 to 255 so that it is not confused with path addresses.

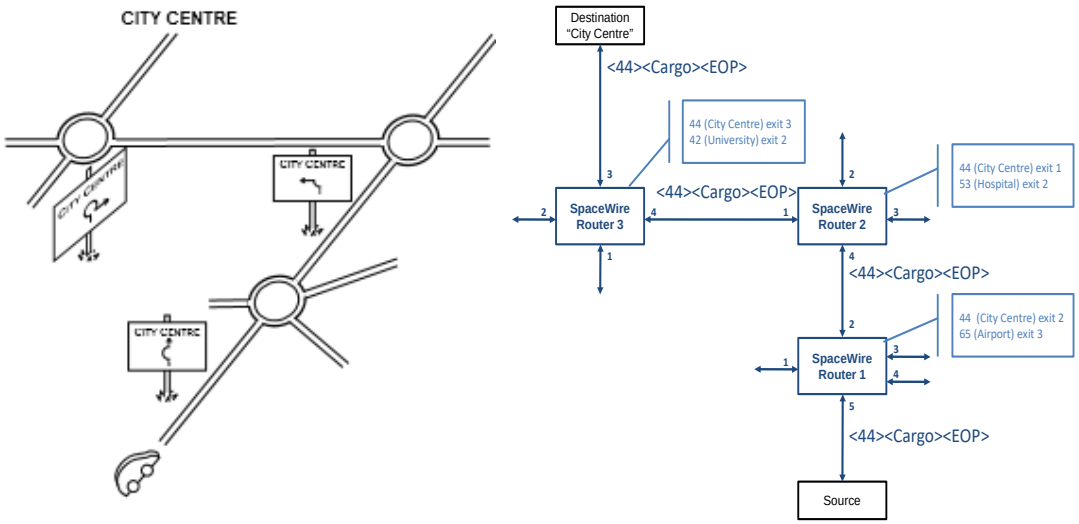


Figure 4-4 Logical Addressing

Logical addressing has the advantage that only a single address byte is required, but it requires the routing tables to be configured before it can be used. Path addressing requires a byte for each routing switch and does not depend on routing tables in the routers.

4.2.3.5 Time-Codes

SpaceWire time-codes provide a means of synchronising units across a SpaceWire system with low jitter. This time information is provided as "ticks", based on an incrementing value that is possibly synchronized to spacecraft time. The time-codes are transmitted over the SpaceWire network with high priority, alleviating the possible need for a separate time distribution network.

4.2.3.6 Distributed interrupts

SpaceWire distributed interrupts are used to signal events across a SpaceWire network with low jitter. Up to 32 different interrupts can be broadcast over the network with high priority.

5 Requirements

5.1 Overview

This clause provides the normative requirements for SpaceWire. It is separated into several functional layers.

- Clause 5.1 is a short overview of the following sub-clauses.
- Clause 5.2 describes the SpaceWire protocol stack and architecture of a SpaceWire Port.
- Clause 5.3 specifies the Physical layer of SpaceWire which covers the specification of the cables, connectors, cable assemblies, line drivers and line receivers.
- Clause 5.4 specifies the Encoding layer which covers the encoding of characters into symbols, symbol serialisation, data-strobe encoding of the serial bit stream, data-strobe decoding, symbol de-serialisation and decoding of symbols into characters.
- Clause 5.5 specifies the Data Link Layer of SpaceWire which is responsible for transferring packets and broadcast codes over a link, initialising the SpaceWire link, and managing the recovery from errors on the link.
- Clause 5.6 describes the Network layer which covers SpaceWire packets, nodes, routing switches, networks, time-code broadcasting and distributed interrupt operation.

5.2 Protocol stack and interface architecture

5.2.1 Protocol stack

a. The SpaceWire protocol stack shall have a Network layer, Data Link layer, Encoding layer, Physical layer and Management information base.

NOTE The SpaceWire protocol stack is illustrated in Figure 5-1.

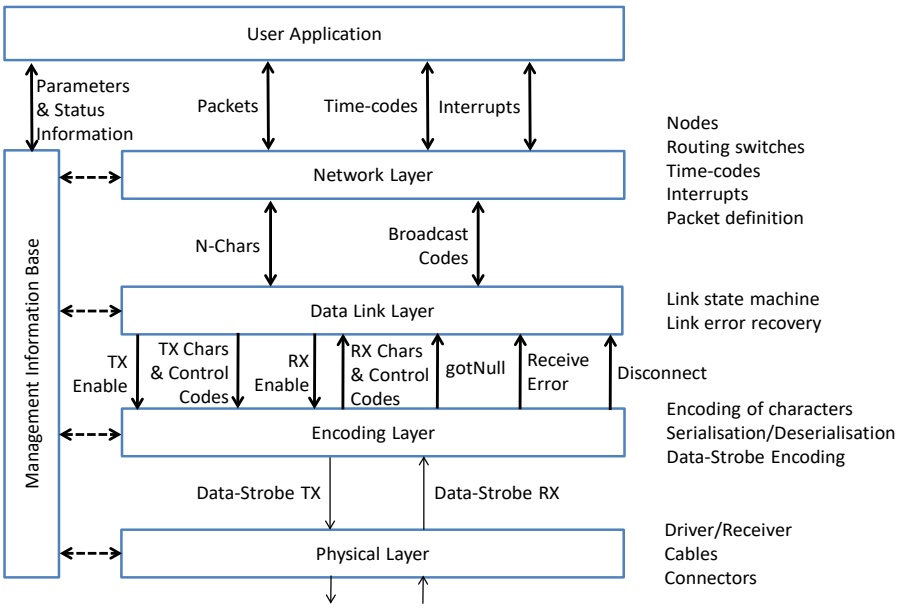


Figure 5-1: SpaceWire protocol stack

5.2.2 Network layer

a. The SpaceWire Network layer shall provide three principal services:

   1. A packet service which sends and receives packets over a SpaceWire network.
   2. A time-code service which sends and receives time-codes over a SpaceWire network.
   3. A distributed interrupt service which sends and receives distributed interrupts over a SpaceWire Network.

b. A SpaceWire implementation shall support the packet service, optionally support the time-code service and optionally support the distributed interrupt service.

c. The SpaceWire Network layer shall accept service requests from user applications.

d. The SpaceWire Network layer shall be responsible for transferring SpaceWire packets, time-codes and distributed interrupts over a SpaceWire Network.

e. The SpaceWire Network layer shall use the services of the SpaceWire Data Link Layer.

5.2.3 Data Link layer

a. The SpaceWire Data Link layer shall provide two services:

   1. An N-Char service which sends and receives N-Chars (the components of packets) over a SpaceWire link.

2. A broadcast code service which sends and receives broadcast codes (time-codes and distributed interrupt codes) over a SpaceWire link.

b. The Data Link layer shall accept service requests from the network layer.

c. The Data Link layer shall be responsible for establishing communications across the SpaceWire link, for controlling the flow of information over the link, for sending and receiving N-Chars, for sending and receiving broadcast codes and for re-establishing communications across the link after errors that occur over the link.

d. The Data Link layer shall use the services of the SpaceWire encoding layer.

5.2.4 Encoding layer

a. The SpaceWire encoding layer shall provide two services:

1. A character encoding service which encodes characters and control codes into symbols, serialise those symbols and data-strobe encode them ready for transmission over the SpaceWire physical layer.

2. A character decoding service which recovers the data bit stream from the data-strobe signals received from the physical layer, de-serialise that bit-stream, and decode the resulting symbols into characters and control codes.

b. The Encoding layer shall accept service requests from the SpaceWire data link layer.

c. The Encoding layer shall be responsible for encoding and decoding of characters into a form suitable for sending over the SpaceWire physical layer.

d. The Encoding layer shall use the services of the SpaceWire physical layer.

5.2.5 Physical layer

a. The SpaceWire Physical layer shall provide two services:

1. A transmit service which transmits the data and strobe signals from the encoding layer over the physical medium.

2. A receive service which receives the data and strobe signals from the physical medium and passes them to the encoding layer.

b. The SpaceWire Physical layer shall accept service requests from the encoding layer.

c. The SpaceWire Physical layer shall be responsible for transmitting and receiving the data and strobe signals over PCB tracks, connectors and cable assemblies.

5.2.6 Management Information Base

a. The SpaceWire Management Information Base shall provide two services:
   1. A set parameter service which writes control or configuration information to the other SpaceWire layers.
   2. A get status service which reads the status or current configuration or control values of the other SpaceWire layers.

b. The SpaceWire Management Information Base shall accept service requests from a user application.

c. The SpaceWire Management Information Base shall be responsible for configuring, controlling and monitoring the operation of the other SpaceWire layers.

d. The SpaceWire Management Information Base shall have direct access to the relevant configuration parameters, control parameters and status parameters in the network layer, data link layer, encoding layer and physical layer.

5.2.7 Service interfaces

The Service interfaces for each layer of the SpaceWire protocol stack are specified in clause 6.

5.2.8 SpaceWire Port architecture

a. A SpaceWire Port shall incorporate a SpaceWire port interface which comprises a packet transmit interface, a packet receive interface, an optional broadcast code transmit interface and an optional broadcast code receive interface.

b. A SpaceWire Port shall incorporate a transmit FIFO (TX FIFO) which stores N-Chars provided by the application via the SpaceWire port interface until they can be sent across the link.

c. A SpaceWire Port shall incorporate a receive FIFO (RX FIFO) which stores received N-Chars until they can be read by the application via the SpaceWire port interface.

d. A SpaceWire Port shall incorporate a flow control manager which manages the flow of data over the link, preventing data from being sent when there is no space for it in the receive FIFO.

e. A SpaceWire Port shall incorporate a link state machine which is responsible for controlling the starting of a link and its recovery from errors.

f. A SpaceWire Port shall incorporate a transmitter which is responsible for encoding the characters to be sent over the link into symbols, serialising those symbols into a bit stream and encoding the bit stream into a data and strobe pair of signals.

g. A SpaceWire Port shall incorporate a receiver which is responsible for decoding the received data and strobe pair of signals into a data bit

stream, de-serialising that bit stream into symbols and decoding the received symbols into characters.

h. A SpaceWire Port shall incorporate a pair of line drivers which convert the data and strobe signals into LVDS or LVTTL signals for driving across the link.

i. A SpaceWire Port shall incorporate a pair of line receivers which receive the LVDS or LVTTL signals and recover the data and strobe signals that were driven across the link.

NOTE The SpaceWire port architecture is illustrated in Figure 5-2.

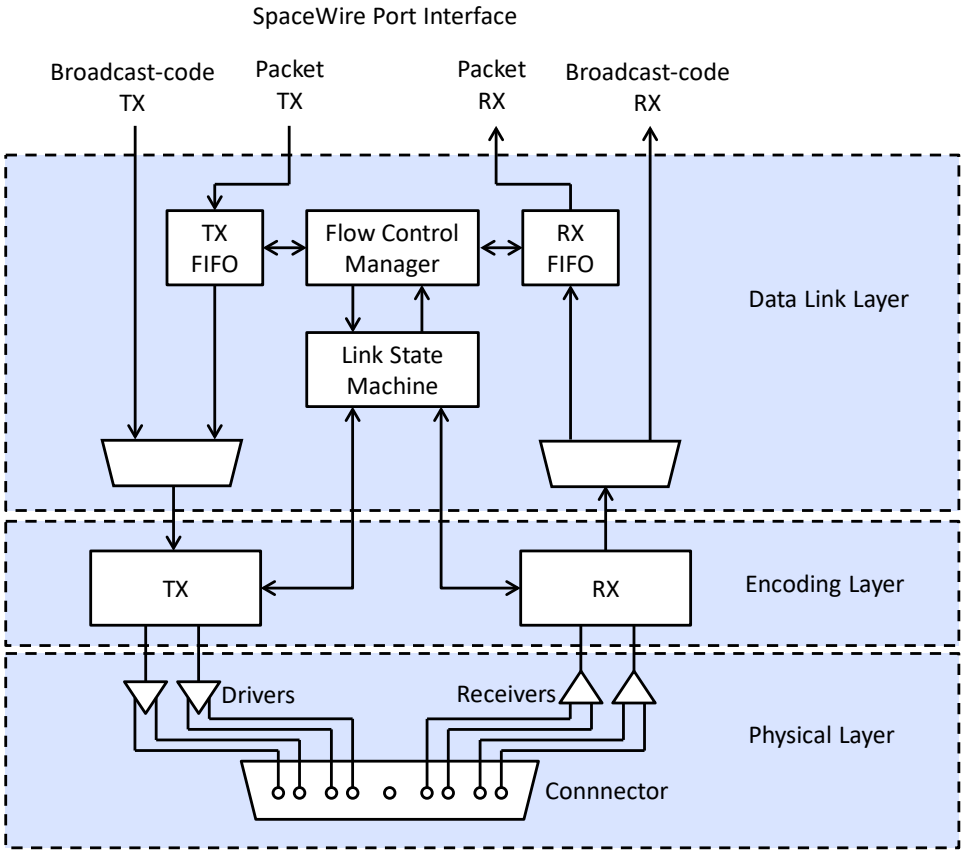


Figure 5-2: SpaceWire port architecture

5.3 Physical layer

5.3.1 Introduction

The Physical layer specifies the line drivers and receivers for transmitting and receiving the SpaceWire data and strobe signals over the physical medium, and specifies the cables, connectors, cable assemblies and PCB tracks that make up that physical medium.

5.3.2 Cables

5.3.2.1 Overview

The cable used in SpaceWire cable assemblies is specified in this clause. The cable assembly itself is specified in clause 5.3.4.

NOTE The rationale for the per-metre values below is based on the hypothesis of a SpaceWire link of up to 10 m length at 200 Mbps.

5.3.2.2 Cable construction

a. The SpaceWire cable shall carry four differential signals.

b. The SpaceWire cable shall comprise four screened twisted-pairs with an overall shield constructed according to ESCC 3902/003 or ESCC 3902/004.

NOTE ESCC 3902/003 describes variants of the SpaceWire cable to suit a range of spacecraft on-board applications. ESCC 3902/004 describes a low-mass variant of the SpaceWire cable.

c. For Type B cable assemblies, specified in clause 5.3.4.4, alternative forms of cable construction may be used provided that the cable characteristics fulfil the requirements in clauses 5.3.2.3 to 5.3.2.6.

NOTE This is to permit special forms of cable construction for specific applications.

5.3.2.3 Differential characteristic impedance

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

b. The characteristic impedance of each differential signal pair shall be verified according to MIL-DTL-17J:2014, Para. 4.8.7, using a time domain reflectometer (TDR) with a rise time of 150 ps or less.

5.3.2.4 Skew

a. The difference in delay of the two conductors forming a differential signal pair shall be less than 0,1 ns/m.

NOTE This is consistent with the ESCC 3902/003 and ESCC 3902/004 specifications.

b. The pair-to-pair skew for the four shielded, twisted pairs cable shall be less than 0,1 ns/m.

NOTE This is straightforward to measure, unlike intra-pair skew. This gives rise to 1 ns skew for a cable of 10 m length.

c. The inter-pair skew for the four shielded, twisted pair cable may be more than 0,1 ns/m provided that when incorporated in a cable assembly the

overall Data-Strobe skew budget requirement for the SpaceWire link is met, see clause 5.3.7.

5.3.2.5 Insertion loss

a. The average insertion loss in decibel per meter through each differential pair in a SpaceWire cable, shall be less than the values determined by equation [5-1] from 1 MHz to 500 MHz:

$$\alpha_{cable} = k1 \cdot \sqrt{f} + k2 \cdot f + \frac{k3}{\sqrt{f}} \; dB/m \qquad [5\text{-}1]$$

where:

f is the Nyquist frequency in MHz

*k*1 is a constant

*k*2 is a constant

*k*3 is a constant

NOTE The insertion loss value is retrieved at the Nyquist frequency applicable at the receiver, which e.g. is 100 MHz for a 200 Mbps link or 200 MHz for a 400 Mbps link.

b. For a SpaceWire cable constructed in accordance with 5.3.2.3a and 5.3.2.3b, and with differential pairs consisting of 28 AWG wires, the constants in Table 5-1 shall be used in conjunction with equation [5-1] to determine the upper limit of the average insertion loss in decibel per meter.

Table 5-1: Constants for a SpW cable using 28 AWG differential pairs

Wire Gauge	K1	K2	K3
28 AWG	4,5E-2	4,0E-4	5E-4

c. The values calculated in 5.3.2.5b should be rounded in accordance with Table 5-2.

NOTE The resulting insertion loss values in Table 5-2 from 1 MHz up to 500MHz concern the cable only. The additional losses due to the connectors of a SpaceWire cable assembly are not taken into account. The values in Table 5-2 have been rounded.

Table 5-2: Insertion loss values 28AWG SpW cable

Frequency (MHz)	**Max Insertion Loss** (dB/m)
1	0,05
10	0,14
100	0,49
250	0,81
500	1,21

d. For a SpaceWire cable constructed in accordance with 5.3.2.2a and 5.3.2.2b and with differential pairs consisting of 26 AWG wires, the constants in Table 5-3 shall be used in conjunction with equation [5-1] to determine the upper limit of the average insertion loss in decibel per meter.

Table 5-3: Constants for a SpaceWire cable using 26 AWG differential pairs

Wire Gauge	**K1**	**K2**	**K3**
26AWG	3,5E-2	1,8E-4	5E-4

e. The values calculated in 5.3.2.5d should be rounded in accordance with Table 5-4.

NOTE The resulting insertion loss values in Table 5-4 from 1 up to 500 MHz concern the cable only. The additional losses due to the connectors of a SpaceWire cable assembly are not taken into account. The values in Table 5-4 have been rounded.

Table 5-4: Insertion loss values for 26 AWG SpaceWire cable

Frequency (MHz)	**Max Insertion Loss** (dB/m)
1	0,05
10	0,11
100	0,37
250	0,6
500	0,87

f. For a SpaceWire cable constructed in accordance with 5.3.2.2c, the constants in Table 5-1 should be used in conjunction with equation [5-1] to determine the upper limit of the average insertion loss in decibel per meter.

5.3.2.6 PSNEXT and PSELFEXT

a. The SpaceWire cable shall conform to the PSNEXT and PSELFEXT values in Table 5-5 and Table 5-6 respectively.

NOTE These values are achievable on cable alone. On assemblies with MDM 9-pin connectors these values cannot be achieved. This performance can be met with EMI and crosstalk optimised connectors.

Table 5-5: Cable PSNEXT specification

Frequency (MHz)	**PSNEXT** (dB)
100	73,0
500	63,5
1000	39,0

Table 5-6: Cable PSELFEXT specification

Frequency (MHz)	**PSELFEXT** (dB)
100	65,0
500	50,5
1000	41,5

5.3.3 Connectors

5.3.3.1 General

a. The connectors for the SpaceWire cable assembly shall be either Type A, or Type B.

b. The Type A SpaceWire connector shall be a micro-miniature D-type connector with nine crimp contacts, as defined in ESCC 3401/029 and ESCC 3401/077.

c. The Type B SpaceWire connector shall be a connector with the following characteristics:

1. Contact pairs with differential impedance of 100 Ohm ± 6 Ohm.
2. Compatible with cables defined in 5.3.2.2.
3. Compatible with the space environment as defined in ESCC 3401.

NOTE This is to allow various connectors to be used for SpaceWire provided that they meet the

conform to 5.3.3.1c. This prevents a connector which has the wrong impedance, for example, from being considered as a SpaceWire connector. It is possible that several different connectors are all classed as Type B connectors, which is acceptable as the aim is to distinguish the Type A connectors from other connectors that have the correct impedance etc. for SpaceWire. Other non-impedance matched connectors are not recommended.

5.3.3.2 Connector with female contacts

a. Connectors with female contacts shall be used on circuit boards and unit assemblies to which SpaceWire cables are attached.

b. Connectors with female contacts used on circuit boards and unit assemblies shall have pin 3 connected to circuit ground.

NOTE This is to prevent a floating conductor in a Type A connector and cable assembly and to provide backwards compatibility with Type AL cable assemblies.

c. Connectors with female contacts that are attached to wire conductors shall:

1. Directly solder or crimp the wire conductors to the contacts.
2. Conform to ECSS-Q-ST-70-08 when conductors are soldered.
3. Conform to ECSS-Q-ST-70-26 when conducts are crimped.

d. The connection electrical resistance between a mated connector with male contacts and a connector with female contacts shall be less than 10 mOhm at DC.

NOTE The connection between the mated connector with male contacts and a connector with female contacts is a low broadband impedance connection. This is difficult to measure but is ensured by the design of the interface between the body of the mated connector with male contacts and connector with female contacts.

e. The connection electrical resistance between the body of the connector with female contacts and unit ground should be less than 10 mOhm at DC.

NOTE The body of the connector with female contacts is connected to the board or unit chassis with a low broadband impedance connection: for example, either by a mechanical mount of a connector on metal wall or on a PCB with chassis connection. This is difficult to measure but is ensured by the design of the interface between the body of the connector with female contacts and the unit chassis.

f. The connector with female contacts should include a conductive gasket for EMI improvement.

g. On flight units, connectors with female contacts and flying leads should be used for connection to a PCB.

NOTE This is to improve robustness against mechanical shock and vibration compared to PCB mounting connectors.

5.3.3.3 Connectors with male contacts

a. Connectors with male contacts shall be used on cable assemblies.

b. The SpaceWire conductors shall be directly soldered or crimped to the contacts.

c. Soldering shall conform to ECSS-Q-ST-70-08.

d. Crimping shall conform to ECSS-Q-ST-70-26.

5.3.3.4 Connector contact identification

5.3.3.4.1 Type A and AL connectors

a. The connector contacts shall be identified according to Table 5-7 and Figure 5-3.

NOTE 1 Pin 3 is connected to circuit ground when using cable assembly Type AL and is not connected when using cable assembly Type A. See clause 5.3.4.3 for details.

NOTE 2 Type A cable assemblies are also recommended for use with equipment that have pin 3 connected to circuit ground. A connection to the inner shields is not made, because pin 3 is not connected in Type A cable assemblies.

