Network Discovery Protocols

PROTOCOL SPECIFICATION

SPACEWIRE PLUG-AND-PLAY PROTOCOL

Reference: SSL/08717/DOC/003
Issue: 1.5
Date: 22/03/2013

SciSys UK Ltd.
Methuen Park
Chippenham
Wiltshire
SN14 0GB
United Kingdom

Tel: +44 (0)1249 466466
Fax: +44 (0)1249 466661

www.scisys.co.uk

Intentionally Blank

DOCUMENT CONTROL

This document has no controlled or maintained paper copies. The master document is held as the Microsoft Word file "DOC 003 SpaceWire Plug-and-Play Protocol Specification v1.5.doc" in the Network Discovery Protocols project directory on the SciSys in-house computer network. All files have widespread access for reading only.

Changes to the document will be notified to its audience.

	Name	Function	Date
Prepared by:	Peter Mendham	Author	22/03/2013
Checked by:	Stuart Fowell, Alun Taylor, Martin Kelly (SciSys); Stuart Mills, Steve Parkes (STAR-Dundee)	Reviewers	
Approved by:	Stuart Fowell	Technical Lead	
Authorised by:	Allan Pascoe	Operations Manager	

ISSUE RECORD

Issue	Issue Date	Sections Affected	Relevant Information
1.0	28/10/2011	All	Initial issue (outline only)
1.1	03/07/2012	All	Completed draft specification
1.2	06/09/2012	All	Updated in response to RIDs
1.3	14/02/2013	All	Updated in response to ESA RIDs at RM1. Initial update covering normative content only. Normative content may contradict informative content.
1.4	09/03/2013	All	Final update in response to ESA RIDS at RM1. Normative and informative content updated.
1.5	22/03/2013	4.2.1.4, 4.4.2 5.3.2.2 5.3.3.3.4	Typographical errors corrected. Extra ownership condition added to permit field modification: Device ID must be non-zero. Active Links field was missing from table, added causing renumber of field IDs.

TABLE OF CONTENTS

TABLE OF FIGURES

None

LIST OF TABLES

None

1. INTRODUCTION

1.1 Purpose and Scope

The SpaceWire standard defines the aspects of a highly flexible and capable communication system; however, SpaceWire does not offer a standard mechanism for discovering the topology of a network, or what devices are attached to it. Nor does it offer a standard mechanism for configuring the various aspects of a SpaceWire network, such as links and routers. SpaceWire also lacks standard features to assist detection or configuration beyond the network, in the service domain.

This lack of standardisation for simple tasks that are required on almost all SpaceWire networks limits the level of interoperability which may exist between devices and software, and the extent to which both hardware and software can be re-used between different applications.

1.1.1 Scope of the Project

The main objective of the Network Discovery Protocols activity is to develop, validate and demonstrate a protocol to permit SpaceWire network discovery and management in a standard and interoperable manner, offering 'plug-and-play' features for SpaceWire.

The goals of the project are:

- to gather requirements for a SpaceWire plug-and play protocol;
- to design and specify a SpaceWire plug-and play protocol;
- to develop an implementation of the protocol encompassing hardware and software;
- to provide a test bench, for validation of the protocol;
- to provide a demonstrator to permit demonstration of protocol features.

1.1.2 Scope of the Document

This document presents a full specification for the SpaceWire plug-and-play protocol in the form of a draft ECSS standard. When complete, this document corresponds to deliverable D3 of the Network Discovery Protocols activity: the Protocol Specification.

1.2 Executive Summary

This document presents a full specification for the SpaceWire plug-and-play protocol in the form of a draft ECSS standard. When complete, this document corresponds to deliverable D3 of the Network Discovery Protocols activity: the Protocol Specification.

After this introduction the document has one further sections:

- Section presents the draft ECSS standard.

1.3 Applicable and Reference Documents

1.3.1 Applicable Documents

AD1 ECSS-E-ST-50-12C, "SpaceWire – Links, nodes, routers", 31 July 2008 .

AD2 ECSS-E-ST-50-51C, "SpaceWire protocol identification", 5 February 2010 .

AD3 ECSS-E-ST-50-52C, "SpaceWire - Remote memory access protocol", 5 February 2010 .

AD4 ECSS-E-ST-50-53C, "SpaceWire - CCSDS packet transfer protocol", 5 February 2010 .

AD5 ECSS Drafting Template with Explanations and Examples, June 2009.

AD6 ECSS Standard Template Version 5.6, August 2010.

AD7 ESA, Statement of Work, Network Discovery Protocols (SpaceWire Plug-and-Play), TEC-EDP/DJ/2010/SoW/02, Issue 1, 4 November 2010.

1.3.2 Reference Documents

RD1 SpaceWire-PnP Protocol Definition, Draft A Issue 2.1, 16th September 2009, Peter Mendham, Space Technology Centre, School of Computing, University of Dundee

RD2 "SpaceWire Plug and Play: A Roadmap", International SpaceWire Conference, November 2008, Peter Mendham, Albert Ferrer Florit, Steve Parkes

RD3 "SpaceWire Plug-and-Play: Fault-Tolerant Network Management for Arbitrary Network Topologies", International SpaceWire Conference, September 2007, Albert Ferrer Florit, Martin Süss

RD4 "SpaceWire Plug-and-Play: An Early Implementation and Lessons Learned", AIAA Infotech@Aerospace 2007 Conference and Exhibit, May 2007, Barry M Cook and C Paul H Walker, 4Links Ltd

RD5 "ESA and NASA requirements on SpaceWire PnP", March 2007, ESA & NASA

RD6 "PnP aspects, ESA contribution", 8th SpaceWire Working Group, January 2007, Albert Ferrer Florit, ESA/ESTEC

RD7 "PnP aspects, 4Links contribution", 8th SpaceWire Working Group, January 2007, Barry Cook, Paul Walker, 4Links Ltd.

RD8 "SpaceWire-D", Deterministic Control and Data Delivery Over SpaceWire Networks Draft B, April 2010, S.Parkes

RD9 "The Operation and Uses of the SpaceWire Time-Code", International SpaceWire Seminar, 2003, Steve Parkes, Space Systems Research Group, University of Dundee

RD10 UoD_SpW_10X_DataSheet_Issue-2.0, 13th October 2006, Chris McClements & Steve Parkes, University of Dundee

RD11 SpW-10X SpaceWire Router User Manual, Issue 3.1, 20th January 2008

RD12 SpaceWire-RTC Datasheet 1.8, Dec 2008, Karl Engström & Sandi Habinc (Aeroflex Gaisler), Peter Sinander (RUAG Space)

RD13 SpaceWire Remote Terminal Controller User's Manual RTC-100-0012 Version 2.4, December 2009, Aeroflex Gaisler & RUAG Space

RD14 SMCS116SpW Data Sheet rev.A, July 2007, ATMEL

RD15 SMCS116SpW User Manual rev1.0, July 2007, P. Rastetter

RD16 SMCS332SpW Data Sheet rev.B, May 2008, ATMEL

RD17 SMCS332SpW User Manual 1.5, 10 July 2007, U. Liebstückel

RD18 http://www.eclipse.org/

RD19 User Requirements – SpaceWire Plug-and-Play Protocol, Network Discovery Protocols, SSL/08717/DOC/001, Issue 1.2, 29 February 2012.

1.3.3 Subordinate Documents

None.

1.4 Terms, Definitions and Abbreviated Terms

None.

1.5 Open Issues and Assumptions

1.5.1 Assumptions

None.

1.5.2 Open Issues

None.

End of Section

2. DRAFT ECSS STANDARD

This section contains the draft ECSS standard.

ECSS-E-ST-50-54C
22 March 2013

Space engineering

SpaceWire - Plug-and-play protocol

This document is a draft version of the specification for a plug-and-play protocol to be used on SpaceWire 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 and product assurance 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-50-54C 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-ST-50-54A	Never issued
ECSS-E-ST-50-54B	Never issued
ECSS-E-ST-50-54C 9 March 2013	First issue

Table of contents

Figures

Tables

1
Scope

There are a number of communication protocols that can be used in conjunction with the SpaceWire Standard (ECSS-E-ST-50-12), to provide a comprehensive set of services for onboard user applications. To distinguish between the various protocols a protocol identifier is used, as specified in ECSS-E-ST-50-51.

This Standard specifies the SpaceWire plug-and-play protocol (SpaceWire-PnP, or SpW-PnP), which is one of these protocols that works over SpaceWire.

The aim of SpaceWire-PnP is to provide facilities for the detection and identification of known and unknown devices on a SpaceWire network of known or unknown topology. The protocol also offers functions to support the management of fundamental elements of SpaceWire networks, such as detecting link errors and managing address assignment. The protocol uses a uniform and extensible mechanism for exposing the management parameters of SpaceWire protocols and services or applications using SpaceWire. The SpaceWire-PnP protocol may therefore be used to detect and manage any protocol or service offered on a SpaceWire network.

SpaceWire-PnP builds on the packet format and semantics standardised as part of the remote memory access protocol (RMAP), as specified in ECSS-E-ST-50-52.

This standard may be tailored for the specific characteristic 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-E-ST-50-12	Space engineering - SpaceWire - Links, nodes, routers and networks
ECSS-E-ST-50-51	Space engineering - SpaceWire protocol identification
ECSS-E-ST-50-52	Space engineering - SpaceWire - Remote memory access protocol
Unicode-6.1	The Unicode Standard, Version 6.1, April 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, ECSS-E-ST-50-12, ECSS-E-ST-50-51 and ECSS-E-ST-50-52 apply.

3.2 Terms specific to the present standard

3.2.1 control device

node with the capability to discover and/or manage other network devices

3.2.2 device

node or routing switch

3.2.3 peripheral device

device with the capability to be discovered and/or managed by other network devices

3.3 Abbreviated terms

The following abbreviations are defined and used within this standard:

Abbreviation	Meaning
App	Application
CAS	Compare and swap
CCSDS	Consultative Committee on Space Data Systems
NMS	Network management service
PnP	Plug-and-play
RMAP	Remote memory access protocol
Rsvd	Reserved
SOIS	Spacecraft onboard interface services
SpW	SpaceWire

3.4 Conventions

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

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

Decimal numbers have no prefix.

4
Principles

4.1 Purpose

The SpaceWire plug-and-play protocol is intended to provide a standard mechanism to permit the detection of SpaceWire devices and discovery of both known and unknown SpaceWire networks. The intention is to support automated mechanisms for use during all stages of SpaceWire network development and use. As such, the protocol provides the capability to discover and manage not only devices, but also the applications available on those devices, and the SpaceWire protocols used to permit the applications to communicate over a SpaceWire network.

4.2 Reference Architecture

4.2.1 View of SpaceWire networks

The SpaceWire plug-and-play protocol considers the SpaceWire network from the perspective of SpaceWire-related protocols. Users of the SpaceWire protocols, are referred to as *applications*. From the perspective of the plug-and-play protocol, these applications are the ultimate sources and destinations of SpaceWire packets. In order to communicate over SpaceWire each application uses a set of communication protocols, with the lowest level protocol being SpaceWire itself. These protocols form the layers of a communications stack. For example, an application may use the RMAP protocol, which is scheduled by a protocol utilising time-division multiplexing, this in turn would use SpaceWire. This view of the network is shown in Figure 4-1.

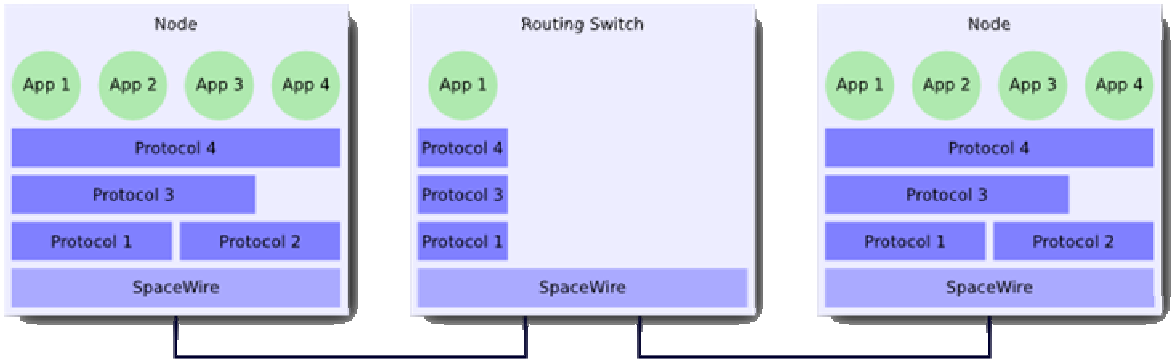


Figure 4-1: SpaceWire plug-and-play view of a SpaceWire network

In Figure 4-1 each of the nodes provides four applications (labelled App 1 – App 4). The vertical arrangement of protocols under each application is intended to represent the protocol stack used by that application to communicate. For example, App 1 uses Protocol 4 which, in turn and on behalf of App 1, uses Protocol 3. Again on behalf of App 1, Protocol 3 uses Protocol 1, which in turn uses SpaceWire. A communications 'channel' is effectively formed, flowing through the communications stack. As an additional example, App 4 uses Protocol 4 which, on behalf of App 4, uses Protocol 2, which in turn uses SpaceWire. The four communications channels, for the four node applications, are depicted as red lines in Figure 4-2.

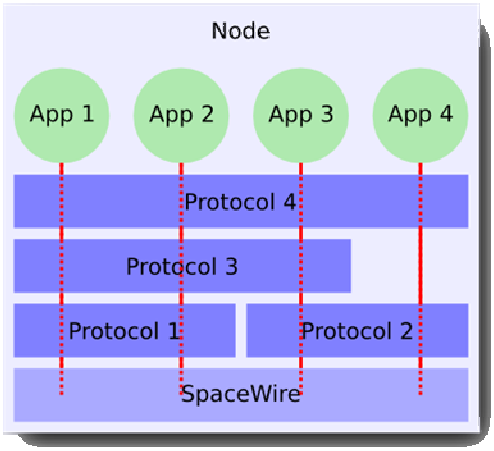


Figure 4-2: Use of Protocols by Applications shown as 'Channels'

Every node and routing switch on a SpaceWire network is referred to by SpaceWire plug-and-play as a *device*. Devices which will be managed by other devices on the network are referred to as *peripheral devices*. Nodes which will be engaged in managing devices on the network are referred to as *control devices*. Devices are the functional elements of a SpaceWire network. The physical *units* of which a SpaceWire network is comprised may each be composed of one or more devices.

4.2.1.1 Plug-and-Play Management Parameters

SpaceWire plug-and-play assumes that each application and protocol may have a number of management parameters which provide information on, or control, its operation. These parameters are depicted in Figure 4-3.

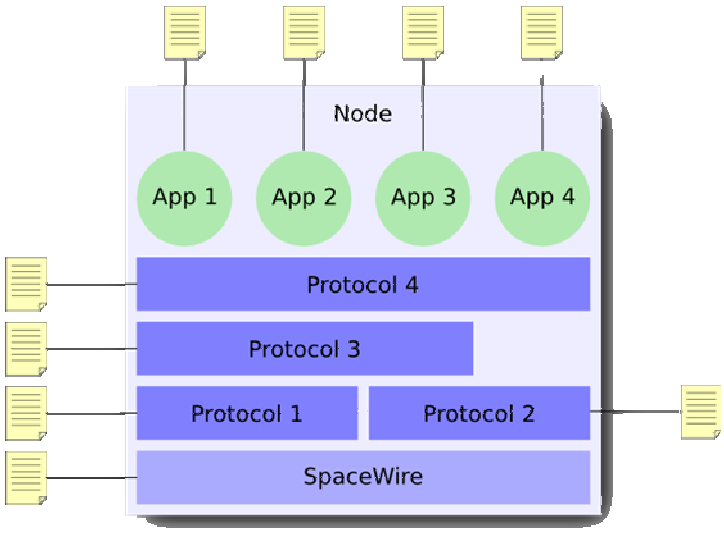


Figure 4-3: Application and protocol management parameters

As each application uses one or more protocols, there may also be management parameters which define an application's use of a protocol. These additional parameters are shown for one application in Figure 4-4. A set of management parameters may be offered for each use of a protocol by an application; for clarity the figure does not show all sets of management parameters.

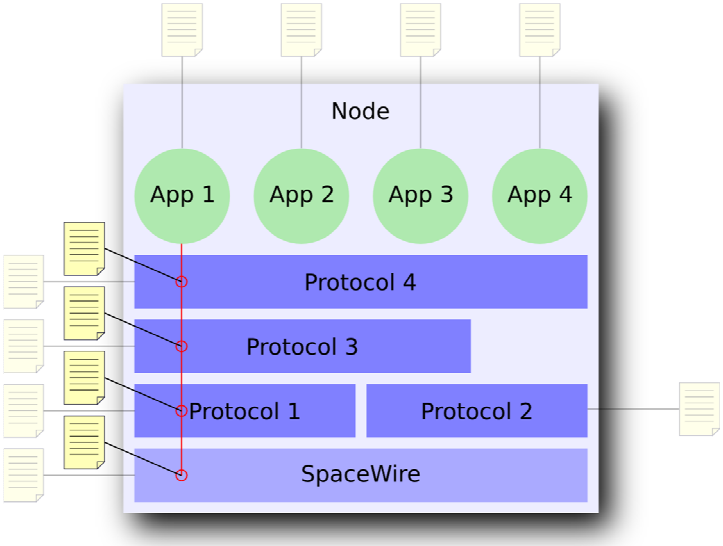


Figure 4-4: Application protocol-use management parameters

The SpaceWire plug-and-play protocol provides a simple, generic mechanism to access management parameters across a network.

The core set of plug-and-play management parameters offered by every peripheral device are those that relate to the device itself. These parameters perform a number of important functions. Firstly, they permit the following to be identified:

1. the type of the device (i.e. whether it is a node or routing switch);
2. the model of the device (i.e. what product the device is);
3. the physical unit that the device belongs to;
4. the connections a device has, enabling network discovery;
5. the protocols a device supports;
6. the applications a device supports;
7. the protocols that each application uses.

Additionally, it is possible to assign an identifier to the device. This is necessary for network discovery to detect network loops.

The parameter access mechanism provided by the SpaceWire plug-and-play protocol then permits management parameters for each supported application, protocol and application protocol-use to be accessed.

4.2.1.2 Device, protocol and service identification

Each device model may be uniquely identified through the use of a Vendor ID and a Product ID. Vendor IDs are uniquely assigned to the organisation manufacturing or selling the device by the SpaceWire Working Group. Product IDs are assigned by the organisation identified by the Vendor ID. It is up to the organisation to ensure that the Product ID uniquely identifies a device model for a given Vendor ID.

Similarly, protocols are uniquely identified by SpaceWire plug-and-play through the use of a Vendor ID and a Protocol ID. There is a special case of a Vendor ID of zero, which indicates a protocol standardised by ECSS. These protocols, together with their identifiers, are listed in ECSS-E-ST-50-51. The protocol IDs specified by ECSS-E-ST-50-51 are all non-zero; an additional special case of a Vendor ID of zero and a Protocol ID of zero permits SpaceWire plug-and-play to identify management parameters associated with SpaceWire itself. Non-zero values of Vendor ID identify the organisation responsible for creating or standardising the protocol. Protocol IDs are assigned by the organisation in an identical manner to Product IDs. This mechanism permits vendor-specific management parameters relating to one or more protocols to be exposed using the standard plug-and-play parameter access mechanisms.

Application identification follows an identical pattern. Each supported application is identified using a Vendor ID and Application ID combination. Again, a Vendor ID of zero identifies an application standardised by ECSS and a non-zero Vendor ID identifies an application created or standardised by another organisation.

4.2.1.3 Protocol and application support lists

Each peripheral device provides a list of the protocols and applications it supports. A control device is then able to access the management parameters

associated with each supported protocol, application or application protocol-use. Rather than specify the unique protocol and/or application identifier with each access operation, access is carried out by specifying the indices of the protocol and application in the support list provided by the device.

For example, suppose a peripheral device provides a list of supported protocols which shows that it supports four protocols, as depicted in Table 4-1. Two standard protocols and two vendor-specific protocols are shown. The Vendor ID and Protocol ID values specified for the vendor-specific protocols are examples only and do not refer to any existing protocol.

Table 4-1: Example protocol support list

Protocol Index	Vendor ID	Protocol ID	Protocol
1	0x0000	0x0000	SpaceWire
2	0x0000	0x0003	SpaceWire-PnP
3	0x0001	0x00F0	Vendor X Protocol A
4	0x0005	0x00F2	Vendor Y Protocol B

If a control device wished to access management parameters associated with SpaceWire it would specify a protocol index of 1. Similarly, if the control device wished to access management parameters associated with SpaceWire-PnP it would specify a protocol index of 2. As both a protocol and application index must always be specified, to access protocol management parameters an application index of zero is specified.

Management parameters associated with application are accessed in an identical manner. The device provides a list of supported applications, the indices of which, together with a protocol index of zero, may be used to access the management parameters associated with an application. An example application support list is shown in Table 4-2. One standard application is listed: the SpaceWire-PnP Network Management Service. Additionally, support for two vendor-specific applications is indicated.

Table 4-2: Example application support list

Application Index	Vendor ID	Application ID	Application	Protocol Use
1	0x0000	0x0001	Network Management Service	1, 2
2	0x0001	0x0001	Vendor X Application A	1, 3
3	0x0005	0x3B97	Vendor Y Application B	1, 4

The management parameters associated with an application's use of a protocol may be accessed by specifying both an application and a protocol index. The protocol indices which are valid for a given application are specified in the table of supported applications provided by the device (as shown in the final column of Table 4-2).

4.2.1.4 Restrictive Assumptions

SpaceWire-PnP makes two assumptions which restrict the design of SpaceWire devices which intend to be compliant with this standard.

1. If a device has multiple links, SpaceWire-PnP requires that access to the protocol and applications supported by a device must be equivalent irrespective of the link used to communicate with the device. This means that the protocol and application support list do not depend on the link used to access the device.
2. The SpaceWire standard restricts the number of path-addressable links on a routing switch to be 31. As the number of fields available to configure a device is finite, SpaceWire-PnP must assume an upper limit on the number of links a device has. For consistency, SpaceWire-PnP limits the number of links a device may have to 31. This ensures that every link in a SpaceWire-PnP network has an equivalent path address.

4.2.2 Network management reference architecture

The SpaceWire plug-and-play protocol assumes a simple layered architecture for carrying out network management activities. The architecture consists of two layers:

1. a network management service; and
2. a communications protocol.

The network management service on a control device carries out network discovery, device identification and device management activities using a communications protocol. A peripheral device permits itself to be managed by offering management parameters which may be accessed using the communications protocol. This is shown in Figure 4-5.

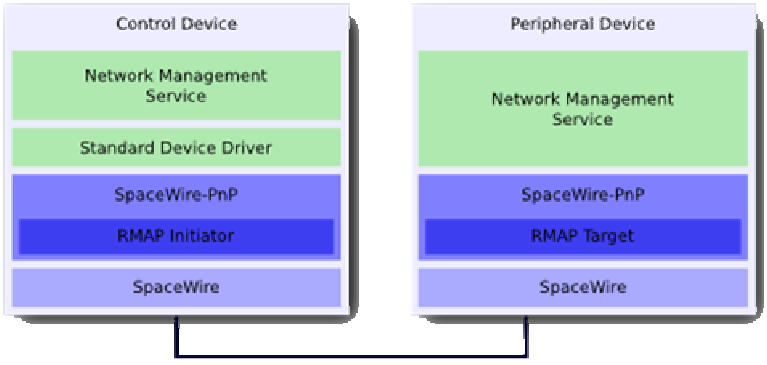


Figure 4-5: Plug-and-play control and peripheral device reference architecture

In Figure 4-5, the network management service on the control device does not access the SpaceWire-PnP communications protocol directly. Rather, protocol access is abstracted by the use of a device driver. This permits the network management service to manage devices which do not support the standard

SpaceWire-PnP communication protocol or the standard peripheral device network management service. Such a situation is depicted in Figure 4-6.