Table 5-7: Connector contact identification

Contact number	Signal name Connector with male contacts	Signal name Connector with female contacts
1	Din+	Din+
2	Sin+	Sin+
3	Not Connected when used with Type A cable assemblies. Circuit Ground when used with Type AL cable assemblies.	Circuit ground for both Type A and Type AL connectors mounted on PCBs or unit assemblies.
4	Sout-	Sout-
5	Dout-	Dout-

Contact number	Signal name Connector with male contacts	Signal name Connector with female contacts
6	Din-	Din-
7	Sin-	Sin-
8	Sout+	Sout+
9	Dout+	Dout+

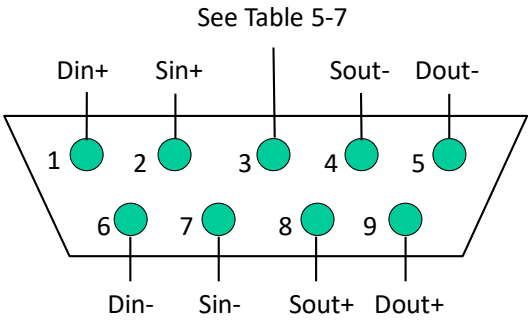


Figure 5-3: SpaceWire connector contact identification

5.3.3.4.2 Type B connector

a. The assignment of signals to connector contacts shall be defined by the supplier.

b. The two signals in each pair of conductors (Din+ and Din, Sin+ and Sin-, Dout+ and Dout-, Sout+ and Sout-) shall be assigned to adjacent pins of the connector.

c. No connector pin shall be connected to the cable shield.

d. The outer shield of the cable, where one is present, shall be circularly connected to the connector body.

e. If the connector design allows it, the inner shields shall be circularly connected to the connector body.

5.3.3.5 PCB mounting connector

a. Flying lead connectors should be used for connection to a PCB.

b. Flying lead connectors used for connection to a PCB should have:

1. All the leads cropped to a short length of less than 25 mm, each with a length difference of less than 3 mm;

2. The wires forming each differential signal pairs twisted together.

NOTE This helps to minimize the discontinuity in impedance caused by the connector.

c. PCB mounting right-angled connectors should not be used.

d. If a PCB mounting connector is used where one half of the differential signal pair in the connector is longer than that of the other half, track length compensation shall be performed at the connector end of the PCB tracks such that the two halves of the differential signal travel together down the PCB track or out of the connector to a cable assembly.

NOTE 1 For example, one half of a differential signal pair is D+ and the other half is D-.

NOTE 2 The reason for this is that in a right-angle connector the topmost row of pins on the right-angled connector has longer leads than the bottom row.

5.3.4 Cable assemblies

5.3.4.1 General

a. Cable assemblies shall consist of one of the following alternatives:

1. Two Type A connectors with male contacts joined by a length of cable;
2. Two Type B connectors joined by a length of cable;
3. A Type A connector with male contacts and a Type B connector joined by a length of cable.

b. A metal backshell shall be used for each connector to create, together with the outer shield, a continuous conductive barrier.

NOTE Its secondary purpose can be to facilitate the termination of the inner shields.

c. The outer shield of the cable shall be 360° terminated to the connector backshell via a low impedance connection which has a DC resistance of less than 10 mOhm.

d. There shall be a low connection impedance between the main body of the connector and the backshell which has a DC resistance of less than 10 mOhm.

e. A cable assembly shall have a low impedance between the connector body at each end of the assembled cable with a DC resistance of less than 1 Ohm."

NOTE A low broadband impedance connection is used connecting the shield to the body of the connector with male contacts in a cable assembly. This is difficult to measure but is ensured by the design of the termination of the outer shield to the body of the connector with male contacts.

f. When the inner shields of the SpaceWire cable are connected to the connector body, it shall be by a low impedance connection, which has a DC resistance of less than 10 mOhm.

g. Any pigtails used to connect the inner shields to the backshell shall be less than 2 cm in length.

5.3.4.2 Cable length and electrical performance

a. The maximum length of the cable assembly shall be determined by data to strobe skew, jitter, and signal attenuation.

NOTE 1 The maximum length depends on the specification of the cable. For shorter length and for small bend radii, flexible cable can be used as long as the signal quality is according to clauses 5.3.6 and 5.3.7.

NOTE 2 Longer length cables can be used at slow data signalling rates provided that the received signal levels and minimum edge separation timings are satisfied at the operating data signalling rate.

b. The maximum permitted inter-pair skew of a cable assembly shall be the maximum inter-pair skew of the cable, as defined in 5.3.2.4, plus 0.07 ns.

NOTE The value 0,07 ns covers the skew in the connectors at both ends of the cable assembly.

c. The maximum intra-pair skew permitted on a cable assembly shall be less than ±0,5 ns.

d. The insertion loss through each differential pair in a cable assembly shall be less than 7 dB at frequencies up to 1,5 times the data signalling rate.

NOTE 7 dB insertion loss corresponds to a signal of 250 mV dropping to approximately 110 mV when travelling one direction along the cable assembly, giving 10% margin above the 100 mV threshold. For a 200 Mbps signal, the frequency at which the insertion loss is measured is 300 MHz.

5.3.4.3 Cable assemblies for Type A connectors

5.3.4.3.1 General

a. Two types of standard cable assembly shall be permitted using Type A connectors: Type A and Type AL.

5.3.4.3.2 Cable assembly Type A

a. Cable assembly A shall use Type A connectors and cable with an outer shield as defined in clause 5.3.2.2.

b. The connector contacts for cable assembly Type A shall be terminated as shown in Figure 5-4 and Table 5-8.

NOTE The cable signal wires cross over to achieve the necessary transmit to receive interconnection, e.g. Dout+ is connected to Din+.

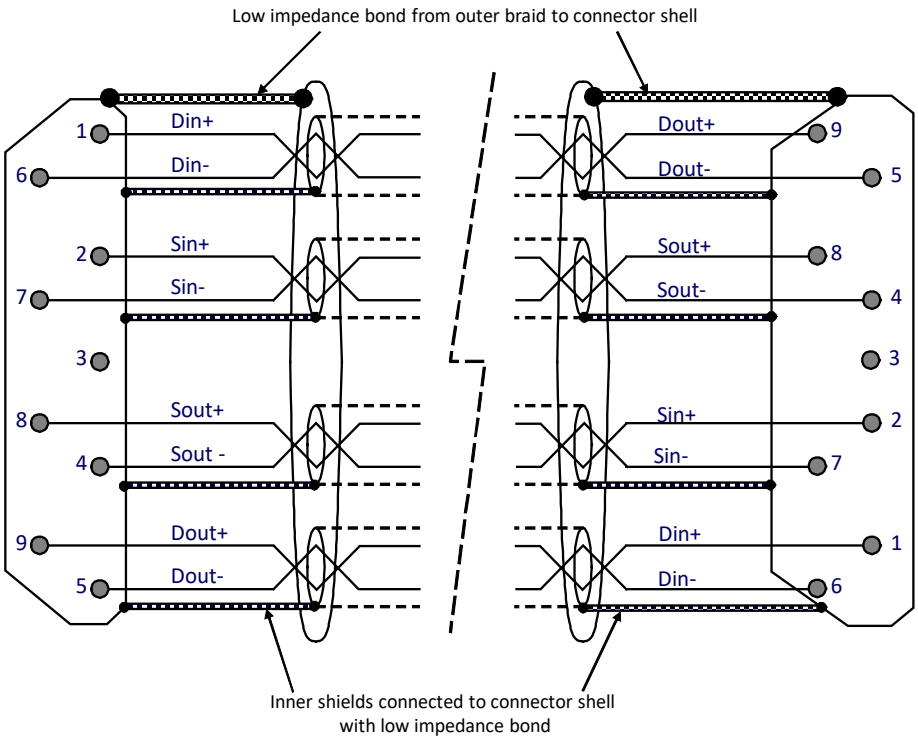


Figure 5-4: SpaceWire cable assembly Type A

Table 5-8: Cable assembly Type A signal wire connections

Signal on cable at end A	Connector pin at end A	Connections at each end of cable	Connector pin at B end	Signal on cable at end B
A-Din+	1	Connection — — Connection	9	B-Dout+
A-Din-	6	Connection — — Connection	5	B-Dout-
A-Sin+	2	Connection — — Connection	8	B-Sout+
A-Sin-	7	Connection — — Connection	4	B-Sout-
A-Outer Shield	Shell	Connection — — Connection	Shell	B-Outer Shield
A-Inner Shields of pairs 1/6, 2/7, 4/8 and 5/9)	Shell	Connection — — Connection	Shell	B-Inner Shields of pairs 1/6, 2/7, 4/8 and 5/9)
A-Sout-	4	Connection — — Connection	7	B-Sin-
A-Sout+	8	Connection — — Connection	2	B-Sin+
A-Dout-	5	Connection — — Connection	6	B-Din-
A-Dout+	9	Connection — — Connection	1	B-Din+
	3	Not Connected		
		Not Connected	3	

c. Pin 3 of the connector shall be left unconnected.

d. The individual shields of each of the four differential signal pairs shall be bonded to the connector shell.

e. The differential far end and near end crosstalk (FEXT and NEXT) between any two pairs in the SpaceWire cable assembly shall be less than -20 dB up to 1 GHz.

NOTE Most of the crosstalk is due to the connectors when Type A connectors are used.

f. A Type A cable assembly shall be labelled with the text "SpaceWire Type A".

5.3.4.3.3 Cable assembly Type AL

a. Cable assembly Type AL should not be used for new designs.

NOTE Cable assembly Type AL is the cable assembly specified in the previous issues of the SpaceWire standard, ECSS-E-ST-50-12C (31 July 2008), prior to the current revision 1. The “L” in the “Type AL” stands for “Legacy”.

b. Cable assembly AL shall use Type A connectors and cable with an outer shield as defined in clause 5.3.2.2.

c. The connector contacts for cable assembly Type AL shall be terminated as shown in Figure 5-5 and Table 5-9.

NOTE The cable signal wires cross over to achieve the necessary transmit to receive interconnection, e.g. Dout+ is connected to Din+.

d. 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 at the transmitting end.

e. The inner shields shall be isolated from the outer shield.

f. The inner shields connected to pin 3 at one end of the cable assembly shall be isolated from the inner shields connected to pin 3 at the other end of the cable assembly.

g. The differential far end and near end crosstalk (FEXT and NEXT) between any two pairs in the SpaceWire cable assembly shall be less than -20 dB up to 1 GHz.

NOTE Most of the crosstalk is due to the connectors.

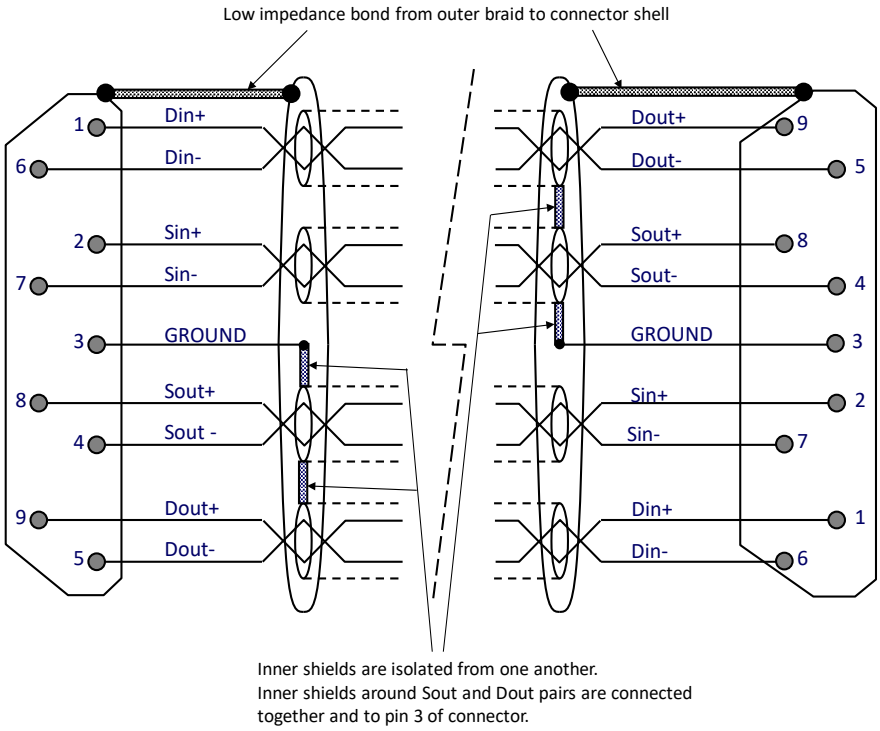


Figure 5-5: SpaceWire cable assembly Type AL

Table 5-9: Cable assembly Type AL signal wire connections

Signal on cable at end A	Connector pin at end A	Connections at each end of cable	Connector pin at B end	Signal on cable at end B
A-Din+	1	Connection —— Connection	9	B-Dout+
A-Din-	6	Connection —— Connection	5	B-Dout-
A-Sin+	2	Connection —— Connection	8	B-Sout+
A-Sin-	7	Connection —— Connection	4	B-Sout-
A-Inner Shields of pairs 1/6 and 2/7	No Connection	——————— Connection	3	B- (Inner Shields of pairs 5,9 and 4,8)
A-Outer Shield	Shell	Connection —— Connection	Shell	B-Outer Shield
A-Inner Shields of pairs 5/9 and 4/8	3	Connection ———————	No Connection	B-Inner Shields of pairs 1/6 and 2/7
A-Sout-	4	Connection —— Connection	7	B-Sin-
A-Sout+	8	Connection —— Connection	2	B-Sin+
A-Dout-	5	Connection —— Connection	6	B-Din-
A-Dout+	9	Connection —— Connection	1	B-Din+

5.3.4.4 Cable assembly Type B

a. Cable assembly Type B is a supplier-defined cable assembly which shall use matched impedance connectors.

b. Cable assembly Type B shall use Type B connectors as defined in 5.3.3.4.2 and cable as defined in clause 5.3.2.2.

c. The individual shields of each of the four differential signal pairs shall be bonded to the connector shell.

d. The differential far end and near end crosstalk (FEXT and NEXT) between any two pairs in the SpaceWire cable assembly shall be less than -50 dB up to 1 GHz.

NOTE Although mostly due to the connectors, this combines the contribution of the cables and of the connectors.

5.3.5 PCB tracks

5.3.5.1 LVDS PCB tracks

a. When using LVDS signals, the PCB tracks shall be differential tracks with a differential impedance of 100 Ohm ± 10 Ohm.

b. The difference in length between the two tracks forming a differential pair shall be less than 3 mm.

NOTE 3 mm corresponds to approximately 20 ps difference in delay which is less than 10% of the rise or fall time of the LVDS signal, so there is an insignificant effect on the differential signal.

c. The difference in length between the pair of tracks used for data and the pair of tracks used for strobe shall be less than 5 mm.

NOTE 5 mm corresponds to a data-strobe skew of approximately 33 ps, which is small compared to the other skew factors, while being straightforward to achieve with PCB layout.

d. The use of vias for LVDS PCB tracks should be minimised.

e. The distance from an external termination resistor to the inputs of a line receiver should be kept as short as possible and less than 20 mm.

NOTE 20 mm corresponds to the distance travelled by the signal during its rise/fall time. The minimum LVDS rise/fall time is 260 ns which corresponds to around 38 mm. The 20 mm recommendation is half of this value.

5.3.5.2 LVTTL PCB tracks

a. When using LVTTL signals, the PCB track shall have an impedance of 50 Ohm ± 5 Ohm.

NOTE Correct operation is not guaranteed when the PCB track is interrupted by connectors, unless those connectors are 50 Ohm matched impedance. When connectors that are not matched impedance are used, reflections can occur that are likely to degrade the signal.

b. When running over PCB tracks which are longer that the rise or fall time of the LVTTL signal times the speed of propagation of the signal along the tracks, the source impedance of the LVTTL driver shall be matched to the line impedance by adding a series resistor close to the source or parallel termination close to the signal destination.

NOTE A PCB track of length 50mm corresponds to a delay of around 333 ps which is a typical minimum value for the rise and fall time of the LVTTL signal.

c. When a PCB track is being used to connect a SpaceWire interface to a nominal and redundant pair of line drivers/receiver, where only the nominal or redundant device is activated at the same time, one or other of the following approaches should be used:

1. The track lengths from driver to all sources is kept less than 50 mm;
2. The line driver/receiver devices are placed close together and the following conditions applied:
   (a) The signals to the driver are terminated at the source with a source termination resistor and the PCB track run to the closest driver device and then on to the further driver device.
   (b) The PCB tracks carrying the signals from the receivers are each run to a source termination resistor as close as possible to and equidistant to the two receivers and then the other side of the source termination resistor run to the SpaceWire interface device.

d. The difference in length between the PCB track used for data and the track used for strobe shall be less than 5 mm.

5.3.6 Line drivers and receivers

5.3.6.1 General

a. SpaceWire shall use LVDS signals for transferring data.

b. When a SpaceWire signal runs entirely inside a unit, LVTTL or LVCMOS may be used for driving SpaceWire data and strobe signals over PCB tracks and cables.

c. When LVDS signalling is unable to achieve mission objectives, other forms of cable line driver and line receiver may be used with SpaceWire, provided that the connectors used are different from the SpaceWire Type A and Type B connectors.

NOTE The different type of connector used with non-LVDS line drivers/receivers is to prevent inadvertent connection of other drivers/receivers to an LVDS driver/receiver which then causes damage to the LVDS driver/receiver.

5.3.6.2 LVDS

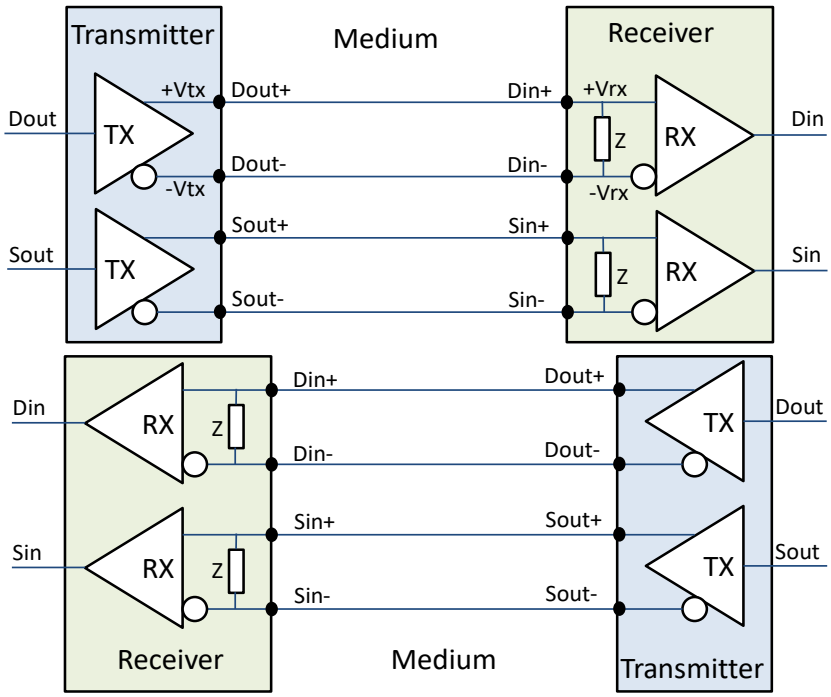
5.3.6.2.1 General

a. Low voltage differential signalling or LVDS as specified for a point-to-point link in the TIA-644-A:2012 standard shall be used for driving SpaceWire data and strobe signals over SpaceWire cable assemblies, as illustrated in Figure 5-6.

NOTE 1 LVDS uses balanced signals with low voltage swing (350 mV typical). The balanced or differential signalling provides adequate noise margin to enable the use of low voltages in practical systems. Low voltage swing means short signal transition times can be achieved with low power consumption. LVDS is appropriate for connections between boards in a unit, and unit to unit interconnections over distances of 10 m or more depending on the data signalling rate. For longer cables the maximum achievable data rate is lower.

NOTE 2 The TIA-644-A standard describes the way in which the specifications can be measured and verified.

b. SpaceWire using LVDS line drivers and line receivers shall be referred to as SpW-LVDS.



Z is the 100 Ω termination impedance

Figure 5-6: SpaceWire LVDS

5.3.6.2.2 LVDS transmit signals

a. When terminated, for measurement purposes, by two 50 Ohm ± 1% termination resistors in series forming the required 100 Ohm ± 1% termination impedance, the two outputs of the LVDS line driver (Out+ and Out-) shall have a common mode voltage, Vcm, measured at the junction of the two 50 Ohm ± 1% termination resistors, of 1,125 V to 1,45 V, as illustrated in Figure 5-7.

NOTE This is to permit use of drivers which have slightly higher Vcm than the TIA-644-A specification permits (1,375 V).

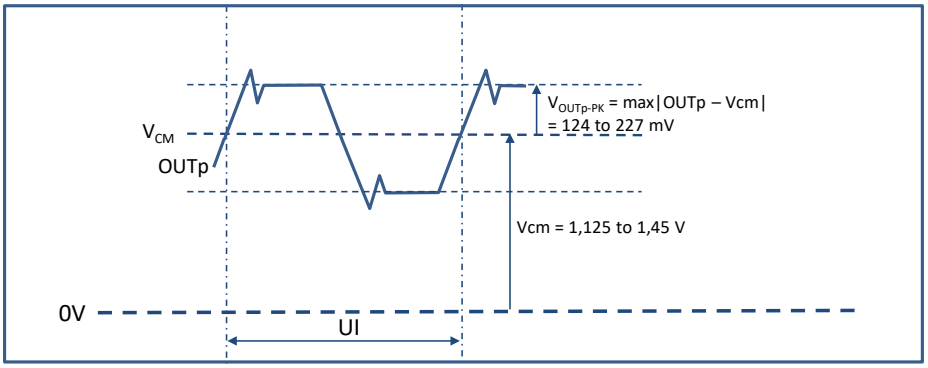
b. When there is large common-mode noise on a signal, an alternative, equivalent termination scheme may be used provided that the termination is matched with the nominal media impedance.

NOTE Large common-mode noise is an indication that something is wrong with the circuit design or grounding arrangement. An alternative termination scheme is one possibility for reducing the common mode noise.