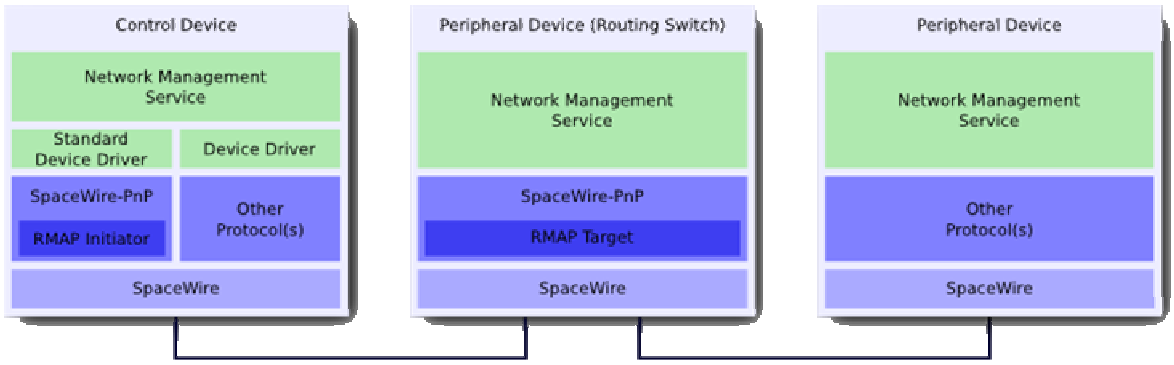


Figure 4-6: Reference architecture highlighting network management service device driver use

This reference architecture is possible providing that the device driver to be used with each device is already known. To permit the use of a device without *a priori* knowledge of the device model it must be possible to, at a minimum, detect the device and its type in a standard way. By providing a standard communications mechanism and a standard core set of management parameters, SpaceWire plug-and-play can be used for network discovery and device identification. Using the information provided by SpaceWire plug-and-play, it is then possible to associate a suitable device driver with a device in order to access other device features.

The decision to place all functionality strictly necessary for network discovery and device identification in the peripheral device means that a basic level of interoperability is assured without standardising the control device network management service or device driver interface. This permits devices to be manufactured supporting plug-and-play without enforcing unnecessary standardisation. To assist the use of SpaceWire-PnP, Annex A illustrates how SpaceWire-PnP may be used by a control device to discover a SpaceWire network.

4.2.3 Device access and ownership

It is possible that a network may have multiple control devices, and that these control devices may not be coordinated. Where this is the case, care must be taken not to corrupt the configuration of peripheral devices. To support operation in such a scenario, SpaceWire plug-and-play regulates the ability to set the value of management parameters on a device. To be able to configure a peripheral device, a control device must first assign the device an identifier. As part of this operation, the peripheral device records the SpaceWire address which the control device specified for the reply information. The peripheral device will then only permit configuration operations which also specify this address for reply information. This enforces the concept of *ownership*, where a single control device is responsible for managing each peripheral device.

Where there are multiple control devices, there is the potential for contention over device ownership. Mechanisms for the resolution of this contention are not specified by this standard.

4.2.4 Mapping on CCSDS/SOIS subnetwork layer

The network management reference architecture is designed to be compliant with the subnetwork service interfaces described by the CCSDS Spacecraft Onboard Interface Services, especially the Device Discovery Service. As such, the SpaceWire plug-and-play protocol is designed to support CCSDS SOIS; however, as the control device network management service is not covered by this standard, compliance with SpaceWire plug-and-play does not ensure compliance with the CCSDS SOIS Device Discovery Service.

4.3 Plug-and-play communications protocol

The plug-and-play communications protocol provides a standard mechanism for accessing peripheral device management parameters from a control device. In order to accomplish this, the communications protocol makes use of the remote memory access protocol (RMAP) specified in ECSS-E-ST-50-52.

The plug-and-play communications protocol provides three operations:

1. write;
2. read; and
3. compare-and-swap.

Each operation accesses peripheral device information in a uniform way. Device information is held in regular sized fields, with each field being 32-bits, and each field has an identifier. Related fields are grouped together into field sets, each of which also has an identifier. Field sets are provided for each supported protocol, service and service protocol use. As such, to identify a field as part of a read, write or compare-and-swap operation, the control device must specify four values:

1. the service identifier;
2. the protocol identifier;
3. the field set identifier; and
4. the field identifier.

A write operation permits a control device to set the value of one or more device fields. Similarly, a read operation permits the control device to access the value of one or more device fields. Where an operation accesses multiple fields, these are a contiguous range of field identifiers, and must all fall within the same field set.

A compare-and-swap operation requests that the peripheral device write the value of a field, only if the current value of that field matches some known value. The peripheral device must therefore read the field, compare it to the specified value and, only if there is a match, write the new value of the field. These read and write operations must be conducted atomically by the

peripheral device. The compare-and-swap operation forms the basis for resolving contention between multiple control devices. As such, the device identifier may only be assigned using a compare-and-swap operation.

4.3.1 SpaceWire-PnP packet addressing

The SpaceWire standard (ECSS-E-ST-50-12) identifies a mechanism for providing access to management information in a routing switch. Here, a path address of zero identifies that the packet should be forwarded to an internal port which may be used for routing switch configuration. A management or configuration packet arriving at the routing switch may therefore be identified by a leading character of zero.

When exploring a network with no *a priori* knowledge, a control device must be able to send a SpaceWire-PnP request to the device connected to a link without knowing the device type (node or routing switch). To make this possible, a SpaceWire-PnP packet sent from a control device always has a path address of zero before the logical address which precedes the protocol identifier (see ECSS-E-ST-50-51).

Depending on the implementation of the device, this zero-value path address may still be present leading the packet received by the SpaceWire-PnP protocol on the peripheral device. The protocol therefore specifically ignores a leading zero character, if it is present, before attempting to process the packet further.

4.4 Network management service

Whilst the communications protocol provides access to fields, the peripheral network management service defines the contents and layout of those fields. The network management service in this standard covers:

1. fields relating to core device information, protocol and application support;
2. fields relating to the status and configuration of the SpaceWire protocol, covering key functions identified by ECSS-E-ST-50-12;
3. fields relating to the implementation and status of the plug-and-play communications protocol on the peripheral device;
4. fields relating to the implementation and status of the network management service on the peripheral device.

Every peripheral device must support the first of these; however, support for the others is identified in the protocol and service support lists provided by the device. Additionally, the network management service for a device may offer access to additional standard, or vendor-specific, fields.

4.4.1 Device Information fields

The device information fields permit various aspects of the device to be identified.

- The type of the device (node or routing switch) is captured as a simple 1-bit flag.
- The Vendor and Product ID permit the device model to be represented numerically; additionally, human-readable descriptions of the vendor and product may be made available as Unicode (UTF-8) strings.
- To assist compatibility and interoperability, the device version may be identified.
- The device may report its current operational status. Although status values are vendor specific, any non-zero value indicates erroneous or failed operation.
- Information about the links a device has is captured. This is sufficient to permit network discovery and includes the number of links a device has, and which links are currently active (connected).
- A 32-bit Device ID may be assigned to the device. The reply address used when the Device ID was assigned is also available to permit control devices exploring the network to determine the device owner.
- The unit to which a device belongs may also be identified through a unit Vendor and Product ID and a 32-bit serial number. The combination of these three identifiers is universally unique.

Device Information also provides a list of supported protocols and applications, together with protocol-application use, as described in Section 4.2.1.3.

A number of different character sets were considered for storing Vendor and Product strings. Although Unicode has restricted support for many East Asian characters (notably those used in Chinese, Japanese, Korean and Vietnamese), it is the most widely-adopted character set with non-Latin character support.

4.4.2 SpaceWire Protocol fields

A device may include SpaceWire in the list of supported protocols provided as part of the Device Information described above. Doing so permits the device to provide access to configuration and management information for the SpaceWire protocol itself.

SpaceWire protocol fields fall into three groups:

- those associated with the device;
- those associated with a single link on the device; and
- those associated with a SpaceWire address (if the device is a routing switch).

Device-level fields permit the control of time-code handling. The internal time-code counter in a device, if present, may be exposed and may be reset. If the device has the capability to generate time-codes, this functionality may also be exposed through standard fields.

The characteristics and status of each of the device links may be determined; additionally, the link state may be controlled. In addition to state control flags identified by the ECSS-E-ST-50-12C, SpaceWire-PnP provides access to two non-standard flags which nevertheless appear in many implementations: a flag

to control the transmission and reception of time-codes over a link and whether transfers over the link should be protected by a watchdog timeout used to prevent blocked links.

Both the link transmit rate, and the rate at which watchdog timeout occurs, are controlled by a two tier divider mechanism, as shown in Figure 4-7. A single base rate is assumed for the device. This may be divided to create a reference rate, again for all links on the device. Individual dividers may then be set to configure the rate for each link. For each divider a valid range is specified.

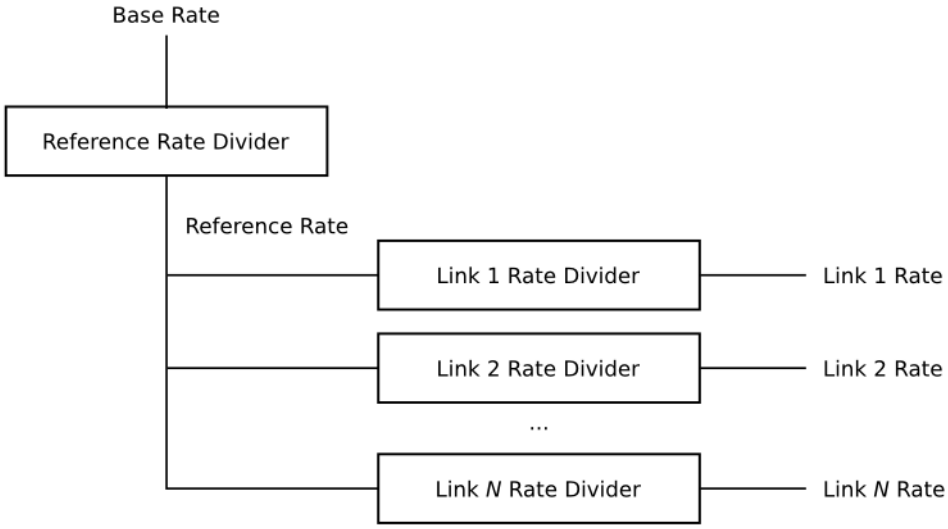


Figure 4-7: Divider Scheme Used for Transmit and Watchdog Rate Configuration

An implementation may use this scheme as much or as little as is relevant. For example, an implementation may require that all links on a device transmit at the same rate (using only the reference rate), or that each link rate is set without any device-level divider (i.e. the reference rate divider is fixed at 1). If an implementation uses a mechanism for controlling transmit or watchdog timeout rates which is not easily represented by this scheme, control may be provided through vendor-specific mechanisms and the rate-related SpaceWire-PnP fields would be reserved.

Extra information to permit the debugging of common network errors can be provided through a dedicated field for each link. This field permits the status of the receive and transmit (if present) first-in-first-out (FIFO) buffers to be determined.

Port Association and Address Control fields permit the determination and configuration of the association between SpaceWire addresses and ports (and therefore links) in a routing switch. Facilities such as group adaptive routing, packet distribution and priority-based arbitration may be managed using these fields.

4.4.3 SpaceWire-PnP Protocol fields

Although this standard requires that devices implementing SpaceWire-PnP support a minimum read and write length (64 bytes and 8 bytes respectively),

an implementation may choose to support longer reads and writes. Such support may be desirable to permit more efficient use of the network. The SpaceWire-PnP Protocol fields permit an implementation to define the longest read and write operation it supports.

4.4.4 Network Management Service fields

The Network Management Service is an application and may therefore expose fields of its own. This standard defines a single field for the service, providing access to a status value indicating the nominal, or otherwise, operation of the service.

4.5 Guide to clause 5

The requirements in Clause 5 are separated into two distinct parts:

1. requirements specifying the communications protocol; and
2. requirements specifying the Network Management Service on peripheral devices.

The communications protocol requirements (clause 5.2) specify the three operations offered by the communications protocol: read, write and compare-and-swap. The way in which the protocol uses RMAP is specified.

The Network Management Service requirements (clause 5.3) specify the various fields which must be, and can be, offered by a peripheral device. The way in which the peripheral device Network Management Service uses the communications protocol is also specified.

5 Protocol requirements

5.1 Overview

The SpaceWire plug-and-play protocol is specified in two parts:

1. the communications protocol used to transfer plug-and-play information across a SpaceWire network; and
2. the Network Management Service on a peripheral device which provides plug-and-play information.

The Network Management Service on a control device is not specified.

5.2 Communications Protocol

5.2.1 Overview

The plug-and-play communications protocol provides a standard mechanism for accessing information on a peripheral device. Information is located in fields which are grouped into field sets. Each field is a regular size: 32-bits.

The plug-and-play protocol is based on the existing remote memory access protocol (RMAP) specified in ECSS-E-ST-50-52C.

5.2.2 Parameters

5.2.2.1 Application_Index

a. The Application_Index parameter shall be used to specify the application to which fields are associated for a read, write or compare-and-swap operation.

b. The Application_Index parameter shall contain a natural number.

c. The valid range of values for the Application_Index parameter shall be 0 to 255 inclusive.

d. The value of the Application_Index parameter shall refer to the application by indexing into the list of supported applications provided by a peripheral device.

e. If the value of the associated Protocol_Index parameter is zero, and the value of the Application_Index parameter is zero, fields associated with the peripheral device shall be accessed.

f. If the value of the associated Protocol_Index parameter is zero, and the value of the Application_Index parameter is non-zero, fields associated with the corresponding application for all protocols shall be accessed.

g. If the value of the associated Protocol_Index parameter is non-zero, and the value of the Application_Index parameter is non-zero, fields associated with the corresponding protocol for the corresponding application shall be accessed.

5.2.2.2 Current_Field_Value

a. The Current_Field_Value parameter shall be used to contain the expected or actual current value of a field for a compare-and-swap operation.

b. The Current_Field_Value parameter shall contain a natural number.

c. The valid range of values for the Current_Field_Value parameter shall be 0 to 4,294,967,295 inclusive.

NOTE This is the range of values which may be held in an unsigned 32-bit number.

5.2.2.3 Field_Count

a. The Field_Count parameter shall be used to contain the desired or actual number of fields to be accessed in a write or read operation.

b. The Field_Count parameter shall contain a natural number.

c. The valid range of values for the Field_Count parameter shall be 0 to 16,383 inclusive.

NOTE This is the number of fields in a field set. The maximum number of fields a read or write operation may access is a complete field set.

5.2.2.4 Field_ID

a. The Field_ID parameter shall be used to specify the field, or first field in a contiguous range, to be accessed in a write, read or compare-and-swap operation.

b. The Field_ID parameter shall contain a natural number.

c. The valid range of values for the Field_ID parameter shall be 0 to 16,383 inclusive.

NOTE This is the number of fields in a field set. The maximum number of fields a read or write operation may access is a complete field set.

5.2.2.5 Field_List

a. The Field_List parameter shall be used to specify

1. the field values to be written in a write operation; or
2. the field values which have been read in a read operation.

b. The Field_List parameter shall consist of 0 to 16,383 field values.

NOTE This is the number of fields in a field set. The maximum number of fields a read or write operation may access is a complete field set.

c. Each field value in a Field_List parameter shall be a natural number.

d. The valid range of values for a field value in a Field_List parameter shall be 0 to 4,294,967,295 inclusive.

NOTE This is the range of values which may be held in an unsigned 32-bit number.

5.2.2.6 FieldSet_ID

a. The FieldSet_ID parameter shall be used to specify the field set to be accessed in a write, read or compare-and-swap operation.

b. The FieldSet_ID parameter shall contain a natural number.

c. The valid range of values for the FieldSet_ID parameter shall be 0 to 31 inclusive.

5.2.2.7 New_Field_Value

a. The New_Field_Value parameter shall be used to contain the desired value of a field for a compare-and-swap operation.

b. The New_Field_Value parameter shall contain a natural number.

c. The valid range of values for the New_Field_Value parameter shall be 0 to 4,294,967,295 inclusive.

NOTE This is the range of values which may be held in an unsigned 32-bit number.

5.2.2.8 Peripheral_Address

a. The Peripheral_Address parameter shall be used to specify the SpaceWire address to be used to reach a peripheral device from a control device.

b. The Peripheral_Address parameter shall consist of 0 or more SpaceWire data characters forming the SpaceWire address.

c. The Peripheral_Address shall be specified using SpaceWire path or regional addressing.

NOTE Using the plug-and-play protocol, it is not possible to address a peripheral with logical addressing.

5.2.2.9 Protocol_Index

a. The Protocol_Index parameter shall be used to specify the protocol to which fields are associated for a read, write or compare-and-swap operation.

b. The Protocol_Index parameter shall contain a natural number.

c. The valid range of values for the Protocol_Index parameter shall be 0 to 31 inclusive.

d. The value of the Protocol_Index parameter shall refer to the protocol by indexing into the list of supported protocols provided by a peripheral device.

e. If the value of the associated Application_Index parameter is zero, and the value of the Protocol_Index parameter is zero, fields associated with the peripheral device shall be accessed.

f. If the value of the associated Application_Index parameter is zero, and the value of the Protocol_Index parameter is non-zero, fields associated with the corresponding protocol for all applications shall be accessed.

g. If the value of the associated Application_Index parameter is non-zero, and the value of the Protocol_Index parameter is non-zero, fields associated with the corresponding protocol for the corresponding application shall be accessed.

5.2.2.10 Reply_Address

a. The Reply_Address parameter shall be used to specify the SpaceWire address to be used to reach the control device from a peripheral device.

b. The Reply_Address parameter shall consist of 0 to 12 SpaceWire data characters forming the SpaceWire address.

c. The Reply_Address shall be specified using SpaceWire path or regional addressing.

NOTE The Reply_Address parameter might contain 0 data characters if logical addressing is being used. The Reply_Address is normally used by the peripheral device to send replies or data back to the control device that requested a write, read or compare-and-swap operation using path addressing. The Reply_Address parameter allows path addressing and regional logical addressing to be used to specify the control device that is to receive the reply.

d. The Reply_Address parameter shall contain zero data characters when a single logical address is to be used for routing the reply to a control device.

NOTE In this case the reply is routed to the control device by the Reply_LA.

e. The Reply_Address shall not include the number of the link on the peripheral device through which to send the reply.

NOTE The peripheral device will automatically send the reply on the link on which the command was received.

5.2.2.11 Reply_LA

a. The Reply_LA field shall be used to specify the logical address to be used to reach the control device from a peripheral device.

b. The Reply_LA field shall consist of a single SpaceWire data character that contains either:

1. The logical address of the control device, if the control device has a logical address, or
2. 0xFE otherwise.

NOTE This is the default logical address as defined by clause 5.2.1.b of ECSS-E-ST-50-51C.

5.2.2.12 Status

a. The Status parameter shall contain the status or error code of a write, read or compare-and-swap operation.

b. The Status parameter shall be a Natural number.

c. Valid values for the status parameter shall be those specified in Clause 5.6 of ECSS-E-ST-50-52C.

5.2.2.13 Transaction_ID

a. The Transaction_ID parameter shall be used to associate:

1. confirmation primitives with request primitives;
2. response primitives with indication primitives; and
3. primitives with RMAP packets.

b. The Transaction_ID parameter shall contain a natural number.

c. The Transaction_ID parameter specified in a response primitive shall have the same value as the Transaction_ID parameter in the indication primitive that caused the response primitive.

5.2.3 Applicability of RMAP

5.2.3.1 RMAP command and reply fields

a. Clauses 5.1.1, 5.1.2 and 5.1.4 to 5.1.17 of ECSS-E-ST-50-52C shall apply.

NOTE These clauses define the various fields of an RMAP command or reply packet. The protocol field is omitted here as its value is different for the plug-and-play protocol.

5.2.3.2 Target SpaceWire Address field

a. The Target SpaceWire Address field shall comprise one or more data characters forming the SpaceWire address which is used to route the command to the target.

NOTE The Target SpaceWire Address is normally stripped off by the time the packet reaches the target.

b. The final data character of the Target SpaceWire Address field shall be zero (0x00).

c. If the Target SpaceWire Address field still contains a data character of zero (0x00) when received by the target, the target shall ignore this character.

NOTE Some SpaceWire interface implementations may not strip off the final zero data character of the Target SpaceWire Address. The data character immediately following the Target SpaceWire Address is the Target Logical Address, which will always have the value 0xFE for SpaceWire-PnP. It is therefore possible to associate a single data character of zero with the Target SpaceWire Address field.

d. SpaceWire path addressing and regional addressing may be used.

5.2.3.3 Protocol Identifier field

a. The Protocol Identifier field shall be an 8-bit field that contains the Protocol Identifier.

b. The Protocol Identifier field shall be set to the value specified for the SpaceWire Plug-and-Play protocol in ECSS-E-ST-50-51C Issue 1.

NOTE This is the value 0x03.

5.2.3.4 Key field

a. The value of the Key field shall be 0x00.

5.2.3.5 Extended Address field

a. The value of the Extended Address field shall be 0x00.

5.2.3.6 Cyclic redundancy code

a. Clause 5.2 of ECSS-E-ST-50-52C shall apply.

5.2.3.7 Error and status codes

a. Clause 5.6 of ECSS-E-ST-50-52C shall apply.

b. Additionally, the status codes in the ranges 0xF0 to 0xFF (inclusive) shall be permitted. The meaning of these status codes is determined by the plug-and-play protocol user.

5.2.3.8 Control device

a. The plug-and-play protocol on a control device shall use an RMAP initiator as defined by clause 5.7.1.2 of ECSS-E-ST-50-52C.

5.2.3.9 Peripheral device

a. The plug-and-play protocol on a peripheral device shall use an RMAP target as defined by clause 5.7.1.3 of ECSS-E-ST-50-52C.

b. The target on a peripheral device shall send replies on the link on which the corresponding RMAP command was received.

NOTE This ensures that the control device does not need to know which link on a peripheral device it is connected to in order to receive a reply.

5.2.4 Write operation

5.2.4.1 Overview

The write operation is invoked by a control device in order to set the value of one or more fields on a peripheral device. The sequence of actions involved in a write operation, highlighting the relationship between NMS (the plug-and-play protocol user), SpaceWire-PnP and RMAP command/reply packets, is shown in Figure 5-1.