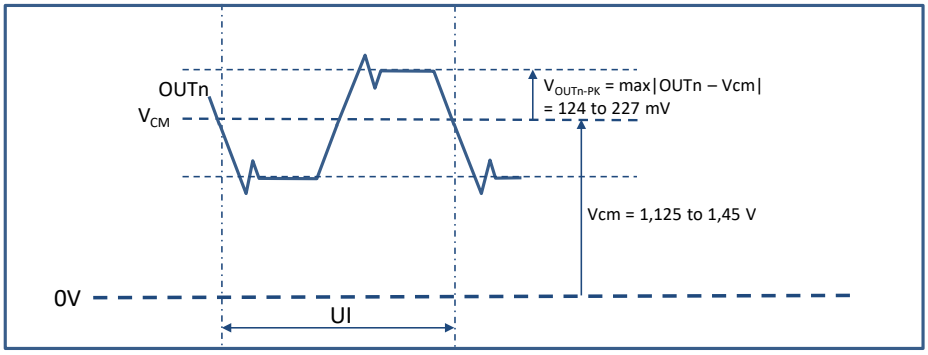
c. When terminated, for measurement purposes, by a 100 Ohm ± 1 Ohm termination resistor, the two outputs of the LVDS line driver (Out+ and Out-) shall have amplitude, $V_{OUTp\text{-}PK}$ or $V_{OUTn\text{-}PK}$, of 124 mV to 227 mV, and a differential amplitude, of $V_{DIFF\text{-}PK}$, of 247 mV to 454 mV, as illustrated in Figure 5-7.

NOTE 1 Figure 4 of TIA-644-A includes two 3,74 kOhm resistors in the test circuitry which simulate multiple line receivers connected to the line driver. These are not necessary when testing SpaceWire LVDS because the SpaceWire link is point-to-point NOT multi-drop.

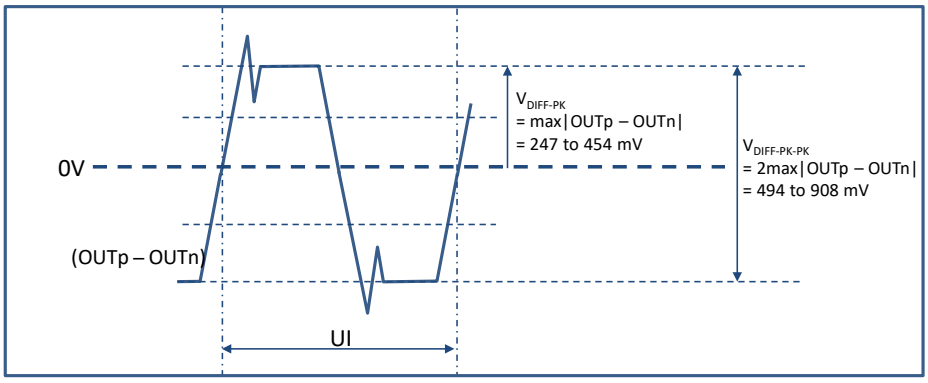
NOTE 2 A higher value of $V_{DIFF-PK}$, within the specified range, allows longer cable lengths, see clause 5.3.6.2.3.



(a) Positive side of the LVDS signal (OUTp)



(b) Negative side of the LVDS signal (OUTn)



(c) Differential LVDS signal (OUTp – OUTn)

Figure 5-7: LVDS line driver output signals

d. When a logic 1 is transmitted, V_{OUTp} shall be greater than V_{OUTn}.

e. When a logic 0 is transmitted, V_{OUTp} shall be less than V_{OUTn}.

f. The steady state difference in magnitude of the common mode voltage, Vcm, when transmitting a logic 1 compared to when transmitting a logic 0 shall be less than 50 mV.

g. The steady state difference in magnitude of V_{OUTp} or V_{OUTn} when transmitting logic 1 compared to when transmitting logic 0 shall be less than 50 mV.

h. The differential output of the line driver, OUTp - OUTn, shall have signal edges with the following characteristics:

1. Be monotonic.
2. Have rise time (Tr) and fall time (Tf) of at least 260 ps and less than 0,3 times the bit period (T), as illustrated in Figure 5-8.

NOTE 1 This corresponds to clause 4.1.4 of TIA-644-A.

NOTE 2 A link speed of 10 Mbps, gives rise and fall times of 30 ns. A 30 ns transition generates significantly less noise than a 3 ns transition. In practice, many LVDS drivers transition in less than 500 ps, potentially generating significant noise. Noise is minimized by maximizing the transition time within the constraint of the above clause.

NOTE 3 For receiving signals with more than 3 ns transition times, an accepted way to improve the noise immunity for both polarities of a slow or potentially non-monotonic transition at the receiver is to use receivers with hysteresis.

i. The differential output of the line driver, OUTp - OUTn, should rise and fall monotonically with a rise time (Tr) and fall time (Tf) of less than 3 ns.

NOTE The 3 ns limit is an additional recommended constraint to TIA-644-A so that for slow data rates the edge speed is sufficiently fast to ensure correct operation of the line receiver in a noisy environment.

j. Ringing on the differential output of the line driver, OUTp - OUTn, shall not be greater than $\pm 0.4 V_{OUT-PK}$.

NOTE This is illustrated in Figure 5-8, which shows the same information as TIA-644-A Figure 8, but referenced to 0V differential, rather than the minimum steady state signal.

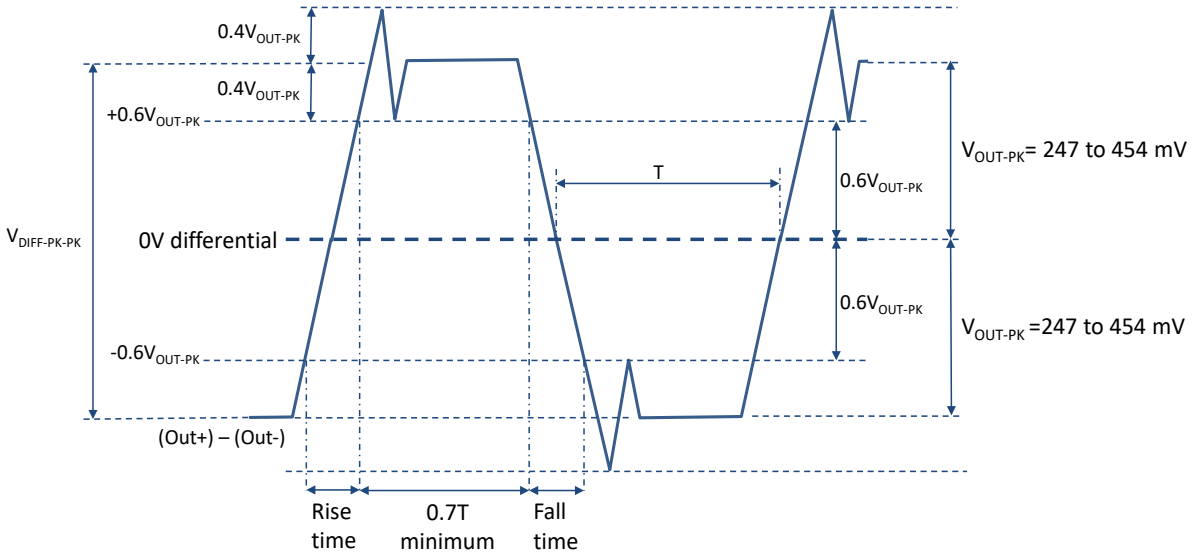


Figure 5-8: LVDS line driver differential output signal

k. The maximum dynamic difference in magnitude between V_{OUTp} or V_{OUTn} shall be less than 150 mV.

NOTE This corresponds to TIA-644-A clause 4.1.5.

5.3.6.2.3 LVDS line receiver characteristics

a. The receive signals shall be terminated by a 100 Ohm ± 10 Ohm termination resistor.

b. The receive signals should be terminated by a 100 Ohm ± 1 Ohm termination resistor when an external termination resistor is being used

NOTE This reduces reflections from the termination resistor and most surface mount resistors are now 1% tolerance.

c. The SpaceWire LVDS line receiver input characteristics and sensitivity shall be as defined in TIA-644-A:2012.

d. A differential signal greater than +100 mV shall result in logic 1 at the line receiver output.

NOTE 1 A differential signal greater than +100 mV occurs when V_{INp} is greater than V_{INn} by more than 100 mV.

NOTE 2 The common mode voltage plus or minus the differential voltage is always in the range 0 V to +2,4 V, as specified in 5.3.6.2.3f. This corresponds to clauses 4.1 and 4.2.4 of TIA-644-A which define the signal sense and the receiver sensitivity respectively.

e. A differential signal less than -100 mV shall result in logic 0 at the line receiver output.

NOTE 1 A differential signal less than -100 mV occurs when V_{INp} is less than V_{INn} by more than 100 mV.

NOTE 2 This corresponds to clauses 4.1 and 4.2.4 of TIA-644-A which define the signal sense and the receiver sensitivity respectively.

f. The line receiver shall maintain correct operation for differential input voltages of up to 600 mV magnitude.

NOTE This corresponds to the maximum permitted receiver differential signal in clause 4.2.4 of TIA-644-A.

g. The line receiver shall tolerate a voltage on the receiver inputs in the range 0 V to +2.4 V relative to the line receiver ground and operate correctly.

NOTE The line receiver is able to operate correctly with a common mode voltage of +0,05 V to +2,35 V when the differential signal is at ±100 mV and with a common mode voltage of +0,3 V to +2,1 V when the differential signal is at ±600 mV. This is as defined in clause 4.2.4 of TIA-644-A.

h. The line receiver should tolerate a voltage on the line receiver inputs beyond the range 0 V to +2,4 V relative to the line receiver ground and operate correctly.

NOTE This is so that the line receiver is more tolerant to voltage difference in the grounds between the line driver and the line receiver.

5.3.6.2.4 Potential difference between two ends of SpaceWire link

a. The maximum potential difference between the circuit ground at one end of a SpaceWire link and the circuit ground at the other end of that SpaceWire link shall be less than ±1 V.

NOTE 1 This corresponds to TIA-644-A clause 5d.

NOTE 2 The common mode voltage can be greater than expected when there is a potential difference between the grounds of the equipment at either end a SpaceWire link. This can damage the LVDS line driver and line receiver. It is preferable to use LVDS line drivers and line receivers with wider common mode voltage tolerance than provided by TIA-644-A.

NOTE 3 The TIA-644-A standard provides limited common mode voltage and many devices can be damaged if the voltage on their pins is less

than -0,3 V or more than 3,9 V, see clauses 5.3.6.2.5.j & k.

NOTE 4 ECSS-E-HB-20-07 describes ways of reducing the potential common mode difference between two units connected by SpaceWire. See clauses 5.2.3.6 and 6.1.2.5.3 through to the end of clause 6.1 in that handbook.

5.3.6.2.5 Fail safe operation of LVDS

a. The LVDS line driver and line receiver circuits shall not be damaged when powered or not powered under the following conditions:
   1. Transmitter outputs are open circuit.
   2. Transmitter outputs are shorted together.
   3. Transmitter outputs are shorted to local ground.
   4. Receiver inputs are open circuit.
   5. Receiver inputs are shorted together.
   6. Receiver inputs are shorted to local ground.

b. When either or both of the line driver outputs are short circuited to ground, the current out of each of the outputs shall be less than 24 mA.

c. When the two line driver outputs are short circuited to each other, the current flowing between the two outputs shall be less than 12 mA.

d. When any of the following fault conditions occur, the line receiver outputs shall not oscillate and be locked to high logic level provided that a noise threshold of 10 mV is not exceeded at the line receiver input.
   1. Driver not powered.
   2. Driver disabled.
   3. Receiver inputs open circuit.

      NOTE 1 A driver is disabled when its outputs are high impedance.

      NOTE 2 The receiver inputs are open circuit when a cable or wire in the cable is disconnected.

e. When the driver is not powered, its output should be high impedance with a leakage current of less than 20 μA.

f. When the line receiver is not powered, its input should be high impedance with a leakage current of less than 20 μA.

   NOTE This excludes current flowing in any external biasing resistors.

g. When external biasing resistors are used to provide fail safe operation or to increase immunity to noise on the line receiver input, the bias current provided by the external biasing resistors should be less than 0,25 mA.

   NOTE This is 10 % of the expected minimum current through the termination resistor.

h. A single fault should not lead to a driver emitting a voltage outside the range 0 V to +3,6 V relative to the driver ground reference.

i. A single fault should not lead to a line receiver emitting a voltage outside the range 0 V to +3,6 V relative to the line receiver ground reference.

j. A driver output should withstand without failing a direct connection to a voltage between -0,3 V and +3,9 V relative to the driver ground reference.

k. A line receiver input should withstand without failing a direct connection to a voltage between -0,3 V and +3,9 V relative to the receiver ground reference.

5.3.6.3 LVTTL

a. LVTTL shall only be used for driving SpaceWire data and strobe signals over short distances of less than 30 cm over PCB tracks and cables inside a unit.

NOTE LVTTL is only intended to be used internally within a unit. LVTTL is specified in the JESD8C.01 standard.

b. SpaceWire using LVTTL line drivers and line receivers shall be referred to as SpW-LVTTL.

5.3.7 Data-Strobe skew

5.3.7.1 General

The components of the data-strobe skew across a SpaceWire link are separated into the following contributions: transmitter, cable assembly and receiver. This is illustrated in Figure 5-9

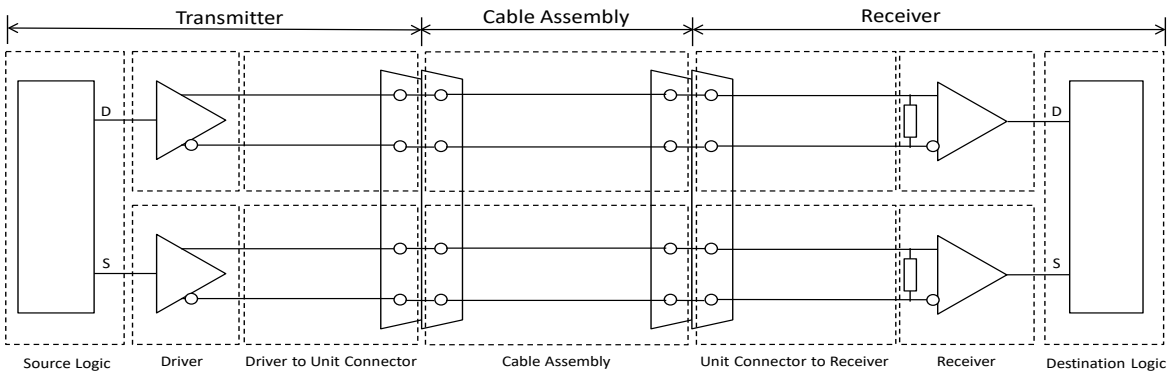


Figure 5-9: Physical layer components

5.3.7.2 Contributers to the Data-Strobe skew

a. The system architect shall specify the operational data signalling rate of the SpaceWire link at system level.

b. The system architect shall define the margin to be applied on the operational data signalling rate.

c. The system architect shall maintain a Data and Strobe skewand jitter budget along the project life-time

NOTE The main objective of the skew and jitter budget is to guarantee that the operational data signalling rate of the SpaceWire signal is reached with an appropriate margin.

d. The system architect shall specify nominal and worst-case contributions of skew and jitter of Data and Strobe at component level: units and cable assembly.

NOTE The worst-case values include allowance for radiation and ageing effect

e. The unit and cable assembly suppliers shall provide designs that are compliant to the specified values of data and strobe skew and jitter for the unit and cable assembly respectively.

f. The unit and cable assembly suppliers shall demonstrate compliance to the specified values of data and strobe skew and jitter initially by analyses followed by test and/or analysis on the developed models for the unit and cable assembly respectively.

g. The unit manufacturer shall provide the specification for the worst-case skew between the differential data and strobe output signals at the SpaceWire connector of a unit ($DSskew_{OUT}$), including the effect of transmitter jitter.

h. The cable assembly manufacture shall provide the worst-case skew between the differential data and strobe signals ($DSskew_{CA}$).

NOTE This maximum permitted value for the differential data to strobe skew in a cable is given by the cable assembly inter-pair skew which is specified in clause 5.3.4.2.

i. The cable assembly manufacturer shall provide the worst-case frequency dependent and data-pattern dependent jitter of the data or strobe signals ($Jitter_{CA}$).

j. The unit manufacturer shall provide the specification for the minimum tolerated separation between signal edges ($Minsep_{IN}$) which can be data to strobe separation, data to data or strobe to strobe.

k. It is possible for the connection between the unit at one end of a SpaceWire link and the unit at the other end to comprise more than one SpaceWire cable assembly, in which case the skew and jitter contributions of each cable assembly shall be aggregated into overall figures to be used in the worst-case performance analysis.

l. The minimum bit unit interval, T_{UIMIN}, shall be the sum of the following components.

1. Maximum, worst case, source unit output D-S skew, $DSskew_{OUT}$;
2. Maximum, worst case, cable assembly differential D-S skew, $DSskew_{CA}$;
3. Maximum, worst case, cable assembly differential jitter, $DSjitter_{CA}$;

4. Receiving unit minimum tolerated separation between signal edges, $Minsep_{IN}$;
5. 10% margin.

NOTE A table giving an example calculation is provided in Table 5-10.

Table 5-10 Example calculation of maximum bit rate

Factor	Abbreviation	Value (ps)	Sum (ps)
Maximum, worst case, source unit output D-S skew	$DSskew_{OUT}$	1234	1234
Maximum, worst case, cable assembly differential D-S skew (e.g. 5 m cable)	$DSskew_{CA}$	500	1734
Maximum, worst case, data or strobe jitter in the cable assembly	$Jitter_{CA}$	500	2234
Receiving unit minimum tolerated separation between signal edges	$Minsep_{IN}$	1384	3618
10% Margin	Margin	362	3980
Minimum Bit Unit Interval	T_{UIMIN}		3980
Maximum bit rate			251 Mbps

m. The maximum operating frequency of the SpaceWire link shall be less than the inverse of the minimum bit unit interval, T_{UIMIN}.

NOTE 1 Figure 5-10 illustrates the effect of skew and jitter on the Data and Strobe signals, where the parameters are as follows:

- 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 can 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 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 data signalling rate.

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

NOTE 2 The t_{dclk} and t_{hold} parameters can be combined into a minimum specification for the separation of consecutive edges on the Data and Strobe signals at the input to the decoder, $t_{ds} = t_{dclk} + t_{hold}$.

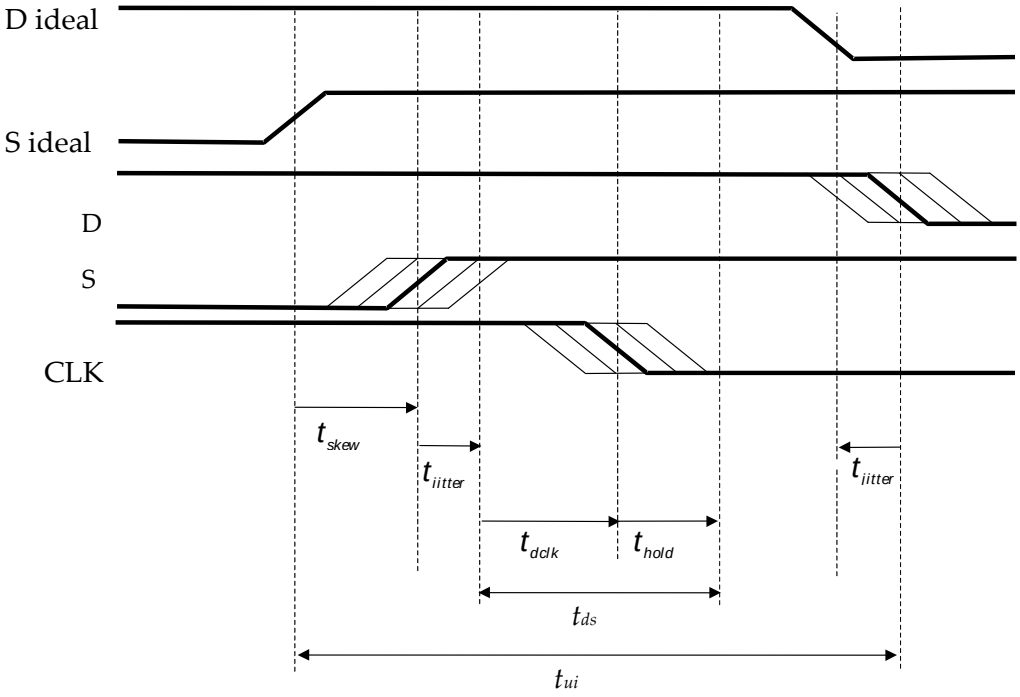


Figure 5-10: Skew and jitter

5.3.8 Physical layer management parameters

a. The following management parameters should be provided to configure, control and monitor the physical layer.

1. Enable line driver (port), which when set enables the line driver of the specified port so that it can transmit information.
2. Enable line receiver (port), which when set enables the line receiver of the specified port so that it can receive information.
3. Connector type and pin out for each connector.

NOTE The connector type and pin out management parameters are pertinent to the Type B connector.

5.4 Encoding layer

5.4.1 Introduction

The encoding layer specifies the encoding/decoding of characters into symbols, the serialisation/de-serialisation of the encoded symbols into a bit stream, and the data-strobe encoding/decoding of the serial bit stream.

5.4.2 Serialisation and de-serialisation

a. Characters and control codes shall be encoded, serialised, data-strobe encoded and transmitted in the order in which they are received from the data link layer.

b. Received characters and control codes shall be passed to the data link layer in the order in which they are received.

c. Only received characters and control codes without a parity error shall be passed to the data link layer.

d. Only while gotNull is asserted, see clause 5.4.5, which indicates that the first Null has been received, shall characters and control codes that are received be passed to the data link layer.

e. The operation of the encoding layer shall be controlled by the data link layer using the following control flags:

   1. Transmit Enable, which enables the transmitter (character encoder, serialiser, data-strobe encoder and line driver) when asserted and resets it when de-asserted.

   2. Receive Enable, which enables the receiver (line receiver, data-strobe decoder, de-serialiser and character decoder) when asserted and resets it when de-asserted.

f. The encoding layer shall inform the data link layer of changes in the encoding layer indicated by the following status flags:

   1. Disconnect, which indicates that the link has been disconnected.

   2. Receive error, which indicates that an error has been detected in a received symbol.

   3. gotNull, which indicates that the first Null control code has been received without any parity errors.

      NOTE A parity error is an example of a receive error.

5.4.3 Character and control code encoding

5.4.3.1 Data characters

a. A data character shall be encoded in 10 bits with the resulting data symbol containing a parity bit, a data-control flag and eight bits of data as illustrated in Figure 5-11.

b. The data-control flag shall be set to zero to indicate that the current symbol holds a data character.

c. The eight-bit data value shall be transmitted least significant bit first.

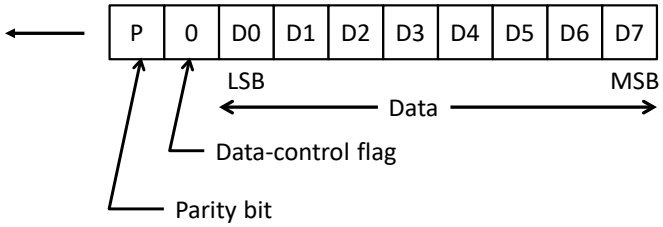


Figure 5-11: Data character encoding

5.4.3.2 Control characters

a. A control character shall be encoded in four bits with the resulting control symbol containing a parity bit, a data-control flag and a two-bit control type as illustrated in Figure 5-12.

b. The data-control flag shall be set to one to indicate that the current symbol holds a control character.

c. A flow control token (FCT) shall be encoded as a control symbol with the two-bit control type set to 0b00.

NOTE The FCT is used in the data link layer to manage the flow of N-Chars over a SpaceWire link.

d. An end of packer marker (EOP) shall be encoded as a control symbol with the two-bit control type set to 0b10.

NOTE The EOP is used in the data link layer to mark the end of a packet indicating that the packet was transferred with no parity error being detected.

e. An error end of packet marker (EEP) shall be encoded as a control symbol with the two-bit control type set to 0b01.

NOTE The EEP is used in the data link layer to terminate a packet prematurely at the point where the error occurred; indicating that an error occurred while the packet was being transferred.

f. An escape character (ESC) shall be encoded as a control symbol with the two-bit control type set to 0b11.

NOTE The ESC is used to form the Null and Broadcast control codes.

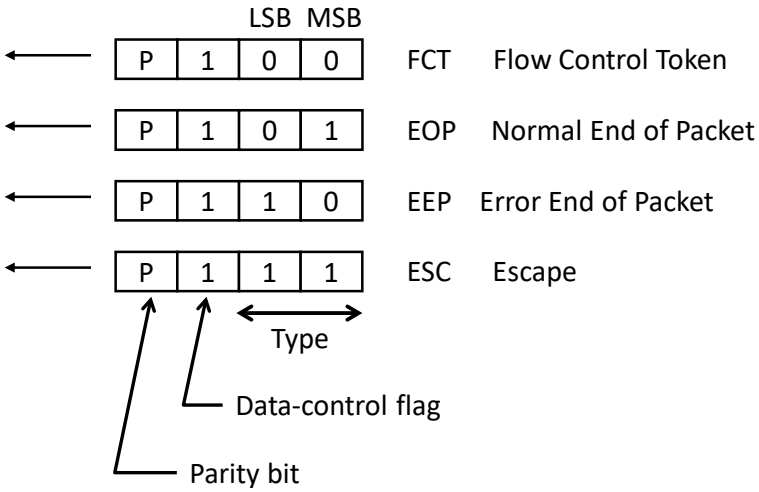


Figure 5-12: Control character encoding

5.4.3.3 Control codes

a. The ESC shall be used as the first character in a control code.

b. The Null control code shall be encoded as an ESC followed by a flow control token (FCT) as illustrated in Figure 5-13.

NOTE 1 The parity bit (P) in the middle of the control code is zero, in accordance with clause 5.4.3.4).

NOTE 2 Null is passed to the encoding layer by the data link layer whenever a link is not sending data or control symbols, to keep the link active and to support link disconnect detection (see clause 5.4.8).

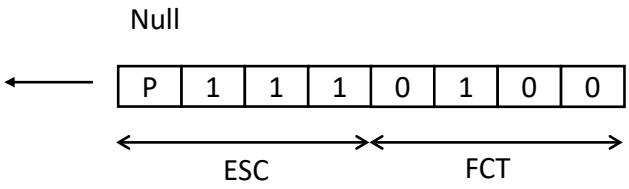


Figure 5-13: Null control code encoding

c. The broadcast code (BC) shall be encoded as an ESC followed by a single data character as illustrated in Figure 5-14.

NOTE 1 The parity bit (P) in the middle of the broadcast code is one, in accordance with clause 5.4.3.4).

NOTE 2 The broadcast code is used for time-codes and distributed interrupt codes.

d. The eight bits of data in the data character of a broadcast code shall be separated into two fields:

1. The most significant two bits (B7:6) form the type field.
2. The least significant six bits (B5:0) form the value field.

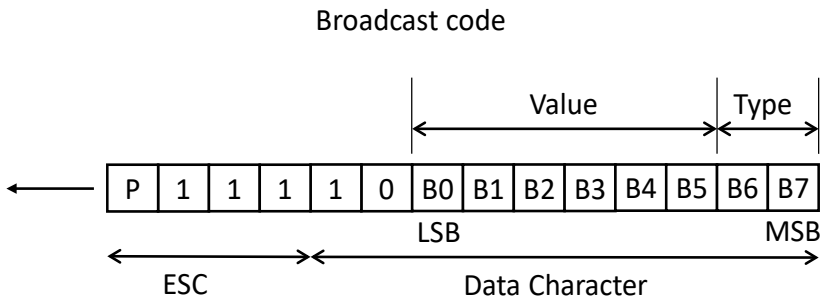


Figure 5-14: Broadcast code encoding

5.4.3.4 Parity

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

b. The parity bit shall cover the previous eight bits of an encoded data character or two bits of an encoded control character, the current parity bit, and the current data-control flag, as illustrated in Figure 5-15.

c. The parity bit shall be set to produce odd parity so that the total number of 1's in the field covered is an odd number.

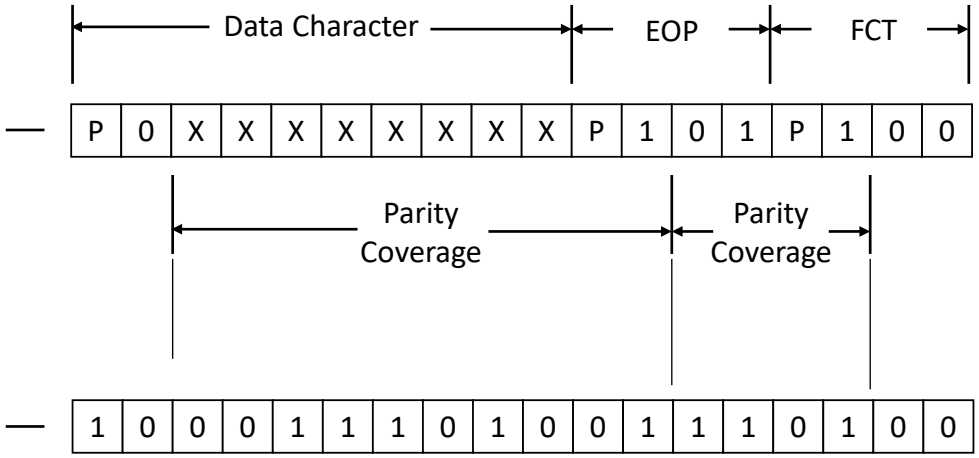


Figure 5-15: Parity coverage

5.4.4 Data strobe encoding and decoding

a. The data bit stream resulting from the serialisation of the symbols to be transmitted shall be encoded using two signals, Data and Strobe, as follows:

1. The Data signal is high when the data bit is 1 and low when the data bit is 0.

2. The Strobe signal changes state whenever the Data has the same value from one bit to the next.

NOTE 1 The Data signal follows the value of the data bit stream.

NOTE 2 The Data-Strobe encoding is illustrated in Figure 5-16.

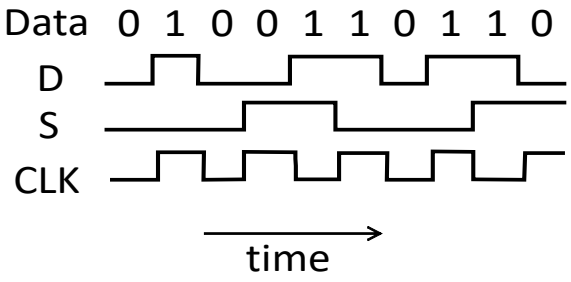


Figure 5-16: Data-Strobe (DS) encoding

b. The Data and Strobe signals shall be set to zero on power up reset.

c. When the SpaceWire output port is reset, it shall be a controlled reset avoiding simultaneous transitions of Data and Strobe signals.

NOTE For example, after stopping transmission the Strobe signal can be reset first, followed by the Data signal. This prevents a simultaneous transition on the data and strobe signals which can cause some IEEE 1355-1995 devices to enter an unrecoverable fault state.

d. When a SpaceWire output port has been transmitting characters and the link state machine enters the ErrorReset state (see clause 5.5.7.2), the data and strobe signals shall be reset with a delay between the reset of the strobe followed by data signal or reset of the data followed by strobe signal.

e. The delay between the reset of the Strobe signal and the Data signal shall be between 500 ns (the period of slowest permitted transmit bit rate, 2 Mbps) and the period of the fastest transmit time for the particular transmitter which is dependent upon implementation.

f. The SpaceWire receiver shall be tolerant of simultaneous transitions on the Data and Strobe lines.

NOTE 1 Being tolerant of simultaneous transitions means that there is no lock-up of the receiver when a simultaneous transition occurs.

NOTE 2 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.

5.4.5 First Null

a. The first bit of the first Null to be sent after the Transmitter Enabled flag is asserted shall be a parity bit, which is set to zero so that the first transition is on the Strobe line.

NOTE This results in the patterns shown in Figure 5-17 appearing when a link is started.

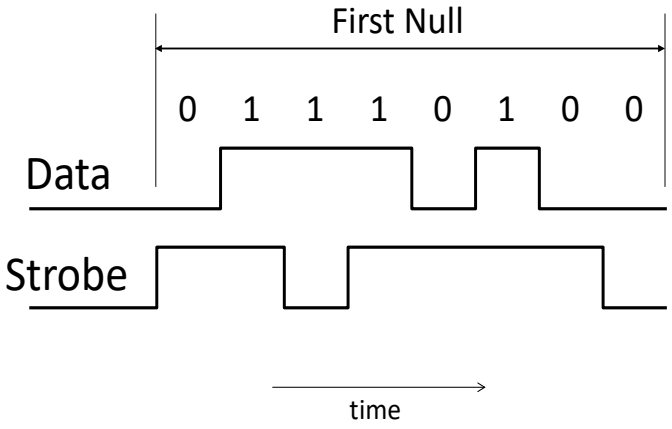


Figure 5-17: Data and strobe signals for first Null

5.4.6 Null detection

a. Null detection shall be enabled whenever the receiver is enabled.

b. The gotNull condition shall only be cleared when the RX Enable flag is de-asserted

c. First Null detection after the receiver is enabled shall include all three parity bits related to the Null.

NOTE The three parity bits related to the Null are the parity bit that covers the data-control flag of the ESC character, the parity bit that covers the ESC character, and the parity bit that covers the FCT character.

d. gotNull shall be asserted when the first Null after the receiver is enabled is detected.

NOTE 1 The first Null after the receiver is enable is the 011101000 sequence of bits as illustrated in Figure 5-18.

NOTE 2 During initialization the character following a Null is a control character (either another Null

or an FCT) so the last parity bit of the Null is zero. See clause 5.4.5 for the format of the first Null transmitted by a SpaceWire interface.

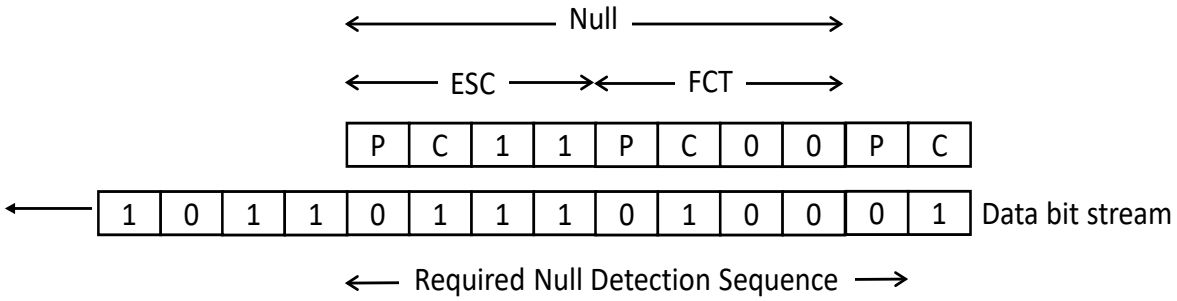


Figure 5-18: Null detection sequence

5.4.7 Parity error

a. Parity detection shall be enabled whenever the receiver is enabled and the gotNull condition is asserted.

b. A parity error shall be detected when the parity is not odd.

NOTE Parity is not odd when there is an even number of bits set to '1', as detailed in clause 5.4.3.4.

5.4.8 Disconnect

a. Disconnects shall be detected by the SpaceWire input port.

b. The disconnect detection shall be enabled when the link state machine leaves the ErrorReset state and the first edge is detected on the Data or Strobe line.

c. A disconnect shall occur when the length of time since the last transition on either the Data or Strobe line is longer than 727 ns (8 cycles of 10 MHz clock + 10 %) and 1 μs maximum (9 cycles of 10 MHz clock - 10 %).

5.4.9 ESC error

a. An escape character (ESC) followed by ESC, EOP or EEP is an invalid sequence and when received shall produce an ESC error.

NOTE ESC followed by FCT is a Null and ESC followed by a data character is a broadcast code.

b. ESC error detection shall be enabled while the gotNull condition is asserted.

5.4.10 Data signalling rate

5.4.10.1 Initial operating data signalling rate

a. After a reset or disconnect (see clause 5.4.8) an output port shall start operating at a data signalling rate of 10 ±1 Mb/s.

b. The SpaceWire output port shall operate at 10 ±1 Mb/s until set to operate at a different data signalling rate.

NOTE 1 The aim is to provide all systems with a common, slow, initial data signalling rate so that system operation can be validated before switching to other and possibly widely different data signalling rates.

NOTE 2 This initial slow data signalling rate is applicable to all SpaceWire output ports, but they need not be capable of higher data signalling rates.

5.4.10.2 Minimum data signalling rate

a. The minimum data signalling rate at which an output port shall operate is 2 Mbps.

NOTE The minimum data signalling rate is the lowest data signalling rate at which a SpaceWire link can operate which is determined by the minimum disconnect timeout (clause 5.4.7) to be greater than 1,37 Mbps, i.e. 1/727 ns.

b. When operation as low as 2 Mbps is not necessary, an output port may have a minimum data signalling rate of more than 2 Mbps but at most 10 Mbps + 10%.

5.4.10.3 Maximum data signalling rate

a. The maximum data signal rate shall be the highest data signalling rate at which a SpaceWire link can operate taking into account signal attenuation, skew and jitter.

b. The maximum data signalling rate shall be specified for each implementation.

c. The maximum signalling data rate shall be the fastest data signalling rate that a SpaceWire link can operate whilst decoding the received data bits correctly.

d. The maximum data signalling rate of a link shall be not less than the initial operating data signalling rate (see clause 5.4.10.1).

5.4.10.4 Operational data signalling rates

a. The output port at one end of a link shall be able to operate at a different data signalling rate to the output port at the other end of the link.

NOTE 1 Once in the Run state it is possible to change the output port data signalling rate from the initial data signalling rate to the intended operating data signalling rate (see clause 5.5.7.7).

NOTE 2 The link is able to run at different speeds in each direction. It is advisable to have both directions of the link running at the same or similar speed because a wide difference in speed restricts the rate of FCTs being returned on the slower direction of the link and thus throttles the data rate in the faster direction.

b. Output ports within a network shall be able to operate at different data signalling rates.

NOTE Overall system performance is improved if all links in a network run at the same or similar speed, since a packet is held up on a fast link if it is elsewhere being routed over a slower link (see clause 5.6.8.7).

5.4.11 Encoding layer management parameters

a. The following management parameters should be provided to configure, control and monitor the encoding layer.

1. Link speed (port), which controls the operational data signalling rate of the specified port.

5.5 Data link layer

5.5.1 Introduction

The Data Link layer specifies how communication is established across a SpaceWire link, how the flow of information is controlled over the link, how N-Chars and broadcast codes are sent and received, and how communications is re-established across the link after an error has occurred.

5.5.2 Data link layer interfaces

a. The data link layer shall receive N-Chars and broadcast codes from the network layer.

NOTE These N-Chars and broadcast codes are encoded by the encoding layer and sent over the SpaceWire link.

b. The data link layer shall pass a sequence of data characters, control characters and control codes to the encoding layer.

NOTE These data characters, control characters and control codes are character encoded, serialised, data-strobe encoded and transmitted by the encoding layer.

c. The data link layer shall receive data characters, control characters and control codes from the encoding layer.

d. The data link layer shall pass N-Chars and broadcast codes to the network layer.

e. The data link layer shall indicate to the Network layer when it is able to accept another N-Char for sending.

f. The data link layer shall indicate to the Network layer when it has an N-Char ready for passing to the Network layer.

g. The data link layer shall indicate to the Network layer when it has a broadcast code ready for passing to the Network layer.

h. The data link layer shall control the operation of the encoding layer using the following control flags:

1. Transmit Enable, which enables the transmitter (character encoder, serialiser, data-strobe encoder and line driver) when asserted and reset it when de-asserted.
2. Receive Enable, which enables the receiver (line receiver, data-strobe decoder, de-serialiser and character decoder) when asserted and reset it when de-asserted.

i. The data link layer shall respond to changes in the encoding layer indicated by the following status flags:

1. Disconnect, which indicates that the link has been disconnected.
2. Receive error, which indicates that an error has been detected in a received symbol.
3. gotNull, which indicates that the first Null character has been received without any parity errors.

NOTE An example of a receive error is a parity error.

5.5.3 Data link layer management interface

a. The data link layer shall be controlled using the following management parameters:

1. PortReset(port), which when asserted resets the specified SpaceWire port.
2. LinkDisabled(port), which when asserted disables the specified SpaceWire port and when de-asserted allows it to operate.
3. LinkStart(port), which when asserted causes the specified SpaceWire port to attempt to start the SpaceWire link by sending Nulls, provided that port is enabled.

4. AutoStart(port), which when asserted causes the specified SpaceWire port to start the SpaceWire link as soon as a Null is received, provided that port is enabled.

NOTE This is the minimum set of management parameters to allow operation of the SpaceWire link.

b. The data link layer should provide the following status information:

1. Link state (port), which is the current state of the data link layer state machine for the specified port;
2. Error flags (port), which is the current state of the following error flags:
   (a) Disconnect;
   (b) Parity Error;
   (c) ESC Error;
   (d) Credit Error;
3. Credit counters (port), which is the values of the transmit and receive credit counters of the specified port.

5.5.4 Flow control

a. The flow of N-Chars across a link shall be controlled using flow control tokens sent from one end of the link (end A) to the other end (end B) to signify that end A is ready to receive some more data.

b. Each flow control token shall be exchanged for eight N-Chars.

c. An FCT shall be sent from the data link layer to the encoding layer when the data link layer is ready to receive a further eight N-Chars.

d. The data link layer shall not send any N-Chars to the encoding layer until it has received one or more FCTs from the encoding layer to indicate that the data link layer at the other end of the link is ready to receive N-Chars.

e. The data link layer shall keep a credit count of the number of N-Chars it has been authorized to send (transmit credit count), as follows:

1. Each time the data link layer receives an FCT from the encoding layer it increments its transmit credit count by eight.
2. Whenever the data link layer sends an N-Char to the encoding layer it decrements its transmit credit count by one.

NOTE EOP and EEP are N-Chars and are credit counted even when empty packets are discarded.

f. If the transmit credit count reaches zero, the data link layer shall cease sending N-Chars to the encoding layer until it receives another FCT from the encoding layer which increases the transmit credit count to eight.

NOTE When the transmit credit count is zero, the data link layer continues to send FCTs, Nulls and broadcast codes to the encoding layer.

g. The transmit credit count shall be set to zero whenever the link state machine enters the *ErrorReset* state.

h. The transmit credit count shall have a maximum value of 56 which corresponds to seven FCTs.

i. A maximum of seven FCTs shall be outstanding at any time.

j. If an FCT is received which causes the transmit credit counter to exceed its maximum value, a credit error shall be raised, see clause 5.5.5.

k. When the link is initialised or re-initialised, one FCT shall be sent for every eight N-Chars that can be held in the receive FIFO up to the maximum of seven FCTs.

l. The data link layer shall keep a credit count of the number of N-Chars it has asked for by passing FCTs to the encoding layer and has yet to receive from the encoding layer (receive credit count) as follows:

   1. Increment the receive credit count by eight each time an FCT is passed to the encoding layer.
   2. Decrement the receive credit count by one each time an N-Char is received from the encoding layer.

m. The receive credit count shall be set to zero whenever the link state machine enters the ErrorReset state.

n. The receive credit count shall have a maximum value of 56, corresponding to seven FCTs.

o. When the data link layer can only hold fewer than 56 received N-Chars, the receive credit count may have a maximum value of less than 56..

   NOTE It is possible for the data link layer to hold more received N-Chars than the maximum receive credit count value.

p. An FCT shall only be sent when there is room in the data link layer to receive another eight more N-Chars from the encoding layer and when the receive credit count has a value of eight or more less than its maximum value.

5.5.5 Flow control errors

a. A credit error shall be detected when either of the following conditions occur:

   1. An N-Char is received when the receive credit counter is zero.
   2. An FCT is received which causes the transmit counter to exceed its maximum permitted value.

      NOTE A credit error indicates that some otherwise undetected error has occurred on the link, which has caused the corruption of the credit count.

5.5.6 Sending priority

a. The order of priority for sending of characters and control codes shall be as follows:

   1. Broadcast codes, highest priority;
   2. FCTs;
   3. N-Chars;
   4. Nulls, lowest priority.

b. When the link state machine is in the Run state and sending of a broadcast code is requested, it shall be sent as soon as the data link layer has finished sending the current character or control code.

c. When sending of an FCT is requested, it shall be sent as soon as the current character or control code has been sent provided that:

   1. No broadcast code is waiting to be sent.

d. When the link state machine is in the Run state and an N-Char is available in the transmit buffer, it shall be sent as soon as the current character or control code has been sent provided that:

   1. No broadcast code is waiting to be sent.
   2. No FCT is waiting to be sent.
   3. The transmit control credit count is above zero.

e. When no broadcast code, FCT or N-Char is ready to be sent, a Null shall be sent to indicate that the link is still active.