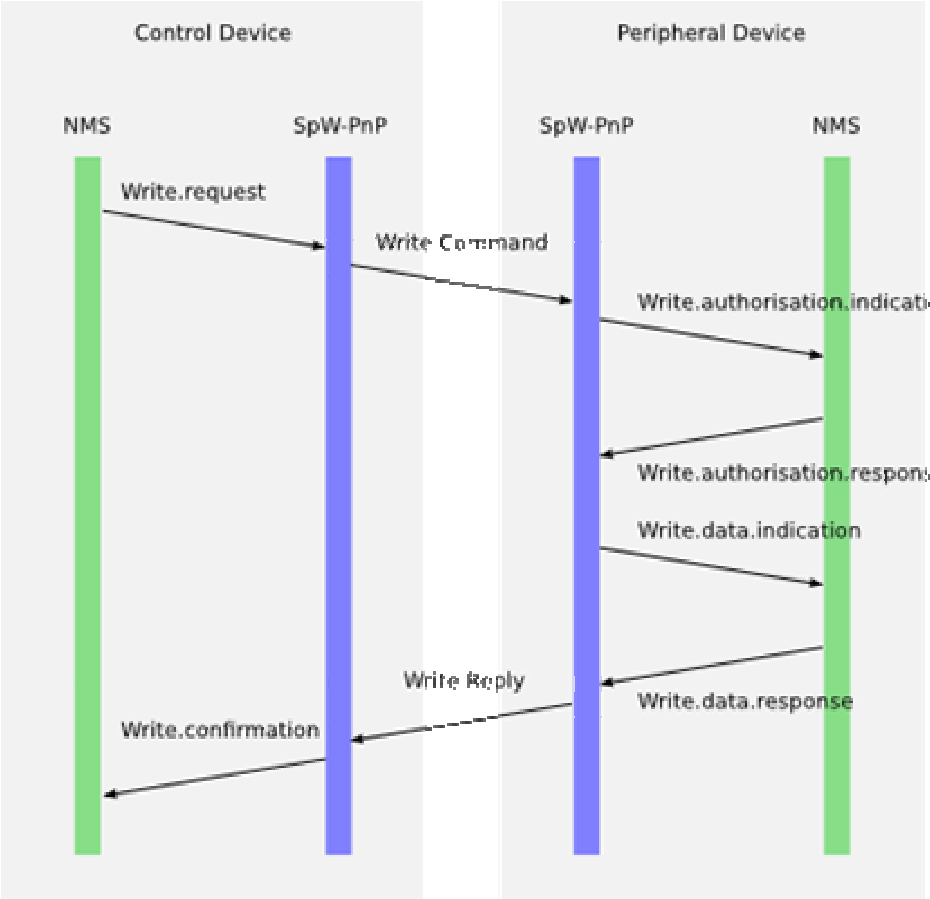


Figure 5-1: Write Operation Sequence

5.2.4.2 Applicability of RMAP

a. The plug-and-play protocol on a control device shall include an RMAP initiator for sending and receiving RMAP packets conveying plug-and-play write operations.

b. The plug-and-play protocol on a peripheral device shall include an RMAP target for receiving and replying to RMAP packets conveying plug-and-play write operations.

c. The following clauses of ECSS-E-ST-50-52C shall apply:

1. clauses 5.3.1 and 5.3.2;
2. clauses 5.3.3.1 to 5.3.3.3;
3. clauses 5.3.3.5 to 5.3.3.12;
4. clauses 5.3.3.4.2 to 5.3.3.4.8.

NOTE This applicability covers all of clause 5.3 of ECSS-E-ST-50-52C except for clause 5.3.3.4.1.

d. When a packet is received at the RMAP target and the Protocol Identifier field contains the value as defined in clause 5.2.3.2 then the RMAP target shall accept the packet.

NOTE This clause replaces clause 5.3.3.4.1 of ECSS-E-ST-50-52C.

5.2.4.3 Control device

a. The plug-and-play protocol service interface on a control device shall comprise the following service primitives relating to the write operation:

1. Write.request;
2. Write.confirmation.

5.2.4.4 Write.request

5.2.4.4.1 Function

a. The Write.request primitive shall be passed by the protocol user to the protocol at the control device to request that one or more fields be written at the peripheral device.

5.2.4.4.2 Semantics

a. The Write.request primitive shall have the following semantics:

Write.request (

Peripheral_Address,

Reply_LA,

Reply_Address,

Transaction_ID,

Application_Index,

Protocol_Index,

FieldSet_ID,

Field_ID,

Field_Count,

Field_List)

5.2.4.4.3 When generated

a. The Write.request primitive shall be generated by the plug-and-play protocol user to request that one or more peripheral device fields be written.

5.2.4.4.4 Effect on receipt

a. On receipt of a Write.request primitive the plug-and-play protocol shall request that the RMAP initiator send a write command.

b. The Target SpaceWire Address field of the write command shall contain the value of the Peripheral_Address parameter suffixed with a data character of value 0x00.

NOTE The suffix address of 0x00 identifies the SpaceWire packet as relating to network management.

c. The Target Logical Address field of the write command shall contain the value 0xFE.

NOTE This is the default logical address as defined by clause 5.2.1.b of ECSS-E-ST-50-51C.

d. The Command field of the write command shall contain the value 0b1111.

NOTE The plug-and-play write operation uses an RMAP Write command specifying incrementing addressing, verification and a reply.

e. The Reply Address field of the write command shall contain the value of the Reply_Address parameter.

f. The Initiator Logical Address field of the write command shall contain the value of the Reply_LA parameter.

g. The Transaction Identifier field of the write command shall contain a value which permits this transaction to be uniquely identified and matched with the value of the Transaction_ID parameter on receipt of a write reply packet.

NOTE The Transaction Identifier field may contain the value of the Transaction_ID parameter.

h. The Address field of the write command shall contain the values of the Application_Index, Protocol_Index, FieldSet_ID and Field_ID parameters encoded as described in Table 5-1.

Table 5-1: Write operation Address field encoding

Bits of Address field	Parameter
0-13	Field_ID
14-18	FieldSet_ID
19-23	Protocol_Index
24-31	Application_Index

i. The Data field of the write command shall contain the value of the Field_List parameter.

j. The field values of the Field_List parameter shall be encoded into the Data field in ascending order.

k. The four bytes comprising each field value of the Field_List parameter shall be encoded in the Data field most significant byte first.

NOTE 1 This is 'big-endian' byte ordering.

NOTE 2 For example, if a field contained in the Field_List parameter had a value of 0x12345678, it would be encoded in the Data field as the bytes 0x12 0x34 0x56 0x78.

l. The Data Length field of the write command shall contain the value of the Field_Count parameter multiplied by 4.

NOTE This is the length of the Data field, in bytes.

5.2.4.5 Write.confirmation

5.2.4.5.1 Function

a. The Write.confirmation primitive shall be passed from the protocol to the protocol user at the control device to confirm the completion of a write operation.

5.2.4.5.2 Semantics

a. The Write.confirmation primitive shall have the following semantics:

Write.confirmation (

Transaction_ID,

Status)

5.2.4.5.3 When generated

a. The Write.confirmation primitive shall be generated on receipt of a write reply by the RMAP initiator.

b. The Transaction_ID parameter shall contain the value of the Transaction_ID parameter supplied to the Write.request primitive which caused the generation of the transaction identified by the Transaction Identifier field of the write reply.

NOTE The Transaction_ID parameter is used by the protocol user to match Write.confirmation primitives to Write.request primitives.

c. The Status parameter shall contain the value of the Status field of the write reply.

5.2.4.5.4 Effect on receipt

a. The effect on receipt of the Write.confirmation primitive shall be defined by the user.

5.2.4.6 Peripheral device

a. The plug-and-play protocol service interface on a peripheral device shall comprise the following service primitives relating to the write operation:

1. Write.authorisation.indication;
2. Write.authorisation.response;
3. Write.data.indication;
4. Write.data.response.

5.2.4.7 Write.authorisation.indication

5.2.4.7.1 Function

a. The Write.authorisation.indication primitive shall be passed from the protocol to the protocol user at the peripheral device to request authorisation for a write operation.

5.2.4.7.2 Semantics

a. The Write.authorisation.indication primitive shall have the following semantics:

Write.authorisation.indication (

Reply_LA,

Reply_Address,

Transaction_ID,

Application_Index,

Protocol_Index,

FieldSet_ID,

Field_ID,

Field_Count)

5.2.4.7.3 When generated

a. The Write.authorisation.indication primitive shall be generated by the protocol on receipt of a valid request for the authorisation of a write command from the RMAP target.

NOTE The RMAP target requests authorisation in response to the receipt of an RMAP command.

b. The Reply_LA parameter shall contain the value of the Initiator Logical Address field of the write command.

c. The Reply_Address parameter shall contain the value of the Reply Address field of the write command.

d. The Transaction_ID parameter shall contain a value which permits this transaction to be uniquely identified on receipt of a Write.authorisation.response primitive.

e. The Application_Index, Protocol_Index and FieldSet_ID and Field_ID parameters shall contain values decoded from the Address field of the write command as described by Table 5-1.

f. The Field_Count parameter shall contain the value of the Data Length field of the write command, divided by 4.

g. If the Data Length field of the write command is not a multiple of 4 bytes then the RMAP target shall be informed that authorisation is denied specifying an “RMAP Command not implemented or authorised” error as specified in clause 5.6 of ECSS-E-ST-50-52C.

h. If the sum of the Field_ID and the Field_Count parameters exceeds 16,384 then the RMAP target shall be informed that authorisation is denied specifying an "RMAP Command not implemented or authorised" error as specified in clause 5.6 of ECSS-E-ST-50-52C.

i. If the RMAP target has been informed that authorisation has been denied under the conditions specified by clauses 5.2.4.7.3.g and 5.2.4.7.3.h then the request for write authorisation is invalid and a Write.authorisation.indication primitive shall not be generated.

5.2.4.7.4 Effect on receipt

a. On receipt of Write.authorisation.indication primitive the protocol user shall determine whether or not to authorise the operation.

b. The protocol user shall then pass a Write.authorisation.response primitive to the protocol.

5.2.4.8 Write.authorisation.response

5.2.4.8.1 Function

a. The Write.authorisation.response primitive shall be passed from the protocol user to the protocol at the peripheral device to authorise or deny authorisation for a write operation.

5.2.4.8.2 Semantics

a. The Write.authorisation.response primitive shall have the following semantics:

Write.authorisation.response (

Transaction_ID,

Status)

5.2.4.8.3 When generated

a. The Write.authorisation.response primitive shall be passed from the protocol user to the protocol at the peripheral device to authorise or deny authorisation for a write operation in response to a Write.authorisation.indication primitive.

5.2.4.8.4 Effect on receipt

a. The Transaction_ID parameter shall be used to match the Write.authorisation.response primitive to an outstanding write command.

b. If no match can be found then the primitive shall be ignored.

c. If the Status parameter is non-zero then the RMAP target shall be informed that authorisation is denied specifying the error identified by the Status parameter.

d. If the Status parameter is zero then the RMAP target shall be informed that the write command is authorised.

5.2.4.9 Write.data.indication

5.2.4.9.1 Function

a. The Write.data.indication primitive shall be passed from the protocol to the protocol user at the peripheral device to request that field data is written.

5.2.4.9.2 Semantics

a. The Write.data.indication primitive shall have the following semantics:

Write.data.indication (

Transaction_ID,

Application_Index,

Protocol_Index,

FieldSet_ID,

Field_ID,

Field_Count,

Field_List)

5.2.4.9.3 When generated

a. The Write.data.indication primitive shall be generated by the protocol on receipt of a request for the writing of data from the RMAP target.

NOTE The RMAP target requests data to be written in response to the authorisation of a received write command.

b. The Transaction_ID parameter shall contain a value which permits this transaction to be uniquely identified on receipt of a Write.data.response primitive.

c. The Application_Index, Protocol_Index, FieldSet_ID and Field_ID parameters shall contain values decoded from the Address field of the write command as described by Table 5-1.

d. The Field_Count parameter shall contain the value of the Data Length field of the write command, divided by 4.

e. The Field_List parameter shall contain the data values from the Data field of the write command.

f. The field values of the Field_List parameter shall be decoded from the Data field in ascending order.

g. The four bytes comprising each field value of the Field_List parameter shall be decoded from the Data field most significant byte first.

NOTE 1 This is 'big-endian' byte ordering.

NOTE 2 For example, if a field value were contained in the Data field as the bytes 0x12 0x34 0x56 0x78, it would be decoded into a field in the Field_List parameter with a value of 0x12345678.

5.2.4.9.4 Effect on receipt

a. On receipt of Write.data.indication primitive the protocol user shall attempt to write the specified field values to the destination.

b. The protocol user shall then pass a Write.data.response primitive to the protocol.

5.2.4.10 Write.data.response

5.2.4.10.1 Function

a. The Write.data.response primitive shall be passed from the protocol user to the protocol at the peripheral device to indicate that field values were written to the destination.

5.2.4.10.2 Semantics

a. The Write.data.response primitive shall have the following semantics:

Write.data.response (

Transaction_ID,

Status)

5.2.4.10.3 When generated

a. The Write.data.response primitive shall be passed from the protocol user to the protocol at the peripheral device to indicate that field values were written to the destination in response to a Write.data.indication primitive.

5.2.4.10.4 Effect on receipt

a. The Transaction_ID parameter shall be used to match the Write.data.response primitive to an outstanding write command.

b. If no match can be found then the primitive shall be ignored.

c. If the Status parameter is non-zero then the RMAP target shall be informed that the write failed specifying the error identified by the Status parameter.

d. If the Status parameter is zero then the RMAP target shall be informed that the write succeeded.