   NOTE This prevents the disconnect detection mechanism from being triggered at the far end of the link.

5.5.7 Link initialisation behaviour

5.5.7.1 General

a. The initialisation, establishing the connection and responding to control requests of the SpaceWire link, shall follow the behaviour described in the following clauses.

   NOTE 1 The link initialisation behaviour is illustrated by way of a state diagram for a link state machine in Figure 5-19.

   NOTE 2 Necessary new conditions not specified in the previous issues of the ECSS-E-ST-50-12C standard are: Sent Null and Sent FCT. In addition, the current version (revision 1) permits the link state machine to immediately move to and remain in the ErrorReset state with the receiver disabled, when LinkDisabled is asserted. The functions of LinkDisabled, LinkStart and AutoStart were separated for clarity.

b. When an error occurs on the link, it shall recover from those errors allowing normal link operation to resume when possible.

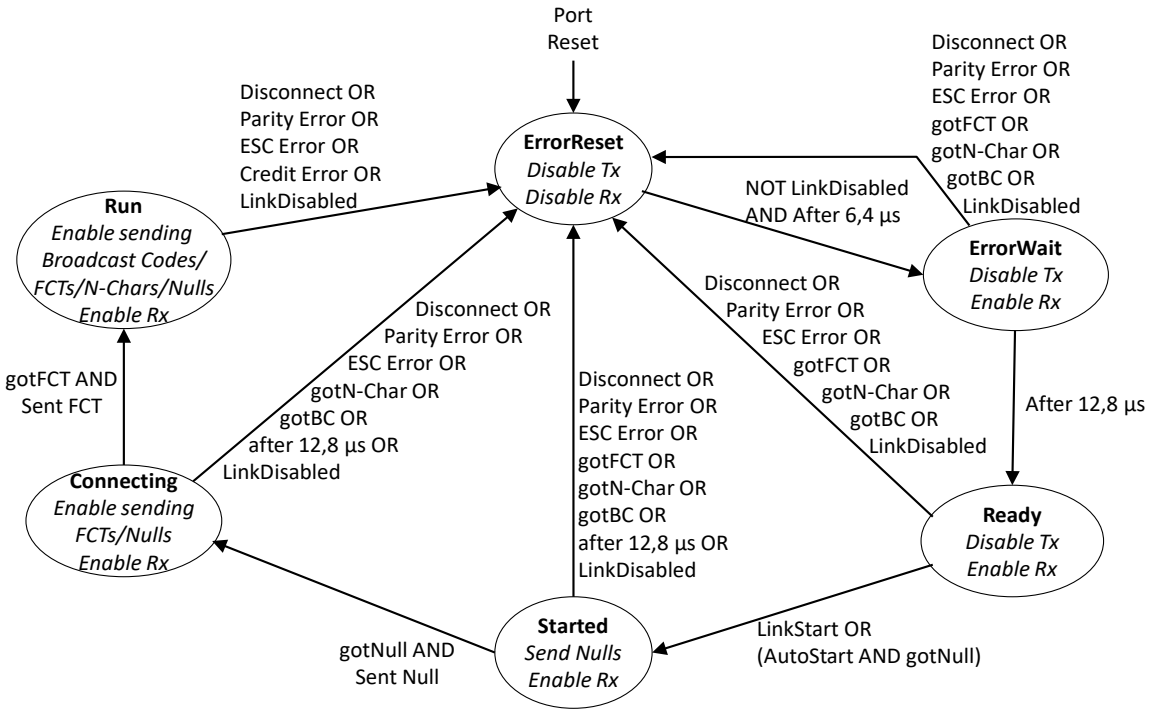


Figure 5-19: Link initialisation behaviour

NOTE 1 Disconnect Error is only enabled after the first transition on the data or strobe line.

NOTE 2 Parity Error, ESC Error, gotFCT, gotN-Char and gotBC are only enabled after the first Null has been received (i.e. gotNull asserted).

NOTE 3 The transition from the Started state to the ErrorReset state has "after 12,8 µs" in black (dark text) which is intentional as this transition is normal operation and not an error condition. It occurs when this end of the line is trying to start and the other end is disabled. The transition from the Connecting state to the ErrorReset state has "after 12,8 µs" in red (lighter text) because it is an error: although a Null has been received indicating that the other end of the link is active, no FCT is received within 12,8 µs so it is a fault.

c. The "after 6,4 µs" condition represents a 6,4 µs (nominal) timing delay which shall have a duration of between 5,82 µs to 7,22 µs.

NOTE 5,82 µs is 64 cycles of a clock with a frequency of 10 MHz + 10% and 7,22 µs is 65 cycles of a clock with a frequency of 10 MHz - 10%.

d. The "after 12,8 μs" condition represents a 12,8 μs (nominal) timing delay which shall have a duration of between 11,64 μs to 14,33 μs.

NOTE 11,64 μs is 28 cycles of a clock with a frequency of 10 MHz + 10% clock and 14,33 μs is 29 cycles of a clock with a frequency of 10 MHz - 10%.

e. When Port Reset is asserted, the following actions shall occur:

1. The TX FIFO and RX FIFO are cleared
2. The link state machine enters the ErrorReset state.

5.5.7.2 ErrorReset state

a. When in the ErrorReset state, the link state machine shall initiate the following actions:

1. De-assert the Transmit Enable control flag.
2. De-assert the Receive Enable control flag.
3. Set the transmit credit counter to zero.
4. Set the receive credit counter to zero.
5. Clear the gotFCT condition.
6. Start a 6,4 μs timer on entering the ErrorReset state.

NOTE De-asserting the Receive Enable control flag causes the Encoding layer to clear the gotNull condition.

b. When in the ErrorReset state the link state machine should initiate the following actions:

1. Discard any broadcast code pending to be sent.

c. The Link state machine shall leave the ErrorReset state on the following condition:

1. When the 6,4 μs timer is elapsed and LinkDisable is de-asserted, move to the ErrorWait state.

5.5.7.3 ErrorWait state

a. When in the ErrorWait state, the Link state machine shall initiate the following actions:

1. Start a 12,8 μs timer on entering the ErrorWait state.
2. De-assert the Transmit Enable control flag.
3. Assert the Receive Enable control flag without storing any N-Chars received from the Encoding Layer in the RX FIFO and without registering any broadcast codes received from the Encoding Layer.

b. The Link state machine shall leave the ErrorWait state on one of the following conditions which are evaluated in the order given:

1. When LinkDisabled is asserted, move to the ErrorReset state.
2. When a disconnect occurs, move to the ErrorReset state.
3. When a parity error occurs, move to the ErrorReset state.

4. When an ESC error occurs, move to the ErrorReset state.
5. When an FCT, N-Char or broadcast code is received from the Encoding Layer, move to the ErrorReset state.
6. When the 12,8 µs timer is elapsed, move to the Ready state.

5.5.7.4 Ready state

a. When in the Ready state, the Link state machine shall initiate the following actions:
   1. De-assert the Transmit Enable control flag.
   2. Assert the Receive Enable control flag without storing any N-Chars received from the Encoding Layer in the RX FIFO and without registering any broadcast codes received from the Encoding Layer.

b. The Link state machine shall leave the Ready state on one of the following conditions:
   1. When LinkDisabled is asserted, move to the ErrorReset state.
   2. When a disconnect occurs, move to the ErrorReset state.
   3. When a parity error occurs, move to the ErrorReset state.
   4. When an ESC error occurs, move to the ErrorReset state.
   5. When an FCT, N-Char or broadcast code is received from the Encoding Layer, move to the ErrorReset state.
   6. When LinkStart is asserted, move to the Started state.
   7. When AutoStart is asserted and gotNull is asserted, move to the Started state.

5.5.7.5 Started state

a. When in the Started state, the Link state machine shall initiate the following actions:
   1. Start a 12,8 µs timer on entering the Started state.
   2. Assert the Transmit Enable control flag, but only pass Nulls to the Encoding Layer.
   3. Assert the Receive Enable control flag without storing any N-Chars received from the Encoding Layer in the RX FIFO and without registering any broadcast codes received from the Encoding Layer.

b. The Link state machine shall leave the Started state on one of the following conditions:
   1. When LinkDisabled is asserted, move to the ErrorReset state.
   2. When a disconnect occurs, move to the ErrorReset state.
   3. When a parity error occurs, move to the ErrorReset state.
   4. When an ESC error, move to the ErrorReset state.
   5. When an FCT, N-Char or broadcast code is received from the Encoding Layer, move to the ErrorReset state.
   6. When enable is asserted, at least one Null has been sent and gotNull is asserted, move to the Connecting state.

7. 12,8 μs after entering the Started state, move to the ErrorReset state.

NOTE The Null that is received causing the gotNull condition to be asserted, can occur in the ErrorWait, Ready or Started state.

5.5.7.6 Connecting state

a. When in the Connecting state, the Link state machine shall initiate the following actions:

1. Start a 12,8 μs timer on entering the Connecting state.
2. Assert the Transmit Enable control flag, but only pass FCTs and Nulls to the Encoding Layer, following the rules described in clause 5.5.6.
3. Assert the Receive Enable control flag without storing any N-Chars received from the Encoding Layer in the RX FIFO and without registering any broadcast codes received from the Encoding Layer.

NOTE If the specific state machine implementation introduces some delay during the transition to the Run state, it is necessary to store any N-Chars received from the Encoding Layer in the RX FIFO and register any broadcast codes received from the Encoding Layer after gotFCT is asserted. This is due to the fact that the far end of the link can already be in the Run state, transmitting data at a much higher data rate.

b. The Link state machine shall leave the Connecting state on one of the following conditions:

1. When LinkDisabled is asserted, move to the ErrorReset state.
2. When a disconnect occurs, move to the ErrorReset state.
3. When a parity error occurs, move to the ErrorReset state.
4. When an ESC error occurs, move to the ErrorReset state.
5. When enable is asserted, at least one FCT has been sent and an FCT has been received (gotFCT asserted), move to the Run state.
6. When an N-Char or broadcast code is received from the Encoding Layer, move to the ErrorReset state.
7. 12,8 μs after entering the Connecting state, move to the ErrorReset state.

5.5.7.7 Run state

a. When in the Run state, the Link state machine shall initiate the following actions:

1. Assert the Transmit Enable control flag, and pass broadcast codes, FCTs, N-Chars and Nulls to the Encoding Layer; following the rules described in clause 5.5.6.
2. Assert the Receive Enable control flag.

3. Pass received N-Chars and broadcast codes to the Encoding Layer.
4. Pass any received broadcast codes to the Network Layer.
5. If and when required, change the SpaceWire output port operating rate (see clause 5.4.10).

b. The Link state machine shall leave the Run state on one of the following conditions:
   1. When LinkDisabled is asserted, move to the ErrorReset state.
   2. When a disconnect occurs, move to the ErrorReset state.
   3. When a parity error occurs, move to the ErrorReset state.
   4. When an ESC error occurs, move to the ErrorReset state.
   5. When a credit error occurs, move to the ErrorReset state.

5.5.7.8 Alternative behaviour when disabled asserted

a. When a SpaceWire device is being used which was developed prior to this current revision of the SpaceWire standard, alternative behaviour of the link initialisation state machine may be provided as follows:
   1. LinkDisabled is only recognised in the Ready and Run state.
   2. When LinkDisabled is asserted in the Ready state, the link state machine remains in the Ready state.
   3. When LinkDisabled is asserted in the Run state, the link state machine moves to the ErrorReset state.

      NOTE This alternative behaviour allows the link state machine of the previous version of the SpaceWire standard ECSS-E-ST-50-12C (31 July 2008) to be used. The improvement in the current version (Revision 1) means that the link state machine immediately moves to and remains in the ErrorReset state with the receiver disabled, when Disable is asserted. The functions of Disable/Enable, Link Start and Autostart have been separated for clarity.

5.5.8 Link error recovery

5.5.8.1 General

a. The recovery of the link from errors shall follow the behaviour described in the following clauses.

   NOTE The link error recovery behaviour is illustrated by way of a state diagram for a link error recovery state machine in Figure 5-20.

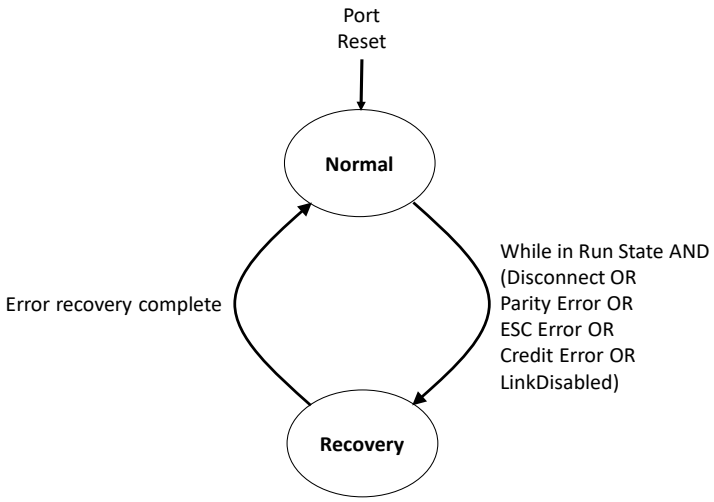


Figure 5-20: Link error recovery behaviour

5.5.8.2 Port Reset

a. When Port Reset is asserted, the Link Error Recovery state machine shall enter the Normal state.

5.5.8.3 Normal state

a. When in the Normal state, the Link Error Recovery state machine shall not take any action.

b. The Link Error Recovery state machine shall leave the Normal state on the following conditions:

   1. When the Link state machine is in the Run state and LinkDisabled is asserted, the Link Error Recovery state machine moves to the Recovery state.
   2. When the Link state machine is in the Run state and a Disconnect, Parity Error, ESC Error, or Credit Error occurs, the Link Error Recovery state machine moves to the Recovery state.

5.5.8.4 Recovery state

a. When in the Recovery state, the Link Error Recovery state machine shall initiate the following actions:

   1. If currently sending a packet, discard the remainder of the packet that has not yet been sent.
   2. If the last character written to the receive FIFO was a data character, write an EEP to the receive FIFO.
   3. When the last character written was an EOP or EEP, another EEP can be added to the receive FIFO.
   4. If an EEP is pending, ready for writing to the receive FIFO and that FIFO is full preventing an EEP being written, wait until there is space in the receive FIFO and then write the EEP.
   5. Record the cause of the error in a status register.

NOTE 1 I.e. if one or more N-Chars have been sent since the link reached the Run state AND the last character sent was not an EOP or EEP, read the transmit FIFO and discard the characters read until an EOP or EEP has been read and discarded.

NOTE 2 Because the remainder of a packet is discarded after an error, very large packets can cause the link to halt for a long period of time while the packet is spilt. The link is initialized but not be able to send a packet until the remainder of the packet being spilt has been discarded. It is important to bear this in mind when determining the size of packets to use in a particular application.

NOTE 3 If the receive FIFO is full it is not possible for the transmitter to send an FCT, hence the link initialisation cannot be completed. The link state machine cycles through the ErrorReset, ErrorWait, Ready and Started state and then times out in the Connecting state because no FCT can be received. When there is room for at least eight N-Chars, at least one FCT can be sent so the link initialisation process is then able to complete successfully.

b. The Link Error Recovery state machine shall leave the Recovery state on the following conditions:

1. When all error recovery actions, listed in 5.5.8.4.b are successfully completed, move to the Normal state, move to the Normal state.

5.5.9 Accepting broadcast codes for sending

a. Broadcast codes passed to the data link layer by the network layer should be discarded unless the link state machine is in the Run state.

NOTE This prevents a broadcast code from being passed to the data link layer and sent some significant time later when the link is started.

5.6 SpaceWire network layer

5.6.1 Introduction

The Network layer specifies how SpaceWire packets, time-codes and distributed interrupts are transferred over a SpaceWire Network.

5.6.2 SpaceWire packets

5.6.2.1 SpaceWire packet

a. A SpaceWire packet shall comprise one or more data characters followed by an end of packet marker (EOP) or error end of packet marker (EEP).

b. The start of a packet shall be either:

1. The first data character sent after a link is initialised or re-initialised following a disconnect.
2. The data character that immediately follows an EOP or EEP.

NOTE The first data character after the end of the previous packet is the start of the next packet.

c. The end of a packet shall be indicated by an EOP or EEP.

d. The packet shall contain zero or more data characters.

NOTE If a packet contains zero data characters, the packet is discarded by the first routing switch that it encounters.

e. Zero or more data characters at the front of a packet shall form a destination address.

f. The remaining data characters, following the destination address and up to the EOP or EEP, shall form the cargo.

NOTE The format of a SpaceWire packet is illustrated in Figure 5-21.

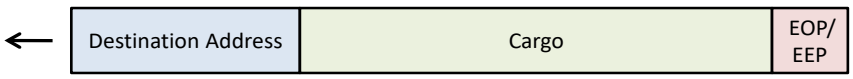


Figure 5-21: SpaceWire packet format

5.6.2.2 N-Char interleaving

a. N-Chars from one packet shall not be interleaved with N-Chars from another packet.

NOTE N-Chars can be interleaved with FCTs, Nulls and broadcast codes as explained in clause 5.5.6.

5.6.3 Broadcast codes

a. A time-code shall be a broadcast code with the broadcast code type set to 0b00.

b. A distributed interrupt code shall be a broadcast code with the broadcast code type set to 0b10.

NOTE The specialisations of broadcast codes are illustrated in the UML diagram of Figure 5-22.

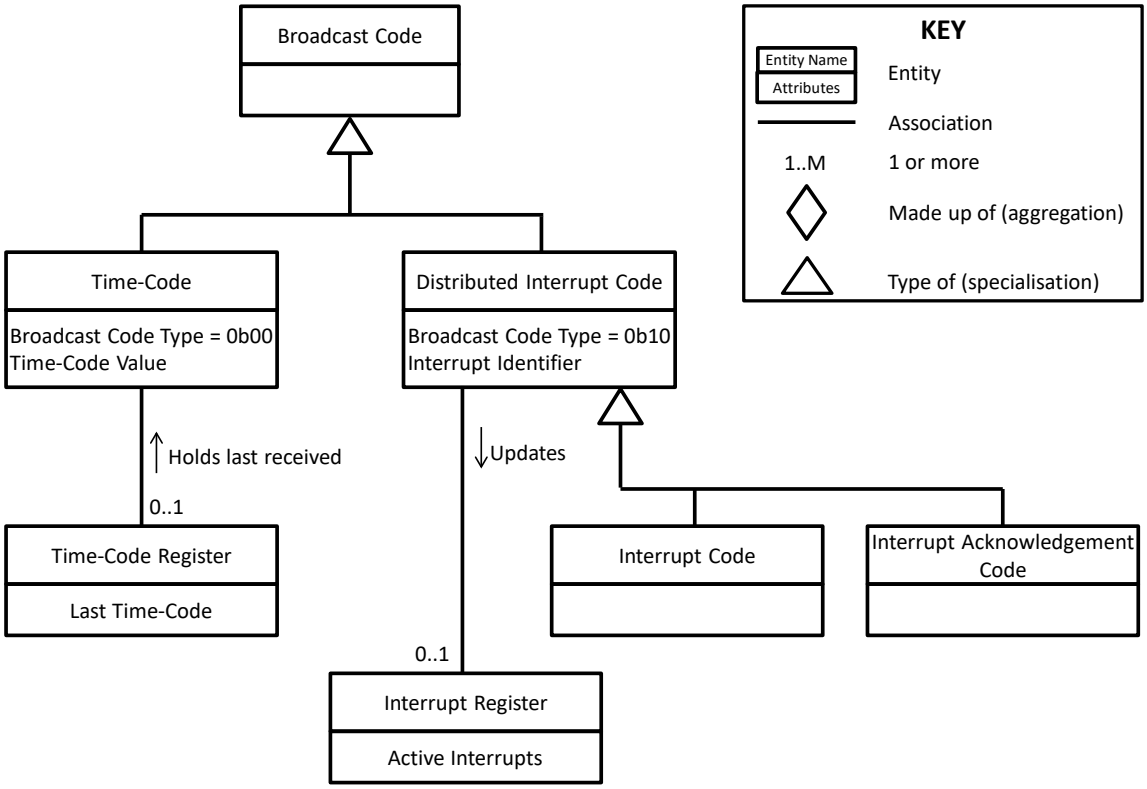


Figure 5-22: Specialisations and relationships of a SpaceWire broadcast code

c. There shall be two types of distributed interrupt codes: interrupt code, and interrupt acknowledgment code.

d. The order of priority with which the network layer passes different types of broadcast codes to the data link layer shall be as follows:

1. Time-code, highest priority;
2. Interrupt acknowledgement code;
3. Interrupt code, lowest priority.

e. A broadcast code that has a broadcast code type set to 0b01 or 0b11 shall be deleted and not forwarded by routing switches.

f. A broadcast code that has a broadcast code type set to 0b01 or 0b11 shall be discarded when it is received by a node.

5.6.4 SpaceWire time-codes

5.6.4.1 General

a. SpaceWire time-codes shall be optionally implemented in nodes and routers.

b. When time-codes are not supported by a node or router, that node or routing switch shall ignore any time-codes it receives.

5.6.4.2 Time-codes

a. The value field of a time-code shall contain a 6-bit time-code value, which is an unsigned binary integer.

NOTE The network layer time-code is illustrated in Figure 5-23.

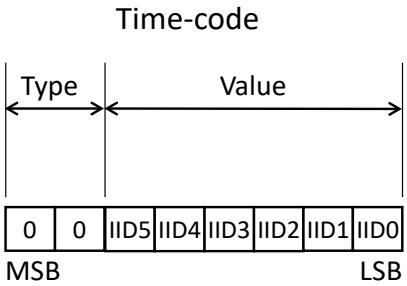


Figure 5-23 Network layer time-code

5.6.4.3 Time-code register

a. A node supporting time-codes shall have one or more time-code registers each of which is associated with one or more end-points.

b. Each time-code register in a node shall hold the value of the last time-code transmitted or received over the end-points it is associated with.

NOTE See clause 5.6.4.5 for an explanation of a valid time-code.

c. A routing switch supporting time-codes shall have one time-code register, which is used to hold the value of the last time-code received.

d. When port reset is asserted, the time-code register shall be set to have a time-code value of zero.