5.2.5 Read operation

5.2.5.1 Overview

The read operation is invoked by a control device in order to get the value of one or more fields from a peripheral device. The sequence of actions involved in a read operation, highlighting the relationship between NMS (the plug-and-play protocol user), SpaceWire-PnP and RMAP command/reply packets, is shown in Figure 5-2.

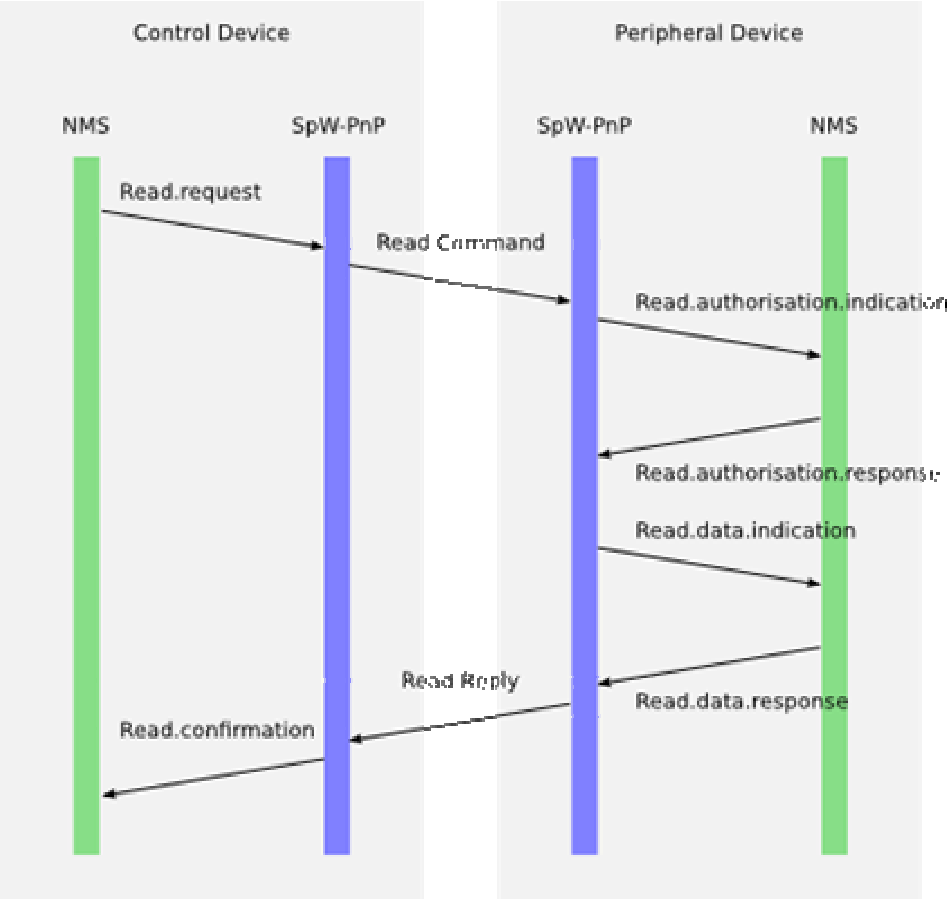


Figure 5-2: Read Operation Sequence

5.2.5.2 Applicability of RMAP

a. The plug-and-play protocol on a control device shall include an RMAP initiator for sending and receiving RMAP packets conveying plug-and-play read operations.

b. The plug-and-play protocol on a peripheral device shall include an RMAP target for receiving and replying to RMAP packets conveying plug-and-play read operations.

c. The following clauses of ECSS-E-ST-50-52C shall apply:

1. clauses 5.4.1 and 5.4.2;
2. clauses 5.4.3.1 to 5.4.3.3;
3. clauses 5.4.3.5 to 5.4.3.13;
4. clauses 5.4.3.4.2 to 5.4.3.4.9.

NOTE This applicability covers all of clause 5.4 of ECSS-E-ST-50-52C except for clause 5.4.3.4.1.

d. When a packet is received at the RMAP target and the Protocol Identifier field contains the value as defined in clause 5.2.3.2 then the RMAP target shall accept the packet.

NOTE This clause replaces clause 5.4.3.4.1 of ECSS-E-ST-50-52C.

5.2.5.3 Control device

a. The plug-and-play protocol service interface on a control device shall comprise the following service primitives relating to the read operation:

1. Read.request;
2. Read.confirmation.

5.2.5.4 Read.request

5.2.5.4.1 Function

a. The Read.request primitive shall be passed by the protocol user to the protocol at the control device to request that one or more fields be read at the peripheral device.

5.2.5.4.2 Semantics

a. The Read.request primitive shall have the following semantics:

Read.request (

Peripheral_Address,

Reply_LA,

Reply_Address,

Transaction_ID,

Application_Index,

Protocol_Index,

FieldSet_ID,

Field_ID,

Field_Count)

5.2.5.4.3 When generated

a. The Read.request primitive shall be generated by the plug-and-play protocol user to request that one or more peripheral device fields be read.

5.2.5.4.4 Effect on receipt

a. On receipt of a Read.request primitive the plug-and-play protocol shall request that the RMAP initiator send a read command.

b. The Target SpaceWire Address field of the read command shall contain the value of the Peripheral_Address parameter suffixed with a data character of value 0x00.

NOTE The suffix address of 0x00 identifies the SpaceWire packet as relating to network management.

c. The Target Logical Address field of the read command shall contain the value 0xFE.

NOTE This is the default logical address as defined by clause 5.2.1.b of ECSS-E-ST-50-51C.

d. The Command field of the read command shall contain the value 0b0011.

NOTE The plug-and-play read operation uses an RMAP Read command specifying incrementing addressing.

e. The Reply Address field of the read command shall be the value of the Reply_Address parameter.

f. The Initiator Logical Address field of the read command shall contain the value of the Reply_LA parameter.

g. The Transaction Identifier field of the read command shall contain a value which permits this transaction to be uniquely identified and matched with the value of the Transaction_ID parameter on receipt of a read reply packet.

NOTE The Transaction Identifier field may contain the value of the Transaction_ID parameter.

h. The Address field of the read command shall contain the values of the Application_Index, Protocol_Index, FieldSet_ID and Field_ID parameters encoded as described in Table 5-1.

i. The Data Length field of the read command shall contain the value of the Field_Count parameter multiplied by 4.

NOTE This is the length of the Data field, in bytes.

5.2.5.5 Read.confirmation

5.2.5.5.1 Function

a. The Read.confirmation primitive shall be passed from the protocol to the protocol user at the control device to confirm the completion of a read operation and to pass the field values read to the protocol user.

5.2.5.5.2 Semantics

a. The Read.confirmation primitive shall have the following semantics:

Read.confirmation (

Transaction_ID,

Field_Count,

Field_List,

Status)

5.2.5.5.3 When generated

a. The Read.confirmation primitive shall be generated on receipt of a read reply by the RMAP initiator.

b. The Transaction_ID parameter shall contain the value of the Transaction_ID parameter supplied to the Read.request primitive which

caused the generation of the transaction identified by the Transaction Identifier field of the read reply.

NOTE The Transaction_ID parameter is used by the protocol user to match Read.confirmation primitives to Read.request primitives.

c. The Field_Count parameter shall contain the value of the Data Length field of the read reply, divided by 4.

d. The Field_List parameter shall contain the data values from the Data field of the read reply.

e. The field values of the Field_List parameter shall be decoded from the Data field in ascending order.

f. The four bytes comprising each field value of the Field_List parameter shall be decoded from the Data field most significant byte first.

NOTE 1 This is 'big-endian' byte ordering.

NOTE 2 For example, if a field value were contained in the Data field as the bytes 0x12 0x34 0x56 0x78, it would be decoded into a field in the Field_List parameter with a value of 0x12345678.

g. The Status parameter shall contain the value of the Status field of the read reply.

5.2.5.5.4 Effect on receipt

a. The effect on receipt of the Read.confirmation primitive shall be defined by the user.

5.2.5.6 Peripheral device

a. The plug-and-play protocol service interface on a peripheral device shall comprise the following service primitives relating to the read operation:

1. Read.authorisation.indication;
2. Read.authorisation.response;
3. Read.data.indication;
4. Read.data.response.

5.2.5.7 Read.authorisation.indication

5.2.5.7.1 Function

a. The Read.authorisation.indication primitive shall be passed from the protocol to the protocol user at the peripheral device to request authorisation for a read operation.

5.2.5.7.2 Semantics

a. The Read.authorisation.indication primitive shall have the following semantics:

Read.authorisation.indication (

Transaction_ID,

Application_Index,

Protocol_Index,

FieldSet_ID,

Field_ID,

Field_Count)

5.2.5.7.3 When generated

a. The Read.authorisation.indication primitive shall be generated by the protocol on receipt of a valid request for the authorisation of a read command from the RMAP target.

NOTE The RMAP target requests authorisation in response to the receipt of an RMAP command.

b. The Transaction_ID parameter shall contain a value which permits this transaction to be uniquely identified on receipt of a Read.authorisation.response primitive.

c. The Application_Index, Protocol_Index and FieldSet_ID and Field_ID parameters shall contain values decoded from the Address field of the read command as described by Table 5-1.

d. The Field_Count parameter shall contain the value of the Data Length field of the read command, divided by 4.

e. If the Data Length field of the read command is not a multiple of 4 bytes then the RMAP target shall be informed that authorisation is denied specifying an "RMAP Command not implemented or authorised" error as specified in clause 5.6 of ECSS-E-ST-50-52C.

f. If the sum of the Field_ID and the Field_Count parameters extracted from the read command (as specified in clauses 5.2.4.6.3.e and 5.2.4.6.3.f respectively) exceeds 16,384 then the RMAP target shall be informed that authorisation is denied specifying an "RMAP Command not implemented or authorised" error as specified in clause 5.6 of ECSS-E-ST-50-52C.

g. If the RMAP target has been informed that authorisation has been denied under the conditions specified by clauses 5.2.5.7.3.e and 5.2.5.7.3.f then the request for read authorisation is invalid and a Read.authorisation.indication primitive shall not be generated.

5.2.5.7.4 Effect on receipt

a. On receipt of Read.authorisation.indication primitive the protocol user shall determine whether or not to authorise the operation.

b. The protocol user shall then pass a Read.authorisation.response primitive to the protocol.

5.2.5.8 Read.authorisation.response

5.2.5.8.1 Function

a. The Read.authorisation.response primitive shall be passed from the protocol user to the protocol at the peripheral device to authorise or deny authorisation for a read operation.

5.2.5.8.2 Semantics

a. The Read.authorisation.response primitive shall have the following semantics:

Read.authorisation.response (

Transaction_ID,

Status)

5.2.5.8.3 When generated

a. The Read.authorisation.response primitive shall be passed from the protocol user to the protocol at the peripheral device to authorise or deny authorisation for a read operation in response to a Read.authorisation.indication primitive.

5.2.5.8.4 Effect on receipt

a. The Transaction_ID parameter shall be used to match the Read.authorisation.response primitive to an outstanding read command.

b. If no match can be found then the primitive shall be ignored.

c. If the Status parameter is non-zero then the RMAP target shall be informed that authorisation is denied specifying the error identified by the Status parameter.

d. If the Status parameter is zero then the RMAP target shall be informed that the read command is authorised.

5.2.5.9 Read.data.indication

5.2.5.9.1 Function

a. The Read.data.indication primitive shall be passed from the protocol to the protocol user at the peripheral device to request that field data is read.

5.2.5.9.2 Semantics

a. The Read.data.indication primitive shall have the following semantics:

Read.data.indication (

Transaction_ID,

Application_Index,

Protocol_Index,

FieldSet_ID,

Field_ID,

Field_Count)

5.2.5.9.3 When generated

a. The Read.data.indication primitive shall be generated by the protocol on receipt of a request for the reading of data from the RMAP target.

NOTE The RMAP target requests data to be read in response to the authorisation of a received read command.

b. The Transaction_ID parameter shall contain a value which permits this transaction to be uniquely identified on receipt of a Read.data.response primitive.

c. The Application_Index, Protocol_Index, FieldSet_ID and Field_ID parameters shall contain values decoded from the Address field of the read command as described by Table 5-1.

d. The Field_Count parameter shall contain the value of the Data Length field of the read command, divided by 4.