5.6.4.4 Time-code master

a. A network supporting time-code broadcast shall have a single time-code master application on a single node which is responsible for sending time-codes over the network.

b. The time-code master shall pass time-codes to one or more end-points for sending across the network.

c. When more than one end-point is being used by a time-code master to send time-codes across the network, the following actions shall occur:

1. Those end-points are all associated with the same time-code register;
2. The time-code master loads the time-code register with the time-code value that is for sending;

3. The time-code master requests each of the end-points to send a time-code with the value held in the time-code register at the same time.

NOTE 1 Sending the same time-code over two or more end-points is intended to support redundant systems. If one time-code is lost, another time-code being broadcast through a redundant rout through the network takes the place of the lost time-code.

NOTE 2 The host system is able to send time-codes without sequential values, but only those with sequential values propagate across the network. Non-sequential time-codes can be used to reinitialise the time-codes on the network to a new value or be used to send specific values over a point-to-point link.

5.6.4.5 Valid time-code

a. When a time-code is received by an end-point or routing switch that has a time-code register, its value shall be compared to the value contained in the time-code register.

b. If the received time-code has a value which is one more, modulo 64, than the value in the time-code register, the received time-code shall be valid.

c. If the received time-code has a value which is not one more, modulo 64, than the value in the time-code register, the received time-code shall be invalid.

5.6.4.6 Valid time-code in routing switch

a. When a valid time-code is received by a routing switch with a time-code register, it shall:

1. Request to send time-codes with the same value as the received valid time-code out through each of its output ports, except for the output port on which the valid time-code arrived.

2. Update the time-code register in the routing switch to the value of the valid time-code.

5.6.4.7 Invalid time-code in routing switch

a. When an invalid time-code is received by a routing switch with a time-code register, it shall update the time-code register in the routing switch to the value of the received time-code.

5.6.4.8 Valid time-code in a node

a. When a valid time-code is received by an end-point associated with a time-code register, it shall:

1. Indicate to the host system that a valid time-code has arrived and make the value of the time-code available to the host system.

2. Update the time-code register associated with the end-point to the value of the valid time-code.

5.6.4.9 Invalid time-code in a node

a. When an invalid time-code is received by an end-point with a time-code register, it shall update the time-code register associated with the end-point to the value of the received time-code.

NOTE This is so that when a time-code is lost for any reason the time-code broadcast mechanism can recover and continue to broadcast time-codes across the network after a few more time-codes have been sent. The number of time-codes that are sent following a missing time-code to recover time-code distribution depends on the size of the network. If there is active redundancy in the network the missing time-code is likely to be recovered by that same time-code being received via another path through the network.

5.6.5 SpaceWire distributed interrupts

5.6.5.1 General

a. SpaceWire distributed interrupts shall be optionally implemented in nodes and routers.

b. When distributed interrupts are not supported by a node or router, that node or routing switch should ignore any interrupt codes and interrupt acknowledgement codes it receives, as well as any host requests to send interrupts and interrupt acknowledgements.

NOTE New routing switch devices without support for distributed interrupts are recommended to ignore interrupt codes and interrupt acknowledgement codes. Nodes without support for distributed interrupts can be blocked from receiving them by a router.

c. There shall be two modes that an interrupt can function in:

1. Interrupt mode, where only interrupt codes are sent across the network;
2. Interrupt with acknowledgement mode, where interrupt codes and interrupt acknowledgement codes are sent across the network.

d. A node or routing switch that supports distributed interrupts shall operate in either interrupt mode or interrupt with acknowledgement mode or be configurable to operate in one mode or the other.

NOTE It is possible for a network to support both interrupt mode and interrupt with

acknowledgement mode provided that a particular interrupt is used in the same mode across the whole network.

e. When operating in the interrupt with acknowledgement mode, there shall be only one interrupt source and only one interrupt acknowledger for a specific interrupt.

NOTE In interrupt mode there can be more than one interrupt source for a specific interrupt.

5.6.5.2 Interrupt codes

a. An interrupt code shall be used to broadcast an interrupt across the network.

NOTE The network layer interrupt code is illustrated in Figure 5-24.

b. Bits 0 to 4 of the value field of an interrupt code shall contain an interrupt identifier, which has a value between 0 and 31.

c. Bit 5 of the value field of an interrupt code shall be 0.

d. An interrupt code shall be associated with one of 32 possible interrupts as determined by the value of the interrupt identifier in the interrupt code.

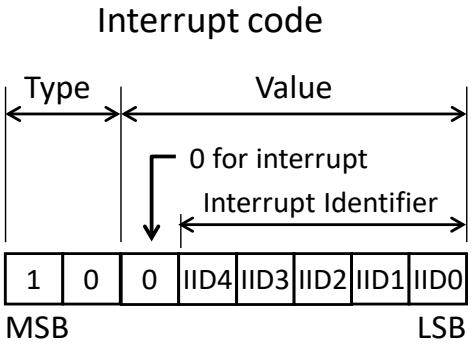


Figure 5-24: Network layer interrupt

5.6.5.3 Interrupt acknowledgement codes

a. An interrupt acknowledgement code shall be used to send an acknowledgement of an interrupt back to the source of the interrupt.

b. Bits 0 to 4 of the value field of an interrupt acknowledgement code shall contain an interrupt identifier, which has a value between 0 and 31.

c. Bit 5 of the value field of an interrupt acknowledgement code shall be 1.

d. An interrupt acknowledgement code corresponding to an interrupt code with an interrupt identifier of value N ($0 \leq N \leq 31$) shall have an interrupt identifier of value N.

NOTE The network layer interrupt acknowledgement code is illustrated in Figure 5-25.

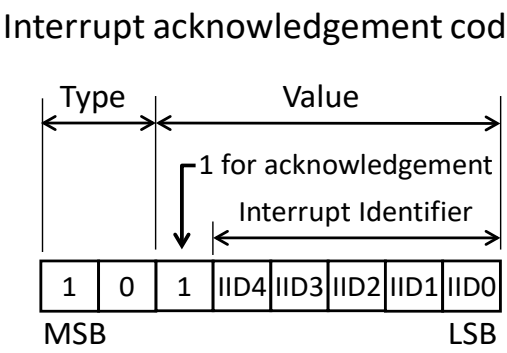


Figure 5-25: Network layer interrupt acknowledgement code

5.6.5.4 Interrupts in a node

a. A node supporting distributed interrupts shall support either sending of interrupt codes, or receiving of interrupt codes, or both sending and receiving of interrupt codes.

b. When a node supports sending of interrupt codes and the host requests an interrupt to be sent, an interrupt code with the interrupt identifier matching the interrupt shall be passed to the data link layer for sending.

c. When operating in interrupt mode, there should be a minimum interval between a node sending one interrupt code with a particular value and sending the next interrupt with that same value which is greater than the maximum propagation time of an interrupt code across the network.

NOTE 1 If an interrupt is used in interrupt mode and the host system requests to send an interrupt too fast after the previous identical interrupt was sent, i.e. before the corresponding interrupt time-out timer in the routing switches has expired, the new interrupt code that the node sends is discarded by a router.

NOTE 2 If an interrupt is sent before the minimum interval, that interrupt can be discarded by a routing switch in the network. Since the same interrupt has already been sent, this is not necessarily a problem at the system level.

d. When operating in interrupt acknowledgement mode, there shall be a minimum interval between a node sending one interrupt code with a particular value and sending the next interrupt with that same value which is greater than the maximum propagation time of a interrupt code across the network and the maximum time for the interrupt acknowledgement code to be generated and return across the network.

NOTE If an interrupt is used in interrupt with acknowledgement mode and the host system requests to send an interrupt too fast after the previous identical interrupt was sent, i.e. either before a corresponding interrupt acknowledgement for the previous interrupt

was received, or before the received interrupt acknowledgement code has propagated across the entire network, the result is indeterminate for that specific interrupt. The new interrupt code that the node sends is usually discarded by a switch but can propagate continuously around the network if the network has circular connections.

e. When a node supports receiving of interrupt codes and an interrupt code is passed to the network layer from the data link layer, the host shall be informed that an interrupt matching the interrupt identifier specified in the interrupt code has arrived.

5.6.5.5 Relaying interrupts within a routing switch

a. Each port of a routing switch that supports distributed interrupts shall be configurable to enable or disable the receiving of interrupt codes in through the input port, and the sending of interrupt codes out of the output port.

b. When a port of a routing switch is disabled from receiving interrupt codes, it shall discard any interrupt codes received on that port.

c. When a port of a routing switch is disabled from sending interrupt codes, it shall not send any interrupt codes on that port.

d. A routing switch that supports distributed interrupts shall contain one 32-bit interrupt relay register that has one bit for each possible interrupt identifier, where bit N of the register relates to interrupt identifier N.

e. When fewer than 32 interrupts are being used, the interrupt relay register may be less than 32 bits.

NOTE If a routing switch constrains the range of interrupt codes that it is prepared to relay it has system-wide implications, possibly relating to the re-use of such a constrained routing switch across missions.

f. On reset the interrupt relay register shall be cleared to zero.

g. Each bit in the interrupt relay register shall represent the state of a particular distributed interrupt, with bit N corresponding to interrupt identifier N.

h. Each bit of the interrupt relay register shall have its own interrupt time-out timer.

i. The interrupt time-out timers shall have a configurable time-out interval.

j. When operating in the interrupt mode, the interrupt time-out interval shall be configured to a value, T_R, which is higher than the worst-case propagation time of the interrupt code in the network.

NOTE 1 If T_R is configured to a value that is too low, the distributed interrupts fail to work properly.

NOTE 2 It is sufficient for one value of T_R to be used by each interrupt time-out timer in each routing switch in a network.

k. When operating in the interrupt acknowledgement mode, the interrupt time-out interval shall be configured to a value, T_{RA}, which is higher than the worst-case propagation time of the interrupt code across the network plus the maximum time it takes for a node to generate an interrupt acknowledgement code plus the time it takes for that interrupt acknowledgement code to propagate across the network.

NOTE If T_{RA} is configured to a value that is too low, the routing switch resets the corresponding bit of the interrupt relay register before the interrupt acknowledgement propagates across the network and the acknowledgement is lost.

l. When a port of a routing switch is configured to enable the receiving of interrupt codes and an interrupt code is received on that port, the following actions shall be performed:

1. The bit in the interrupt relay register specified by the interrupt identifier value of the received interrupt code is checked.
2. If the interrupt relay register is less than 32 bits and the specified bit is non-existent, the received interrupt code is discarded.
3. If the specified bit in the interrupt relay register is 0:
   (a) The specified bit is set to 1.
   (b) The received interrupt code is passed to all ports of the routing switch for onward transmission, except for the port on which the received interrupt code arrived and the ports that are disabled from sending interrupt codes.
   (c) The interrupt time-out timer associated with the specified bit in the interrupt relay register is started.
4. If the specified bit in the interrupt relay register is 1, the received interrupt code is discarded.

NOTE This prevents repeated propagation of interrupt codes in networks with circular connections.

m. When an interrupt time-out timer expires, the corresponding bit in the interrupt relay register shall be reset to 0.

5.6.5.6 Interrupt acknowledgements in a node

a. A node supporting distributed interrupts in the interrupt with acknowledgement mode shall conform to clause 5.6.5.4.

b. A node supporting distributed interrupts in the interrupt with acknowledgement mode shall support either sending of interrupt acknowledgement codes, or receiving of interrupt acknowledgement codes, or both sending and receiving of interrupt acknowledgement codes.

c. When a node supports sending of interrupt acknowledgement codes and the host requests an interrupt acknowledgement to be sent, an interrupt acknowledgement code with the specified interrupt identifier shall be passed to the data link layer for sending.

d. A node not supporting sending of interrupt acknowledgements should discard any host requests to send interrupt acknowledgements.

e. There shall be a delay between the interrupt code arriving and the interrupt acknowledgement being generated, which is greater than the propagation time of the interrupt code across the network.

NOTE If the host system tries to send an interrupt acknowledgement too soon after a corresponding interrupt code has been received, i.e. before the interrupt code has propagated across the entire network, the result is indeterminate for that specific interrupt. The new interrupt acknowledgement code that the node sends can either be discarded by a router, or repeatedly propagated through the network if the network has circular connections.

f. The delay between the interrupt code arriving and the interrupt acknowledgement being generated shall be less than the maximum time determined for a node to generate an interrupt acknowledgement code (see clause 5.6.5.5.k).

NOTE If the host system is too slow in sending an interrupt acknowledgement after a corresponding interrupt code has been received, i.e. the interrupt acknowledgement code is sent after the corresponding interrupt time-out timer in the routing switches has expired, the interrupt acknowledgement code is discarded by the first router.

g. When a node supports receiving of interrupt acknowledgement codes and an interrupt acknowledgement code is passed to the network layer from the data link layer, the host shall be informed that an interrupt acknowledgement matching the interrupt identifier in the interrupt acknowledgement code has arrived.

h. A node not supporting receipt of interrupt acknowledgements should discard any received interrupt acknowledgement codes.

5.6.5.7 Relaying interrupt acknowledgements within a routing switch

a. A routing switch not supporting distributed interrupts in the interrupt with acknowledgement mode should discard any received interrupt acknowledgement codes.

NOTE A routing switch supporting distributed interrupts in the interrupt with acknowledgement mode operates in a similar

way to a routing switch supporting the interrupt mode described in clause 5.6.5.5.

b. Each port of a routing switch that supports distributed interrupts in the interrupt with acknowledgement mode shall be configurable to enable or disable the receiving of interrupt acknowledgement codes in through an input port and the sending of an interrupt acknowledgement code out of an output port.

c. When a port of a routing switch is disabled from receiving interrupt acknowledgement codes, it shall discard any interrupt acknowledgement codes received on that port.

d. When a port of a routing switch is disabled from sending interrupt acknowledgement codes, it shall not send any interrupt acknowledgement codes on that port.

e. In the interrupt with acknowledgement mode, the interrupt time-out timers shall have a configurable time-out value, T_{RA}.

NOTE One common value of T_{RA} for all routers in a network is sufficient.

f. When operating in the interrupt acknowledgement mode, T_{RA} shall be configured to a value which is greater than the worst-case propagation time of the interrupt code in the network, plus the maximum delay of the interrupt handler, plus the worst-case propagation time of the interrupt acknowledgement code in the network.

NOTE If T_{RA} is configured to a value that is too low, the distributed interrupts fails to work properly.

g. When a port of a routing switch is configured to enable the receiving of interrupt acknowledgement codes and an interrupt acknowledgement code is received on that port, the bit in the interrupt relay register specified by the interrupt identifier value of the received interrupt acknowledgement code shall be checked and the following actions taken:

1. If the interrupt relay register is less than 32 bits and the specified bit is non-existent, the received interrupt acknowledgement code is discarded
2. If the specified bit in the interrupt relay register is 1:
   (a) The specified bit is reset to 0.
   (b) The received interrupt acknowledgement code is passed to all ports of the routing switch for onward transmission, except for the port on which it arrived and those ports that are disabled from sending interrupt acknowledgement codes.
   (c) The interrupt time-out timer associated with the specified bit in the interrupt relay register is stopped and set to its time-out value.
3. If the specified bit in the interrupt relay register is 0, the received interrupt acknowledgement code is ignored.

NOTE This prevents repeated propagation of the interrupt acknowledgement codes in networks with circular connections.

h. As specified in 5.6.5.5, when an interrupt time-out timer expires, the corresponding bit in the interrupt relay register shall be reset to 0.

NOTE This recovers from the situation when an interrupt acknowledgement is lost for some reason.

5.6.6 SpaceWire nodes

a. A SpaceWire node shall act as:

1. A source of packets and broadcast codes sent over a SpaceWire network,
2. A destination for packets and broadcast codes received over a SpaceWire network, or
3. Both a source and destination of packets and broadcast codes.

b. A SpaceWire node shall comprise:

1. One or more end-points each of which provides an interface between one port and the host system, and
2. Zero or more time-code registers.

NOTE The UML diagram in Figure 5-26 illustrates the components and specialisations of a SpaceWire node.

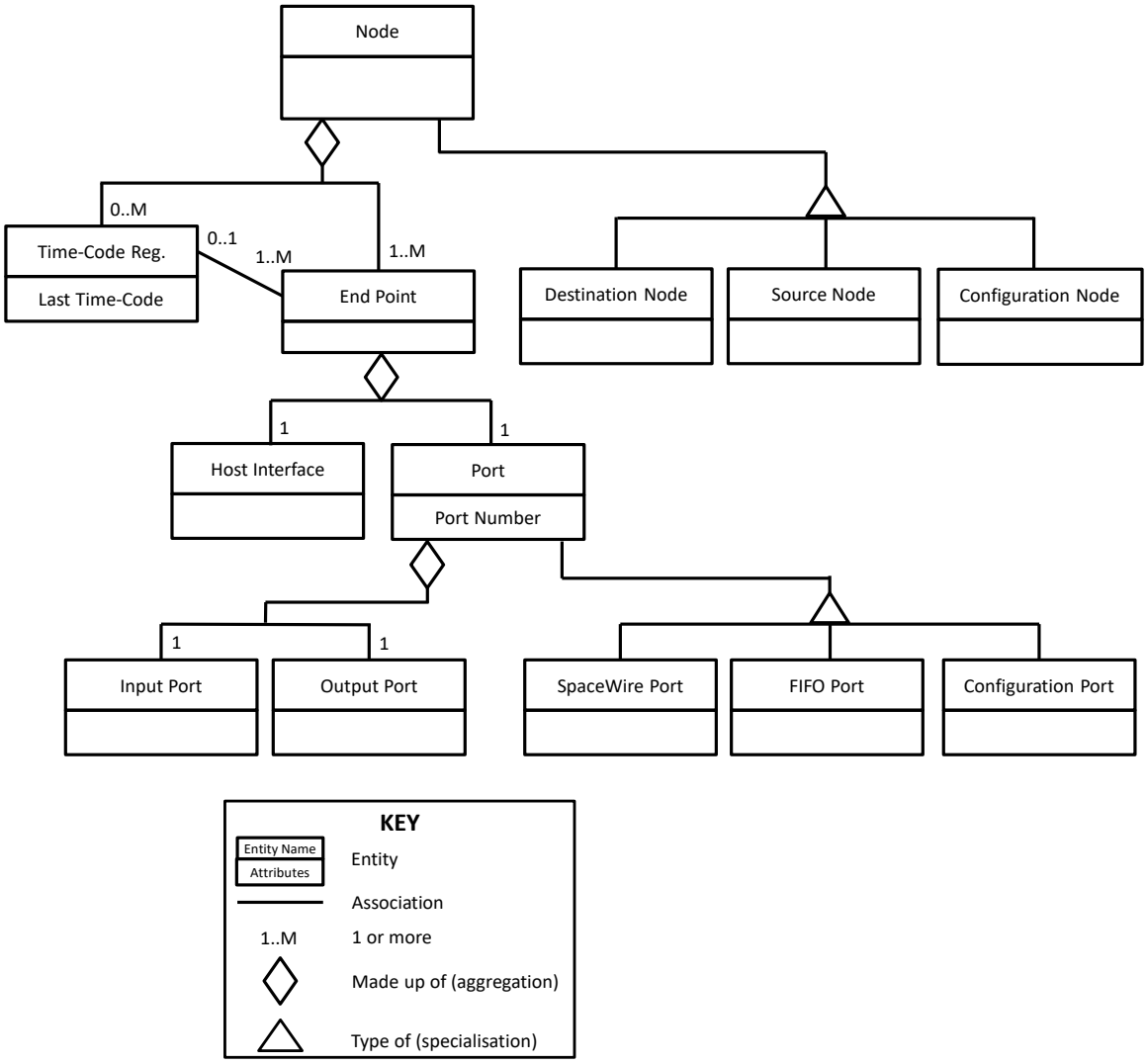


Figure 5-26: Components and specialisations of a SpaceWire node

c. An end-point shall comprise:

   1. One host interface.
   2. One port, which can be a SpaceWire port, FIFO port or configuration port.

d. An end-point shall accept packets from the host system for sending through the port and across the SpaceWire network it is attached to.

e. An end-point shall pass packets received from the SpaceWire network through its port to the host system.

f. An end-point that is acting as a time-code master shall accept time-codes from the host system for broadcasting over the SpaceWire network via the port.

   NOTE When two or more end-points in a time-code master node are associated with the same time-code register, they all broadcast a time-code at the same time. This is useful for networks with redundant paths. Time-codes are sent over each

of the redundant paths to ensure robustness against one time-code being lost due to an error on a link.

g. An end-point that supports time-codes shall pass time-codes, received over the SpaceWire network via the port, to the host system.

h. Each end-point shall optionally be associated with a time-code register, which is used to validate time-codes when they are received by the end-point.

i. An end-point that supports distributed interrupts shall pass distributed interrupts, received over the SpaceWire network via the port, to the host system.

j. An end-point that supports distributed interrupts shall accept interrupt codes and interrupt acknowledgment codes from the host system for broadcasting over the SpaceWire network via the port.

k. When a node has a configuration port, it shall be connected via a host interface to a configuration application on the host system which is responsible for configuring the node.

5.6.7 SpaceWire node management parameters

a. A SpaceWire node shall provide the management parameters for the port in each of its end-points as detailed in the following clauses:

1. Clause 5.5.3 data link layer management parameters
2. Clause 5.4.9 encoding layer management parameters
3. Clause 5.3.8 physical layer management parameters

5.6.8 SpaceWire routing

5.6.8.1 Routing switch

a. A SpaceWire routing switch shall forward a packet, arriving at one port, towards its required destination, through another port or back through the same port as determined by the first data character of the packet and the contents of the routing table.

NOTE A packet can be forwarded through two or more ports when packet multicast is being used, see clause 5.6.8.10.

b. A SpaceWire routing switch shall comprise:

1. One or more ports, which include SpaceWire ports that interface to the SpaceWire network and FIFO ports that provide a parallel interface into the routing switch.
2. A switch matrix which connects an input port to an output port.
3. A routing table which together with the leading byte of a packet (the destination address) determines which port a packet is forwarded through.

4. A configuration node which is accessible through the routing switch using port address 0 and which is used to configure the routing switch including the routing table and parameters for each SpaceWire port.
5. Zero or one time-code register which holds the value of the last time-code received.
6. Zero or one interrupt relay register which indicates which interrupts are currently active.

NOTE 1 A currently active interrupt is one that is being sent across the network.

NOTE 2 A FIFO port is the means by which parallel interfaces of various forms can be connected into the router, including large FIFO interfaces, DMA and parallel buses. The term FIFO port is used because it contains a small FIFO whatever the nature of the parallel interface.

NOTE 3 The UML diagram in Figure 5-27 illustrates the components of a SpaceWire routing switch.

c. A SpaceWire routing switch that has a time-code register shall broadcast valid time codes as detailed in clause 5.6.3.

d. A SpaceWire routing switch that has an interrupt relay register shall broadcast distributed interrupt codes as detailed in clause 5.6.5.

e. A configuration node in a routing switch shall be accessed via a configuration port connected to the switch matrix.

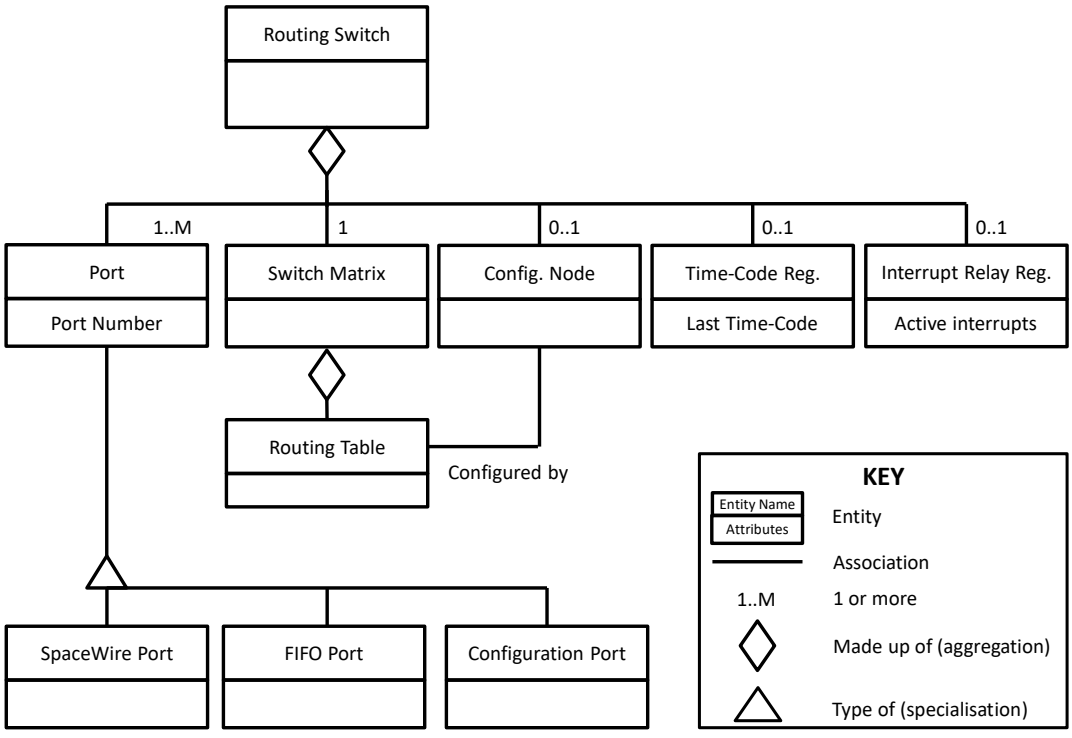


Figure 5-27: Components of a SpaceWire routing switch

5.6.8.2 Port addressing

a. A configuration port shall be assigned port address 0.

b. SpaceWire and FIFO ports shall be assigned port addresses 1-31, so that there is a maximum of 31 external ports on a routing switch.

c. A routing switch shall use the leading data character of a packet to determine the output port of the routing switch that the packet is forwarded through.

5.6.8.3 Path addressing

a. A leading data character in the range 0 to 31 shall be all or part of a path address which is used to determine the route the packet takes through the network.

NOTE Table 5-11 summarises the function of each address.

b. When the leading data character of a packet has a value of 0, the packet shall be routed to the routing switch configuration port.

c. When the leading data character of a packet has a value between 1 and 31, the packet shall be routed to the output port with the corresponding number.

NOTE For example, a leading data character with a value of 6 results in the packet being routed to port 6.

d. A path address shall comprise one or more data characters at the front of a packet; the first data character specifying the output port of the first routing switch encountered by the packet, the second data character the output port of the second routing switch, and so on for all the routing switches on the path through the network from source to destination.

e. When the leading data character is a path address, that data character shall be discarded once is has been used by the routing switch, thus exposing the next data character as the leading data character of the packet ready for use by the next routing switch encountered by the packet.

NOTE A leading data character is a path address when it has a value in the range 0 to 31.

f. A routing switch configuration port shall be accessed using path addressing only.

5.6.8.4 Logical addressing

a. A leading data character in the range 32 to 255 shall be a logical address which is used to identify the destination of the packet and in conjunction with the routing table in the routing switch determine which port of the routing switch the packet is forwarded through.

NOTE Table 5-11 summarises the function of each address.

b. The routing table in the routing switch shall hold the logical address to physical port mapping and determine whether leading data character deletion is applied.

c. When the leading data character of a packet has a value of between 32 and 255, that packet shall be routed to the output port that is referred to in the corresponding location of the routing table within the routing switch.

NOTE For example, a leading data character with a value of 49 results in entry 49 of the routing table specifying the output port that the packet is forwarded through. If entry 49 contains the value 7, the packet is forwarded through port 7.

d. The logical address 255 is reserved and shall not be used.

e. A logical address shall comprise a single data character at the front of a packet which is used by all routing switches on the path from source to destination to look up the required output port numbers in their routing tables.

NOTE If logical address deletion is being used (see clause 5.6.8.6) it is possible to have more than one logical address at the start of a packet.

f. When the leading data character is a logical address, that data character shall be retained as the leading data character of the packet once it has been used by the routing switch, so that the same logical address is used by all routing switches encountered by the packet, unless logical address deletion is being used (see clause 5.6.8.6).

NOTE A leading data character is a logical address when has a value is in the range 32 to 255.

Table 5-11: Address function

Address range	Function	Leading data character deletion
0 0x00	Direct access to internal configuration port.	Always deleted.
1-31 0x01-0x1F	Direct access to physical output ports.	Always deleted.
32-254 0x20-0xFE	Logical address which is mapped to a physical output port via the routing table.	A logical address is not normally deleted. See clause 5.6.8.6 for cases when a logical address is deleted.
255 0xFF	Reserved logical address, which is treated in the same way as any other logical address.	Treated in the same way as a logical address.

5.6.8.5 Addressing errors

a. If a port with a specific number is non-existent in the routing switch, a packet arriving with a leading data character that references that non-existent port shall be discarded.

b. A packet arriving with a leading data character that references a non-existent port should result in an invalid address error being registered.

c. If the entry in the routing table referenced by the leading data character of a packet has not been configured to contain a valid port number, the packet shall be discarded.

d. A packet arriving with a leading data character that references an entry in the routing table which contains an invalid port number should result in an invalid address error being registered.

NOTE The default behaviour of logical address 255 is the same as other logical addresses. Since logical address 255 is not to be used, a packet arriving with that address is discarded and an invalid address error registered.

5.6.8.6 Logical address deletion

a. A routing switch shall be configurable to delete a logical address from the front of the packet (leading data character with a value in the range 32 to 255) before it is forwarded through that port.

b. When a leading character is deleted by a routing switch, only one data character shall be deleted by that routing switch.

c. When a packet arrives, a routing switch shall:

1. First use the leading data character (logical address) of the packet to determine the output port that the packet is forwarded through.
2. If the routing table entry for the logical address specifies that the logical address is deleted, remove the leading data character, exposing another logical address or path address at the front of the packet.
3. Forward the packet through the allocated port.

NOTE The deletion of a logical address is used to expand the number of nodes that can be accessed using logical addressing. A large network is divided into regions where a logical address applies. When crossing a boundary between regions, the logical address is deleted exposing a new logical address that applies to the next region. Regions using logical addressing can be connected to regions using path addressing.

5.6.8.7 Wormhole routing

a. When a packet arrives at a routing switch, the output port that it is routed through shall be determined and allocated to that input port.

b. If the allocated output port is able to accept the start of another packet, the input port shall be connected to the output port so that the packet arriving at the input port is forwarded through the output port.

c. As each N-Char of the packet arrives at the input port it shall be transferred to the output port it is connected to for transmission.

d. An output port shall not transmit any other packet until the packet that it is currently transmitting has finished being sent or has been terminated following an error.

e. If an input port is waiting for packet characters to arrive, the output port that it is connected to shall also wait.

f. If an output port is waiting to transmit packet characters, the input port it is connected to shall also wait.

NOTE An output port is waiting to transmit packet characters when it is waiting for FCTs to arrive to indicate that there is space in the input port at the far end of the link.

g. If the allocated output port is busy, the newly arrived packet shall wait at the input port until the allocated output port is free to transmit the new packet.

h. When the output port finishes transmission of a packet, it shall be available to accept a packet from another input port.

5.6.8.8 Arbitration

a. When two or more input ports have been allocated to the same output port and that output port has just finished sending a packet, the two or more input ports shall compete for connection to the output port.

b. The routing switch shall arbitrate between the two or more input ports competing for access to an output port to decide which input port is connected to the output port and send the next packet through it.

c. A fair arbitration scheme shall be used.

NOTE If a routing switch implements a non-fair arbitration algorithm, there is a possibility that one or more ports can be permanently excluded because others are always selected.

d. When the allocated output port becomes free, the input port connected to it after arbitration shall transfer one packet to the output port and then free the output port for subsequent arbitration and use by the same or another input port.

5.6.8.9 Group adaptive routing

a. Group adaptive routing shall be optional in a routing switch.

b. A routing switch that supports group adaptive routing shall be able to route a packet with a particular logical address to one of several ports.

c. A routing switch that supports group adaptive routing shall:

1. Have entries for each logical address in the routing table which specify one or more output ports that a packet with that logical address is able to be routed to.
2. Route a packet arriving with a leading data character that is a logical address to one port, from the set of ports specified in the routing table, which is currently free.
3. When all of the set of output ports specified for a logical address are currently sending packets, arbitrate between all the input ports waiting to use the specified output ports as each of these output ports becomes free.
4. When an input port wins the arbitration, connect it to the output port that it competed for.

NOTE A free port is one that is not currently used to send a packet.

d. When using a legacy routing switch, that routing switch may also support group adaptive routing for path addresses.

NOTE This clause is included because ECSS-E-ST-50-12C (31 July 2008) allows group adaptive routing with path addresses. This is, however, not recommended as it makes path addressing non-deterministic, thus making it difficult to reliably configure a network, using device configuration ports.

5.6.8.10 Packet multicast

a. Packet multicast (sending of a particular packet to several destinations) shall be optional in a source node.

NOTE 1 It is not necessary for a source node to implement packet multicast.

NOTE 2 Packet multicasting is used with extreme care as it introduces many new ways to block or deadlock sections of the network.

b. Packet multicast in a routing switch shall be optional,.

NOTE It is not necessary for a routing switch to implement packet multicast.

c. A routing switch that supports packet multicast shall provide that support as follows:

1. The routing table in the routing switch is configurable to specify the set of ports (multicast set) that a packet with a particular logical address is concurrently sent through.

2. When a packet arrives with a logical address that is configured for multicast, that packet is sent to each of the output ports in the multicast set.
3. The packet to be multicast is sent to the designated output ports only when all of them are ready to accept a new packet.
4. If one or more of the output ports in the multicast set cannot accept a new N-Char during packet transfer, the input port waits, and the N-Char is not sent to any of the multicast output ports until they are all ready.

d. When a packet arrives with a logical address that is configured for multicast and one or more of the ports in the multicast set are disabled, that packet should be sent to the remaining, enabled output ports.

5.6.8.11 Routing switch reset

a. On reset of a routing switch the following shall occur:
1. All ports are reset.
2. Entries in the routing table for all logical addresses are set to not valid, so that a packet arriving with that logical address is discarded, until the corresponding entry in the routing table is configured.
3. Group adaptive routing is disabled.
4. Packet multicast is disabled.

5.6.8.12 Port time-out

a. A configurable port time-out should be provided in the routing switch which is used by all ports to detect a packet that has become stuck.

b. When logic resources permit, a separate configurable port time-out may be used for each port.

c. A packet shall be regarded by a port as being stuck when the following conditions are all fulfilled:
1. It has started.
2. It has not yet ended.
3. The time since the last data character was sent from the input port to the output port is longer than the port time-out period.

NOTE 1 A packet is started when the first data character of the packet is passed to the output port.

NOTE 2 A packet is not yet ended until the EOP or EEP for that packet is passed to the output port.

d. When a packet is stuck in a routing switch that support port time-outs, the following actions shall be taken to clear the stuck packet:
1. The packet currently being sent is discarded.
2. An EEP is sent to the Encoding Layer.
3. A stuck packet error condition is registered so that the fact that it has occurred can be read via the routing switch configuration port.

NOTE 1 For discarding of a packet see clause 5.5.8.4.

NOTE 2 An EEP sent to the Encoding Layer is sent to the output port.

e. It shall be possible to disable the port time-out so that it is not timed-out when a packet is stuck.

5.6.9 SpaceWire routing switch management parameters

a. The management parameters of a SpaceWire routing switch shall be accessed via the routing switch configuration port.

b. A SpaceWire routing switch shall provide the management parameters for each of its ports as detailed in the following clauses:

1. Clause 5.5.3 data link layer management parameters
2. Clause 5.4.9 encoding layer management parameters
3. Clause 5.3.8 physical layer management parameters

c. The routing table in a routing switch shall be configurable via the routing switch configuration port.

d. The port time-out timer for each port shall be configurable via the routing switch configuration port.

e. The distributed interrupt parameters for each port shall be configurable via the routing switch configuration port.

5.6.10 SpaceWire network

a. A SpaceWire network shall comprise one or more nodes and zero or more routing switches interconnected with SpaceWire links.

NOTE The UML diagram in Figure 5-28 illustrates the components of a SpaceWire network.

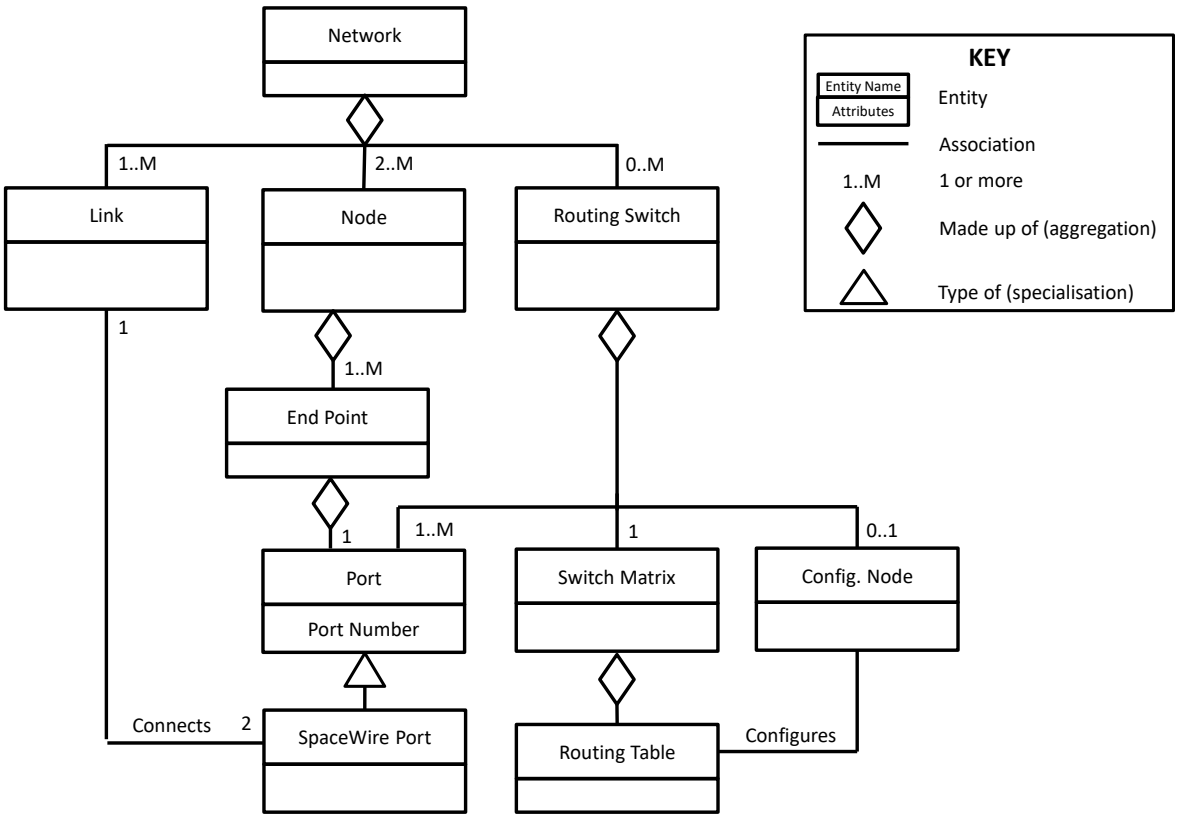


Figure 5-28: Components of a SpaceWire network

b. A SpaceWire network shall send packets from each source node to one or more destination nodes.

c. Packets from a source node shall travel across a SpaceWire link to a routing switch or to the destination node.

d. Packets from a routing switch shall travel across SpaceWire links to other routing switches or to a destination node.

e. A SpaceWire network shall support broadcast of time-codes and distributed interrupt codes as options.

NOTE It is not necessary for a SpaceWire network to support time-codes and distributed interrupt codes.

f. For test purposes it should be possible to loop a port back on itself.

5.7 SpaceWire management information base

5.7.1 Introduction

The Management Information Base specifies the way in which the operations of other SpaceWire layers are configured, controlled and monitored.

5.7.2 General

a. The SpaceWire Management Information Base shall provide a means for a user application in a node to configure and control that node.

b. The SpaceWire Management Information Base shall provide a means for a user application in a node to read the current configuration, control and status information of that node.

c. The SpaceWire Management Information Base shall provide a means for a configuration port in a routing switch to configure and control that router.

d. The SpaceWire Management Information Base shall provide a means for a configuration port in a routing switch to read the current configuration, control and status information of that router.

5.7.3 Physical layer management parameters

a. The physical layer management parameters detailed in clause 5.3.8 shall be incorporated in the SpaceWire Management Information Base.

5.7.4 Encoding layer management parameters

a. The encoding layer management parameters detailed in clause 5.4.9 shall be incorporated in the SpaceWire Management Information Base.

5.7.5 Data link layer management parameters

a. The data link layer management parameters detailed in clause 5.5.3 shall be incorporated in the SpaceWire Management Information Base.

5.7.6 Network layer management parameters

a. The network layer management parameters for a node detailed in clause 5.6.7 shall be incorporated in the SpaceWire Management Information Base.

b. The network layer management parameters for a routing switch detailed in clause 5.6.9 shall be incorporated in the SpaceWire Management Information Base.

6
Service interfaces

6.1 Network layer service interface

6.1.1 Packet service interface

6.1.1.1 Service primitives

a. The service primitives that shall be associated with the SpaceWire packet service are:
   1. SEND_PACKET.request;
   2. RECEIVE_PACKET.indication.

6.1.1.2 SEND_PACKET.request

6.1.1.2.1 Function

a. The SEND_PACKET.request primitive shall be used to send a SpaceWire packet across a SpaceWire network.

6.1.1.2.2 Semantics

a. The SEND_PACKET.request primitive shall have the following parameters:
   1. SEND_PACKET.request (End-point, SpaceWire Packet).

6.1.1.2.3 When Generated

a. When the user has a packet to send over the SpaceWire network, it shall generate a SEND_PACKET.request primitive to request to send the SpaceWire packet over the network.

6.1.1.2.4 Effect On Receipt

a. On receipt of the SEND_PACKET.request primitive the SpaceWire end-point receiving the request shall send the SpaceWire packet though its port.

6.1.1.3 RECEIVE_PACKET.indication

6.1.1.3.1 Function

a. A SpaceWire end-point receiving a SpaceWire packet shall pass a RECEIVE_PACKET.indication primitive to the Read SpaceWire Packet service user to indicate that a SpaceWire packet has arrived at the end-point and is waiting to be read.

6.1.1.3.2 Semantics

a. The RECEIVE_PACKET.indication primitive shall have the following parameters:
   1. RECEIVE_PACKET.indication (End-point, SpaceWire Packet, EOP/EEP).

6.1.1.3.3 When Generated

a. The RECEIVE_PACKET.indication primitive shall be passed to the Receive Packet Service user when a SpaceWire packet is received.

6.1.1.3.4 Effect On Receipt

a. The effect on receipt of the RECEIVE_PACKET.indication primitive by the Receive Packet Service user shall be that the service user reads the received SpaceWire packet.

6.1.2 Time-code service interface

6.1.2.1 Service primitives

a. The service primitives that shall be associated with the time-code service are:
   1. TIME-CODE.request;
   2. TIME-CODE.indication.

6.1.2.2 TIME-CODE.request

6.1.2.2.1 Function

a. The TIME-CODE.request primitive shall be used by the SpaceWire user to request a SpaceWire end-point to broadcast a time-code over the SpaceWire network.

6.1.2.2.2 Semantics

a. The TIME-CODE.request primitive shall have the following parameters:
   1. TIME-CODE.request (End-point, Time-code value).

6.1.2.2.3 When Generated

a. When the SpaceWire user wants to broadcast a time-code, it shall generate a TIME-CODE.request primitive to request one or more

SpaceWire end-points belonging to the node to send the time-code through its SpaceWire port.

6.1.2.2.4 Effect On Receipt

a. On receipt of the TIME-CODE.request primitive the SpaceWire end-point shall send the time-code through its SpaceWire port.

6.1.2.3 TIME-CODE.indication

6.1.2.3.1 Function

a. The function of the TIME-CODE.indication primitive shall be to indicate to the SpaceWire user that a valid time-code has arrived at an end-point.

6.1.2.3.2 Semantics

a. The TIME-CODE.indication primitive shall have the following parameters:
   1. TIME-CODE.indication (End-point, Time-code value).

6.1.2.3.3 When Generated

a. The TIME-CODE.indication primitive shall be passed to the SpaceWire user, when a valid time-code is received.

6.1.2.3.4 Effect On Receipt

a. The effect on receipt of the TIME-CODE.indication primitive by the SpaceWire user shall be application dependent.