5.2.5.9.4 Effect on receipt

a. On receipt of Read.data.indication primitive the protocol user shall attempt to read the specified field values from the destination.

b. The protocol user shall then pass a Read.data.response primitive to the protocol.

5.2.5.10 Read.data.response

5.2.5.10.1 Function

a. The Read.data.response primitive shall be passed from the protocol user to the protocol at the peripheral device to indicate that field values were read from the destination.

5.2.5.10.2 Semantics

a. The Read.data.response primitive shall have the following semantics:

Read.data.response (

Transaction_ID,

Field_Count,

Field_List,

Status)

5.2.5.10.3 When generated

a. The Read.data.response primitive shall be passed from the protocol user to the protocol at the peripheral device to indicate that field values were read from the destination in response to a Read.data.indication primitive.

5.2.5.10.4 Effect on receipt

a. The Transaction_ID parameter shall be used to match the Read.data.response primitive to an outstanding read command.

b. If no match can be found then the primitive shall be ignored.

c. If the Status parameter is non-zero then the RMAP target shall be informed that the read failed specifying a "General" error as specified in clause 5.6 of ECSS-E-ST-50-52C.

d. If the Status parameter is zero then the RMAP target shall be informed that the read succeeded.

e. If the Status parameter is zero then the Data field of the read reply generated by the RMAP target shall contain the value of the Field_List parameter.

f. The field values of the Field_List parameter shall be encoded into the Data field in ascending order.

g. The four bytes comprising each field value of the Field_List parameter shall be encoded in the Data field most significant byte first.

NOTE 1 This is 'big-endian' byte ordering.

NOTE 2 For example, if a field contained in the Field_List parameter had a value of 0x12345678, it would be encoded in the Data field as the bytes 0x12 0x34 0x56 0x78.

h. The Data Length field of the read reply shall contain the value of the Field_Count parameter multiplied by 4.

NOTE This is the length of the Data field, in bytes.

5.2.6 CAS operation

5.2.6.1 Overview

The CAS (compare-and-swap) operation is invoked by a control device in order to set the value of a single field on a peripheral device providing that the current value of the field is equal to a known value. A compare-and-swap operation carries out the following actions at the peripheral device:

- read the current value of the field;
- compare it with the specified value;
- if the current value is the same as the specified value, write the new value and reply indicating success;
- if the current value is different to the specified value, do not write and reply indicating failure.

These actions are conducted atomically, i.e. there should be no way for another operation to modify the value of the field between the read and write actions.

The sequence of actions involved in a compare-and-swap operation, highlighting the relationship between NMS (the plug-and-play protocol user), SpaceWire-PnP and RMAP command/reply packets, is shown in Figure 5-3.

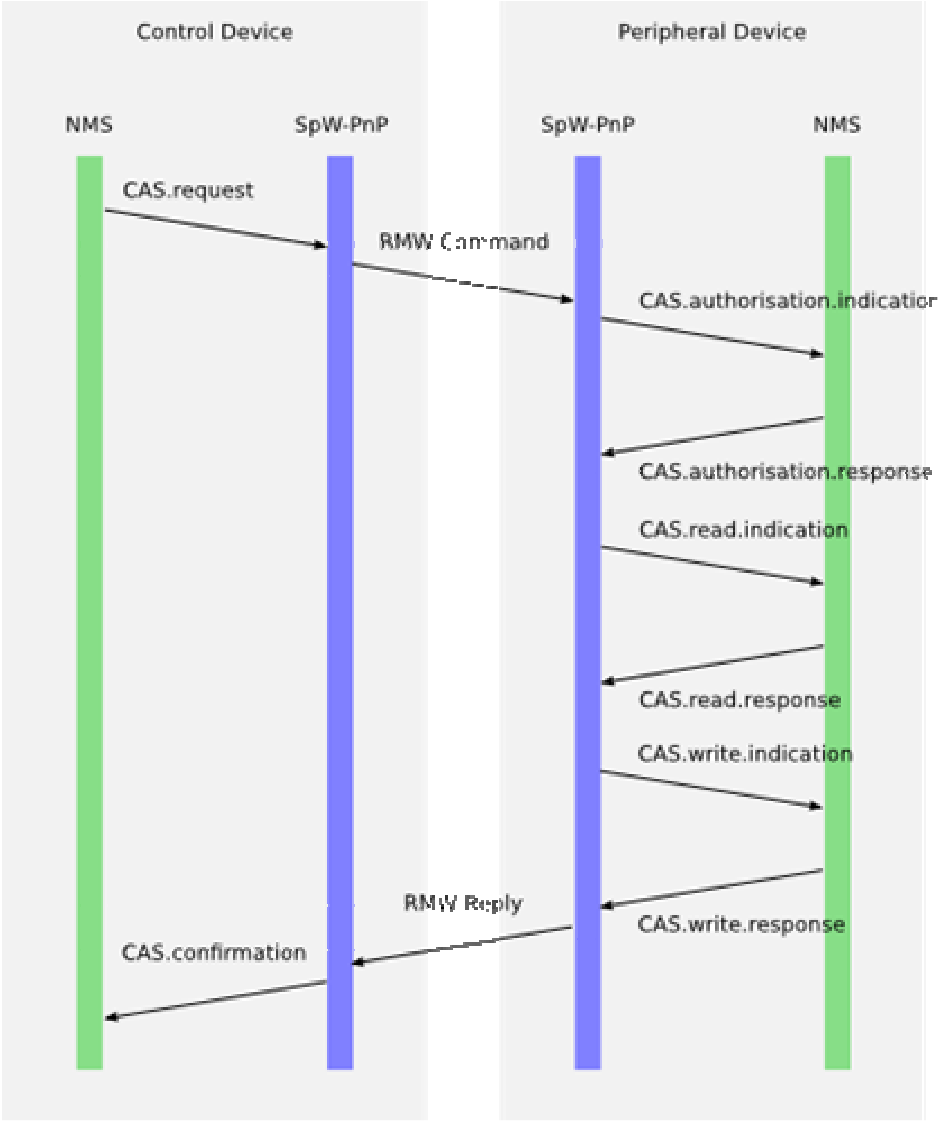


Figure 5-3: CAS Operation Sequence

5.2.6.2 Applicability of RMAP

a. The plug-and-play protocol on a control device shall include an RMAP initiator for sending and receiving RMAP packets conveying plug-and-play CAS operations.

b. The plug-and-play protocol on a peripheral device shall include an RMAP target for receiving and replying to RMAP packets conveying plug-and-play CAS operations.

c. The following clauses of ECSS-E-ST-50-52C shall apply:

1. clauses 5.5.1 and 5.5.2;

2. clauses 5.5.3.1 to 5.5.3.3;

3. clauses 5.5.3.5 to 5.5.3.14;

4. clauses 5.5.3.4.2 to 5.5.3.4.15.

NOTE This applicability covers all of clause 5.5 of ECSS-E-ST-50-52C except for clause 5.5.3.4.1.

d. When a packet is received at the RMAP target and the Protocol Identifier field contains the value as defined in clause 5.2.3.2 then the RMAP target shall accept the packet.

NOTE This clause replaces clause 5.3.3.4.1 of ECSS-E-ST-50-52C.

e. Data shall only be written if the data read matches that specified in the Mask field of the read-modify-write command.

f. The data to be written shall be the data specified in the Data field of the read-modify-write command.

5.2.6.3 Control device

a. The plug-and-play protocol service interface on a control device shall comprise the following service primitives relating to the CAS operation:

1. CAS.request;

2. CAS.confirmation.

5.2.6.4 CAS.request

5.2.6.4.1 Function

a. The CAS.request primitive shall be passed by the protocol user to the protocol at the control device to request that a compare and swap operation be conducted on a field at the peripheral device.

5.2.6.4.2 Semantics

a. The CAS.request primitive shall have the following semantics:

CAS.request (

Peripheral_Address,

Reply_LA,

Reply_Address,

Transaction_ID,

Application_Index,

Protocol_Index,

FieldSet_ID,

Field_ID,

Current_Field_Value,

New_Field_Value)

5.2.6.4.3 When generated

a. The CAS.request primitive shall be generated by the plug-and-play protocol user to request that a compare-and-swap operation be carried out on a peripheral device fields.

5.2.6.4.4 Effect on receipt

a. On receipt of a CAS.request primitive the plug-and-play protocol shall request that the RMAP initiator send a read-modify-write command.

b. The Target SpaceWire Address field of the read-modify-write command shall contain the value of the Peripheral_Address parameter suffixed with a data character of value 0x00.

NOTE The suffix address of 0x00 identifies the SpaceWire packet as relating to network management.

c. The Target Logical Address field of the read-modify-write command shall contain the value 0xFE.

NOTE This is the default logical address as defined by clause 5.2.1.b of ECSS-E-ST-50-51C.

d. The Command field of the read-modify-write command shall contain the value 0b0111.

e. The Reply Address field of the read-modify-write command shall contain the value of the Reply_Address parameter.

f. The Initiator Logical Address field of the read-modify-write command shall contain the value of the Reply_LA parameter.

g. The Transaction Identifier field of the read-modify-write command shall contain a value which permits this transaction to be uniquely identified and matched with the value of the Transaction_ID parameter on receipt of a read-modify-write reply packet.

NOTE The Transaction Identifier field may contain the value of the Transaction_ID parameter.

h. The Address field of the read-modify-write command shall contain the values of the Application_Index, Protocol_Index, FieldSet_ID and Field_ID parameters encoded as described in Table 5-1.

i. The Data field of the read-modify-write command shall contain the value of the New_Field_Value parameter.

j. The four bytes comprising the field value of the New_Field_Value parameter shall be encoded in the Data field most significant byte first.

NOTE 1 This is 'big-endian' byte ordering.

NOTE 2 For example, if a field contained in the Field_List parameter had a value of 0x12345678, it would be encoded in the Data field as the bytes 0x12 0x34 0x56 0x78.

k. The Mask field of the read-modify-write command shall contain the value of the Current_Field_Value parameter.

l. The four bytes comprising the field value of the Current_Field_Value parameter shall be encoded in the Mask field most significant byte first.

m. The Data Length field of the write command shall contain the value 0x08.

NOTE This is the length of the Data field plus the length of the Mask field, in bytes.

5.2.6.5 CAS.confirmation

5.2.6.5.1 Function

a. The CAS.confirmation primitive shall be passed from the protocol to the protocol user at the control device to confirm the completion of a compare-and-swap operation.

5.2.6.5.2 Semantics

a. The CAS.confirmation primitive shall have the following semantics:

CAS.confirmation (

Transaction_ID,

Status)

5.2.6.5.3 When generated

a. The CAS.confirmation primitive shall be generated on receipt of a read-modify-write reply by the RMAP initiator.

b. The Transaction_ID parameter shall contain the value of the Transaction_ID parameter supplied to the CAS.request primitive which caused the generation of the transaction identified by the Transaction Identifier field of the read-modify-write reply.

NOTE The Transaction_ID parameter is used by the protocol user to match CAS.confirmation primitives to CAS.request primitives.

c. The Status parameter shall contain the value of the Status field of the read-modify-write reply.

5.2.6.5.4 Effect on receipt

a. The effect on receipt of the CAS.confirmation primitive shall be defined by the user.

5.2.6.6 Peripheral device

a. The plug-and-play protocol service interface on a peripheral device shall comprise the following service primitives relating to the compare-and-swap operation:

1. CAS.authorisation.indication;
2. CAS.authorisation.response;
3. CAS.read.indication;
4. CAS.read.response;

5. CAS.write.indication;
6. CAS.write.response.

5.2.6.7 CAS.authorisation.indication

5.2.6.7.1 Function

a. The CAS.authorisation.indication primitive shall be passed from the protocol to the protocol user at the peripheral device to request authorisation for a compare-and-swap operation.