6.1.3 Distributed interrupt service interface

6.1.3.1 Service primitives

a. The service primitives that shall be associated with the distributed interrupt service are:
   1. DISTRIBUTED_INTERRUPT.request;
   2. DISTRIBUTED_INTERRUPT.indication;
   3. DISTRIBUTED_INTERRUPT_ACK.request;
   4. DISTRIBUTED_INTERRUPT_ACK.indication.

6.1.3.2 DISTRIBUTED_INTERRUPT.request

6.1.3.2.1 Function

a. The DISTRIBUTED_INTERRUPT.request primitive shall be used by the SpaceWire user to request a SpaceWire end-point to send an interrupt code over the SpaceWire network.

6.1.3.2.2 Semantics

a. The DISTRIBUTED_INTERRUPT.request primitive shall have the following parameters:

   1. DISTRIBUTED_INTERRUPT.request (End-point, Interrupt Identifier).

6.1.3.2.3 When Generated

a. When the SpaceWire user has a distributed interrupt to send, it shall generate a DISTRIBUTED_INTERRUPT.request primitive to request a SpaceWire end-point to send the interrupt code.

6.1.3.2.4 Effect On Receipt

a. On receipt of the DISTRIBUTED_INTERRUPT.request primitive the SpaceWire end-point shall immediately send the interrupt code through its port.

6.1.3.3 DISTRIBUTED_INTERRUPT.indication

6.1.3.3.1 Function

a. The function of the DISTRIBUTED_INTERRUPT.indication primitive shall be to indicate to the SpaceWire user that an interrupt code has arrived at an end-point.

6.1.3.3.2 Semantics

a. The DISTRIBUTED_INTERRUPT.indication primitive shall have the following parameters:

   1. DISTRIBUTED_INTERRUPT.indication (End-point, Interrupt Identifier).

6.1.3.3.3 When Generated

a. The DISTRIBUTED_INTERRUPT.indication primitive shall be passed to the SpaceWire user, when a valid interrupt code is received.

6.1.3.3.4 Effect On Receipt

a. The effect on receipt of the DISTRIBUTED_INTERRUPT.indication primitive by the SpaceWire user shall depend upon the application.

6.1.3.4 DISTRIBUTED_INTERRUPT_ACK.request

6.1.3.4.1 Function

a. The function of the DISTRIBUTED_INTERRUPT_ACK.request primitive shall be to clear a distributed interrupt in the receiving node or when distributed interrupt acknowledgments are being used to clear the distributed interrupt from the entire network.

6.1.3.4.2 Semantics

a. The DISTRIBUTED_INTERRUPT_ACK.request primitive shall have the following parameters:

   1. DISTRIBUTED_INTERRUPT_ACK.request (End-point, Interrupt Identifier)

6.1.3.4.3 When Generated

a. When a distributed interrupt has been received by a node and passed to the host system, the host system shall acknowledge receipt of the distributed interrupt by generating the DISTRIBUTED_INTERRUPT_ACK.request primitive to request an end-point to clear the distributed interrupt.

6.1.3.4.4 Effect On Receipt

a. The effect on receipt of the DISTRIBUTED_INTERRUPT_ACK.request primitive the SpaceWire end-point shall clear the distributed interrupt and if distributed interrupt acknowledgements are enabled send a interrupt acknowledgement code to clear that distributed interrupt from the entire network.

6.1.3.5 DISTRIBUTED_INTERRUPT_ACK.indication

6.1.3.5.1 Function

a. The function of the DISTRIBUTED_INTERRUPT_ACK.indication primitive shall be to indicate to the SpaceWire user that an interrupt acknowledgement code has been received.

6.1.3.5.2 Semantics

a. The DISTRIBUTED_INTERRUPT_ACK.indication primitive shall have the following parameters:

   1. DISTRIBUTED_INTERRUPT_ACK.indication (End-point, Interrupt Identifier).

6.1.3.5.3 When Generated

a. The DISTRIBUTED_INTERRUPT_ACK.indication primitive shall be passed to the SpaceWire user, when a valid interrupt acknowledgement code is received.

6.1.3.5.4 Effect On Receipt

a. The effect on receipt of the DISTRIBUTED_INTERRUPT_ACK.indication primitive by the SpaceWire user shall depend upon the application.

6.2 Data link layer service interface

6.2.1 N-Char service interface

6.2.1.1 Service primitives

a. The service primitives that shall be associated with the N-Char service are:
   1. SEND_NCHAR.request;
   2. READ_NCHAR.request.

6.2.1.2 SEND_NCHAR.request

6.2.1.2.1 Function

a. The SEND_NCHAR.request primitive shall be used to send an N-Char through a port across a link.

6.2.1.2.2 Semantics

a. The SEND_NCHAR.request primitive shall have the following parameters:
   1. SEND_NCHAR.request (Port, N-Char).

6.2.1.2.3 When Generated

a. When the SpaceWire network layer has an N-Char to send over through a particular port, it shall generate a SEND_NCHAR.request primitive to request to load the N-Char into the port for sending over the link.

6.2.1.2.4 Effect On Receipt

a. On receipt of the SEND_NCHAR.request primitive the SpaceWire port receiving the request shall send the N-Char provided that the link is in the Run state.

6.2.1.3 READ_NCHAR.request

6.2.1.3.1 Function

a. The READ_NCHAR.request primitive shall be used to read an N-Char received by a port.

6.2.1.3.2 Semantics

a. The READ_NCHAR.request primitive shall have the following parameters:
   1. READ_NCHAR.request (N-Char).

6.2.1.3.3 When Generated

a. When the network layer is ready to read an N-Char from a port, it shall generate a READ_NCHAR.request primitive to request to read a received N-Char from the port.

6.2.1.3.4 Effect On Receipt

a. The effect on receipt of the READ_NCHAR.request primitive by the data link layer shall be that an N-Char from the specified port is read by the network layer.

6.2.2 Broadcast code service interface

6.2.2.1 Service primitives

a. The service primitives that shall be associated with the BROADCAST_CODE service are:
   1. BROADCAST_CODE.request;
   2. BROADCAST_CODE.indication.

6.2.2.2 BROADCAST_CODE.request

6.2.2.2.1 Function

a. The BROADCAST_CODE.request primitive shall be used by the network layer to request a port to broadcast a broadcast code over the SpaceWire network.

6.2.2.2.2 Semantics

a. The BROADCAST_CODE.request primitive shall have the following parameters:
   1. BROADCAST_CODE.request (Port, Broadcast code value).

6.2.2.2.3 When Generated

a. When the network layer wants to send a broadcast code through a port, it shall generate a BROADCAST_CODE.request primitive to request the port to send the broadcast code.

6.2.2.2.4 Effect On Receipt

a. On receipt of the BROADCAST_CODE.request primitive the port shall send the broadcast code immediately after the character current has finished being sent.

6.2.2.3 BROADCAST_CODE.indication

6.2.2.3.1 Function

a. The function of the BROADCAST_CODE.indication primitive shall be to indicate to the network layer that a broadcast code has arrived at a port.

6.2.2.3.2 Semantics

a. The BROADCAST_CODE.indication primitive shall have the following parameters:
   1. BROADCAST_CODE.indication (Port, Broadcast code value).

6.2.2.3.3 When Generated

a. The BROADCAST_CODE.indication primitive shall be passed to the network layer, when the broadcast code is received.

6.2.2.3.4 Effect On Receipt

a. The effect on receipt of the BROADCAST_CODE.indication primitive by the network layer shall be for the network layer to validate the broadcast code and forward it to the host system attached to an end-point or broadcast it through all other ports of a routing switch.

6.3 Encoding layer service interface

6.3.1 Encoding service interface

6.3.1.1 Service primitives

a. The service primitives that shall be associated with the encoding service are:
   1. TX_CHAR.request;
   2. TX_ENABLE.request.

6.3.1.2 TX_CHAR.request

6.3.1.2.1 Function

a. The TX_CHAR.request primitive shall be used by the data link layer to send a character through its output port.

6.3.1.2.2 Semantics

a. The TX_CHAR.request primitive shall have the following parameters:
   1. TX_CHAR.request (Character).

6.3.1.2.3 When Generated

a. When the data link layer wants to send a character through its output port, it shall generate a TX_CHAR.request primitive to encode the character, serialise it and send it through its output port.

6.3.1.2.4 Effect On Receipt

a. On receipt of the TX_CHAR.request primitive the encoding layer shall encode the character, serialise it, data-strobe encode it, and pass it to the physical layer for driving across the physical medium to the receiver at the far end of the link.

6.3.1.3 TX_ENABLE.request

6.3.1.3.1 Function

a. The TX_ENABLE.request primitive shall be used by the data link layer to request that the output port is enabled.

6.3.1.3.2 Semantics

a. The TX_ENABLE.request primitive shall have no parameters:
   1. TX_ ENABLE.request ().

6.3.1.3.3 When Generated

a. When the data link layer wants to enable the output port so that it can make a connection and send data, it shall generate a TX_ENABLE.request primitive.

6.3.1.3.4 Effect On Receipt

a. On receipt of the TX_ENABLE.request primitive the encoding layer shall enable its output port.

6.3.2 Decoding service interface

6.3.2.1 Service primitives

a. The service primitives that shall be associated with the decoding service are:
   1. RX_CHAR.request;
   2. RX_ENABLE.request;
   3. DISCONNECT.indication;
   4. RECEIVE_ERROR.indication;
   5. gotNull.indication.

6.3.2.2 RX_CHAR.request

6.3.2.2.1 Function

a. The RX_CHAR.request primitive shall be used by the data link layer to receive a character through its input port.

6.3.2.2.2 Semantics

a. The RX_CHAR.request primitive shall have the following parameters:
   1. RX_CHAR.request.

6.3.2.2.3 When Generated

a. When the data link layer wants to read a character from its input port, it shall generate a RX_CHAR.request primitive to read the next character received by the input port.

6.3.2.2.4 Effect On Receipt

a. On receipt of the RX_CHAR.request primitive the encoding layer shall pass a received, de-serialised and decoded character to the data link layer.

6.3.2.3 RX_ENABLE.request

6.3.2.3.1 Function

a. The RX_ENABLE.request primitive shall be used by the data link layer to enable the input port to receive characters and control codes.

6.3.2.3.2 Semantics

a. The RX_ENABLE.request primitive shall not have any parameters:
   1. RX_ENABLE.request ().

6.3.2.3.3 When Generated

a. When the data link layer wants to enable its input port, it shall generate a RX_ENABLE.request primitive.

6.3.2.3.4 Effect On Receipt

a. On receipt of the RX_ENABLE.request primitive the encoding layer shall enable its input port.

6.3.2.4 DISCONNECT.indication

6.3.2.4.1 Function

a. The function of the DISCONNECT.indication primitive shall be to indicate to the data link layer that the SpaceWire link is disconnected.

6.3.2.4.2 Semantics

a. The DISCONNECT.indication primitive shall have no parameters:
   1. DISCONNECT.indication ().

6.3.2.4.3 When Generated

a. The DISCONNECT.indication primitive shall be passed to the data link layer, when the SpaceWire link becomes disconnected.

6.3.2.4.4 Effect On Receipt

a. The effect on receipt of the DISCONNECT.indication primitive by the data link layer shall be for data link layer to move to the ErrorReset state.

6.3.2.5 RECEIVE_ERROR.indication

6.3.2.5.1 Function

a. The function of the RECEIVE_ERROR.indication primitive shall be to indicate to the data link layer that an error has occurred in the receiver.

6.3.2.5.2 Semantics

a. The RECEIVE_ERROR.indication primitive shall have the following parameters:

1. RECEIVE_ERROR.indication (RX_Error_Flags).

NOTE Where the RX_Error_Flags indicate the type of error: parity error or ESC error.

6.3.2.5.3 When Generated

a. The RECEIVE_ERROR.indication primitive shall be passed to the data link layer, when a receive error occurs.

6.3.2.5.4 Effect On Receipt

a. The effect on receipt of the RECEIVE_ERROR.indication primitive by the data link layer shall be for data link layer to move to the ErrorReset state.

6.3.2.6 gotNull.indication

6.3.2.6.1 Function

a. The function of the gotNull.indication primitive shall be to indicate to the data link layer that the first Null has been received after the receiver has been enabled.

6.3.2.6.2 Semantics

a. The gotNull.indication primitive shall not provide any parameters:

1. gotNull.indication ().

6.3.2.6.3 When Generated

a. The gotNull.indication primitive shall be passed to the data link layer, when the first Null is received without any errors after the receiver has been enabled.

6.3.2.6.4 Effect On Receipt

a. The effect on receipt of the gotNull.indication primitive by the data link layer shall be according to the state machine of clause 5.5.6.

6.4 Physical layer service interface

6.4.1 Line transmit service interface

6.4.1.1 Service primitives

a. The service primitives that shall be associated with the line transmit service are:

1. DS_TX.request.

6.4.1.2 DS_TX.request

6.4.1.2.1 Function

a. The DS_TX.request primitive shall be used by the encoding layer to drive the data and strobe signals on to the physical medium.

6.4.1.2.2 Semantics

a. The DS_TX.request primitive shall have the following parameters:

DS_TX.request (Data, Strobe).

6.4.1.2.3 When Generated

a. When the encoding layer wants to drive the data and strobe signals over the physical medium, it shall generate a DS_TX.request primitive.

6.4.1.2.4 Effect On Receipt

a. On receipt of the DS_TX.request primitive the physical layer shall drive the data and strobe signals over the physical medium to the line receiver at the far end of the link.

6.4.2 Line receive service interface

6.4.2.1 Service primitives

a. The service primitives that shall be associated with the line receive service are:

1. DS_RX.request.

6.4.2.2 DS_RX.request

6.4.2.2.1 Function

a. The DS_RX.request primitive shall be used by the encoding layer to receive data and strobe signals from the line receiver.

6.4.2.2.2 Semantics

a. The DS_RX.request primitive shall have the following parameters:

DS_RX.request.

6.4.2.2.3 When Generated

a. When the encoding layer wants to read the data and strobe signals received over the physical medium by the line receiver, it shall generate a DS_RX.request primitive.

6.4.2.2.4 Effect On Receipt

a. On receipt of the DS_RX.request primitive the physical layer shall pass the data and strobe signals received over the physical medium to the encoding layer.

6.5 Management information base service interface

6.5.1 Set parameter service interface

6.5.1.1 Service primitives

a. The service primitive that shall be associated with the set parameter service is:

1. SET_PARAMETER.request.

6.5.1.2 SET_PARAMETER.request

6.5.1.2.1 Function

a. The SET_PARAMETER.request primitive shall be used to write a value to a SpaceWire management parameter.

6.5.1.2.2 Semantics

a. The SET_PARAMETER.request primitive shall have the following parameters:

1. SET_PARAMETER.request (Management Parameter, Value).

6.5.1.2.3 When Generated

a. When a SpaceWire user application has to configure or control the SpaceWire port, it shall generate a SET_PARAMETER.request primitive to request to write a new value to a specified configuration or control register.

6.5.1.2.4 Effect On Receipt

a. On receipt of the SET_PARAMETER.request primitive the management information base shall write the specified value to the specified configuration or control register.

6.5.2 Get parameter service interface

6.5.2.1 Service primitives

a. The service primitive that shall be associated with the get parameter service is:
   1. GET_PARAMETER.request.

6.5.2.2 GET_PARAMETER.request

6.5.2.2.1 Function

a. The GET_PARAMETER.request primitive shall be used to read the current value of a SpaceWire management parameter and status information.

6.5.2.2.2 Semantics

a. The GET_PARAMETER.request primitive shall have the following parameters:
   1. GET_PARAMETER.request (Management Parameter).

6.5.2.2.3 When Generated

a. When a SpaceWire user application wants to read the current status or to check the value of a management parameter, it shall generate a GET_PARAMETER.request primitive to request to read the specified management parameter.

6.5.2.2.4 Effect On Receipt

a. The effect on receipt of the GET_PARAMETER.request primitive by the management information base shall be that the specified management parameter is read and its value returned to the user application.

Annex A (informative) Technical Changes

This Annex gives a brief overview of the technical changes introduced in this revision of the standard.

The standard has been restructured into a series of layers with clearly defined interfaces. A comparison of the levels in the ECSS-E-ST-50-12C (31 July 2008) to the layers of this current standard is provided in Figure A-1.

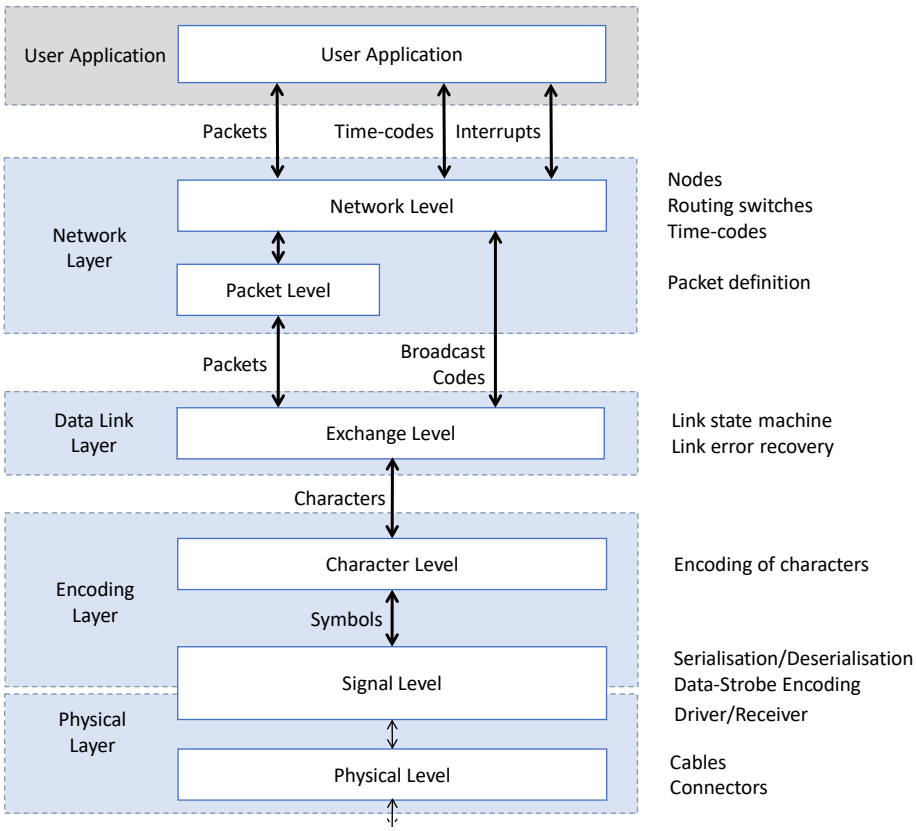


Figure A-1: **Comparison of SpaceWire layers to ECSS-E-ST-50-12C levels**

The layers of the SpaceWire Rev1 standard are compared to the OSI layers in Figure A-2. There is a one to one correspondence, except for the OSI Data Link layer which covers both the SpaceWire Data Link and Encoding layers.

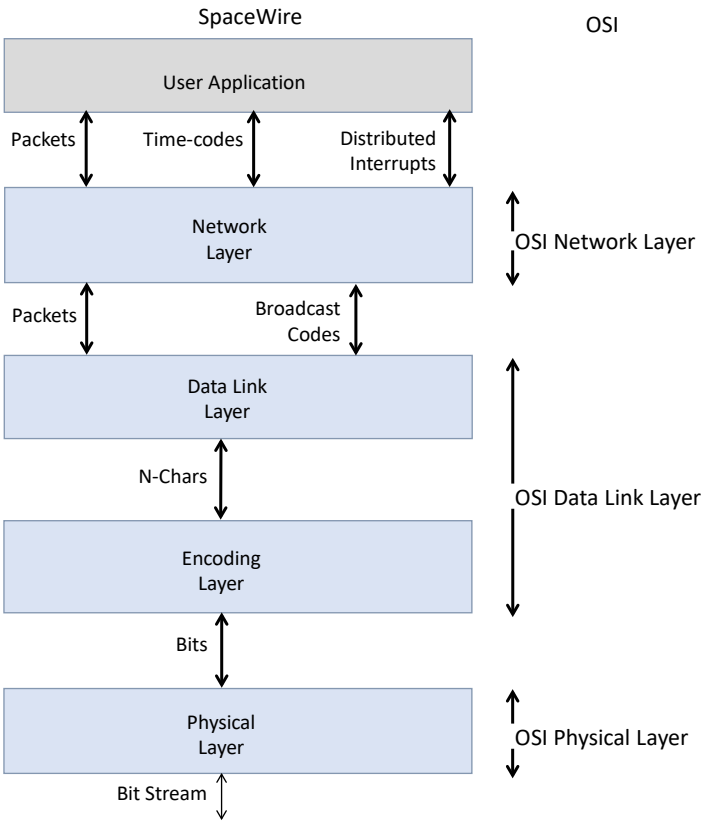


Figure A-2: **Comparison of SpaceWire layers to OSI layers**

The SpaceWire Rev1 standard aimed to improve the SpaceWire standard without making substantial technical changes. The principal changes are summarised below:

- The protocol stack has been rationalised, see clause 5.2.
- Three types of cable assemblies have been introduced: Type AL which is identical to the ECSS-E-ST-50-12C (31 July 2008) cable assembly, Type A which is similar the Type AL except that the inner shields are connected to the outer shield and pin 3 is not connected in the cable assembly, and Type B which is a generic specification allowing matched impedance connectors to be used for SpaceWire. See clause 5.3.4.
- The link initialisation state machine has been improved, see clause 5.5.7. Existing devices compliant to ECSS-E-ST-50-12C (31 July 2008) are interoperable with a new device incorporating the improvements and clause 5.5.7.8 ensures that older devices remain compliant to the new revision of the standard.
- Distributed interrupts have been added and the definition of time-codes amended, see clause 5.6.5.
- Throughout the document various other minor corrections and clarifications have been made to make the standard easier to understand and use.

Bibliography

ECSS-S-ST-00	ECSS system - Description, implementation and general requirements
ECSS-E-ST-50-12C (31 July 2008)	Space engineering – SpaceWire – Links, nodes, routers and networks
ECSS-E-HB-20-07	Space engineering - Electromagnetic compatibility (EMC) handbook
http://www.spacewire.esa.int	SpaceWire website
JESD8C.0:2007	Interface Standard for Nominal 3.0 V/3.3 V Supply Digital Integrated Circuits, September 2001
ESA Bulletin, Volume 145	S. Parkes, P. Armbruster and M. Suess, “SpaceWire On-Board Data-Handing Network”, ESA Bulletin, Volume 145, pp 34-45, February 2011.