5.2.6.7.2 Semantics

a. The CAS.authorisation.indication primitive shall have the following semantics:

CAS.authorisation.indication (

Reply_LA,

Reply_Address,

Transaction_ID,

Application_Index,

Protocol_Index,

FieldSet_ID,

Field_ID)

5.2.6.7.3 When generated

a. The CAS.authorisation.indication primitive shall be generated by the protocol on receipt of a valid request for the authorisation of a read-modify-write command from the RMAP target.

NOTE The RMAP target requests authorisation in response to the receipt of an RMAP command.

b. The Reply_LA parameter shall contain the value of the Initiator Logical Address field of the read-modify-write command.

c. The Reply_Address parameter shall contain the value of the Reply Address field of the read-modify-write command.

d. The Transaction_ID parameter shall contain a value which permits this transaction to be uniquely identified on receipt of a CAS.authorisation.response primitive.

e. The Application_Index, Protocol_Index, FieldSet_ID and Field_ID parameters shall contain values decoded from the Address field of the read-modify-write command as described by Table 5-1.

f. If the Data Length field of the read-modify-write command does not contain the value 0x08 then the RMAP target shall be informed that authorisation is denied specifying an “RMAP Command not implemented or authorised” error as specified in clause 5.6 of ECSS-E-ST-50-52C.

g. If the RMAP target has been informed that authorisation has been denied under the conditions specified by clause 5.2.6.7.3.f then the request for read-modify-write authorisation is invalid and a CAS.authorisation.indication primitive shall not be generated.

5.2.6.7.4 Effect on receipt

a. On receipt of CAS.authorisation.indication primitive the protocol user shall determine whether or not to authorise the operation.

b. The protocol user shall then pass a CAS.authorisation.response primitive to the protocol.

5.2.6.8 CAS.authorisation.response

5.2.6.8.1 Function

a. The CAS.authorisation.response primitive shall be passed from the protocol user to the protocol at the peripheral device to authorise or deny authorisation for a compare-and-swap operation.

5.2.6.8.2 Semantics

a. The CAS.authorisation.response primitive shall have the following semantics:

CAS.authorisation.response (

Transaction_ID,

Status)

5.2.6.8.3 When generated

a. The CAS.authorisation.response primitive shall be passed from the protocol user to the protocol at the peripheral device to authorise or deny authorisation for a compare-and-swap operation in response to a CAS.authorisation.indication primitive.

5.2.6.8.4 Effect on receipt

a. The Transaction_ID parameter shall be used to match the CAS.authorisation.response primitive to an outstanding read-modify-write command.

b. If no match can be found then the primitive shall be ignored.

c. If the Status parameter is non-zero then the RMAP target shall be informed that authorisation is denied specifying the error identified by the Status parameter.

d. If the Status parameter is zero then the RMAP target shall be informed that the read-modify-write command is authorised.

5.2.6.9 CAS.read.indication

5.2.6.9.1 Function

a. The CAS.read.indication primitive shall be passed from the protocol to the protocol user at the peripheral device to request that field data is read.

5.2.6.9.2 Semantics

a. The CAS.read.indication primitive shall have the following semantics:

CAS.read.indication (

Transaction_ID,

Application_Index,

Protocol_Index,

FieldSet_ID,

Field_ID)

5.2.6.9.3 When generated

a. The CAS.read.indication primitive shall be generated by the protocol on receipt of a request for the reading of data from the RMAP target.

NOTE The RMAP target requests data to be read in response to the authorisation of a received read-modify-write command.

b. The Transaction_ID parameter shall contain a value which permits this transaction to be uniquely identified on receipt of a CAS.read.response primitive.

c. The Application_Index, Protocol_Index, FieldSet_ID and Field_ID parameters shall contain values decoded from the Address field of the read-modify-write command as described by Table 5-1.

5.2.6.9.4 Effect on receipt

a. On receipt of CAS.read.indication primitive the protocol user shall attempt to read the specified field value from the destination.

b. The protocol user shall then pass a CAS.read.response primitive to the protocol.

5.2.6.10 CAS.read.response

5.2.6.10.1 Function

a. The CAS.read.response primitive shall be passed from the protocol user to the protocol at the peripheral device to indicate that a field value was read from the destination.

5.2.6.10.2 Semantics

a. The CAS.read.response primitive shall have the following semantics:

CAS.read.response (

Transaction_ID,

Current_Field_Value,

Status)

5.2.6.10.3 When generated

a. The CAS.read.response primitive shall be passed from the protocol user to the protocol at the peripheral device to indicate that the field value was read from the destination in response to a CAS.read.indication primitive.

5.2.6.10.4 Effect on receipt

a. The Transaction_ID parameter shall be used to match the CAS.read.response primitive to an outstanding read-modify-write command.

b. If no match can be found then the primitive shall be ignored.

c. If the Status parameter is non-zero then the RMAP target shall be informed that the read failed specifying a “General” error as specified in clause 5.6 of ECSS-E-ST-50-52C.

d. If the Status parameter is zero then the RMAP target shall be informed that the read succeeded.

e. If the Status parameter is zero then the Data field specified to the RMAP target as the data read shall contain the value of the Current_Field_Value parameter.

f. The four bytes comprising the field value of the Current_Field_Value parameter shall be encoded in the Data field most significant byte first.

NOTE 1 This is ‘big-endian’ byte ordering.

NOTE 2 For example, if a field contained in the Field_List parameter had a value of 0x12345678, it would be encoded in the Data field as the bytes 0x12 0x34 0x56 0x78.

5.2.6.11 CAS.write.indication

5.2.6.11.1 Function

a. The CAS.write.indication primitive shall be passed from the protocol to the protocol user at the peripheral device to request that a field value is written.

5.2.6.11.2 Semantics

a. The CAS.write.indication primitive shall have the following semantics:

CAS.write.indication (

Transaction_ID,

Application_Index,

Protocol_Index,

FieldSet_ID,

Field_ID,

New_Field_Value)

5.2.6.11.3 When generated

a. The CAS.write.indication primitive shall be generated by the protocol on receipt of a request for the writing of data from the RMAP target.

NOTE The RMAP target requests data to be written in response to the authorisation of a received read-modify-write command.

b. The Transaction_ID parameter shall contain a value which permits this transaction to be uniquely identified on receipt of a CAS.write.response primitive.

c. The Application_Index, Protocol_Index, FieldSet_ID and Field_ID parameters shall contain values decoded from the Address field of the write command as described by Table 5-1.

d. The New_Field_Value parameter shall contain the data value from the Data field of the write command.

e. The four bytes comprising the field value of the New_Field_Value parameter shall be decoded from the Data field most significant byte first.

NOTE 1 This is 'big-endian' byte ordering.

NOTE 2 For example, if a field value were contained in the Data field as the bytes 0x12 0x34 0x56 0x78, it would be decoded into a field in the Field_List parameter with a value of 0x12345678.

5.2.6.11.4 Effect on receipt

a. On receipt of CAS.write.indication primitive the protocol user shall attempt to write the specified field value to the destination.

b. The protocol user shall then pass a CAS.write.response primitive to the protocol.

5.2.6.12 CAS.write.response

5.2.6.12.1 Function

a. The CAS.write.response primitive shall be passed from the protocol user to the protocol at the peripheral device to indicate that the field value was written to the destination.

5.2.6.12.2 Semantics

a. The CAS.write.response primitive shall have the following semantics:

CAS.write.response (

Transaction_ID,

Status)

5.2.6.12.3 When generated

a. The CAS.write.response primitive shall be passed from the protocol user to the protocol at the peripheral device to indicate that the field value was written to the destination in response to a CAS.write.indication primitive.

5.2.6.12.4 Effect on receipt

a. The Transaction_ID parameter shall be used to match the CAS.write.response primitive to an outstanding read-modify-write command.

b. If no match can be found then the primitive shall be ignored.

c. If the Status parameter is non-zero then the RMAP target shall be informed that the read-modify-write failed specifying the error identified by the Status parameter.

d. If the Status parameter is zero then the RMAP target shall be informed that the write succeeded.

5.3 Network management service

5.3.1 Overview

The Network Management Service on a peripheral device is standardised to permit greater interoperability between devices during network discovery and management. Core device information, contained in the Device Information fields, is provided by all devices. The ability to configure features of the SpaceWire protocol, or to query information about the plug-and-play protocol or the Network Management Service itself, may also be provided by the device. Support for these features is indicated in the Device Information.

5.3.2 Field Access

5.3.2.1 Use of plug-and-play communication protocol

a. The peripheral device Network Management Service shall use the plug-and-play communication protocol as specified in clause 5.2.

5.3.2.2 Authorisation

a. It shall be possible to read any field in a field set which has one or more defined fields.

b. It shall be possible to read any field identified as Reserved.

c. The contents of any field identified as Reserved shall be zero.

d. It shall not be possible to write to any field identified as read-only.

e. If authorisation is denied due to an attempt to write to a read-only field, a status of Read-Only Field (as defined in Table 5-2) shall be specified to the plug-and-play protocol.

f. Any information written to a Reserved field which is not identified as read-only shall be discarded.

g. If all fields in a field set are undefined then it shall not be possible to apply read, write or compare-and-swap operations to any field in the field set.

h. If authorisation is denied due to an attempt to apply a write or compare-and-swap operation to a field set in which all fields are reserved, a status of Field Set Reserved (as defined in Table 5-2) shall be specified to the plug-and-play protocol.

i. If a field is not identified as read-only then it shall be possible to apply the compare-and-swap operation to the field in order to set the field value.

j. If a field is not identified as read-only and the write operation is not explicitly forbidden then it shall be possible to apply the write operation to the field in order to set the field value.

k. It shall only be possible to set the value of a field other than the Device ID field (see clause 5.3.3.2.7) using a write or compare-and-swap operation if:

1. the Device ID field (see clause 5.3.3.2.7) is non-zero;

2. the Reply Logical Address of the command matches the Owner Logical Address in the Link Information field (see clause 5.3.3.2.5);

3. the Reply Address of the command matches the Owner Address in the Owner Address fields (see clause 5.3.3.2.6) governed by the Owner Address Length field of the Link Information field (see clause 5.3.3.2.5); and

4. the link being used to access the device matches the Owner Link in the Link Information field (see clause 5.3.3.2.5).

NOTE 1 This ensures that only the current owner of a device may modify the device configuration.

NOTE 2 Only the write operation permits the modification of multiple fields (by the current owner) as a CAS operation is limited to a single field.

l. If authorisation is denied due to an invalid reply address as determined by clause 5.3.2.2.k, a status of Unauthorised Access (as defined in Table 5-2) shall be specified to the plug-and-play protocol.

m. The check for reply address validity, specified in clause 5.3.2.2.k, shall take precedence over other all authorisation checks.

5.3.2.3 Status Values

a. The status codes listed in Table 5-2 shall be used for the peripheral device network management service.

Table 5-2: Protocol Authorisation Status Values

Status Value	Meaning
0xF0	Unauthorised Access
0xF1	Reserved Field Set
0xF2	Read-Only Field
0xF3 – 0xFF	Reserved

5.3.2.4 RMAP Target Read and Write Support

a. The RMAP target used by the communications protocol in the peripheral device shall support read commands requesting eight fields (64 bytes) or more.

b. The RMAP target used by the communications protocol in the peripheral device shall support write commands setting two fields (8 bytes) or more.

5.3.2.5 Implementation timing

a. The peripheral device network management service, using the plug-and-play protocol and RMAP target, shall be capable of completing the transmission of a reply within 100 ms of the reception of the first character of the command.

5.3.3 Device information

5.3.3.1 Overview

Every peripheral device offers Device Information which permits the device to be uniquely identified and allows a control device to discover an unknown network. Additionally, the Device Information lists the protocols and applications supported by the device and which may be managed by referring to their respective indices.

5.3.3.2 Fields

5.3.3.2.1 Device Vendor and Product ID field

a. The Device Vendor and Product ID field shall identify the vendor and type of the device.

b. The vendor shall be identified by the Vendor ID.

c. The Vendor ID shall be a 16-bit identifier uniquely identifying the organisation manufacturing or selling the device.

d. The Vendor ID shall be assigned by the SpaceWire Working Group.

e. The Product ID shall be a 16-bit identifier uniquely identifying the device type for a given organisation manufacturing or selling the device.

f. The Product ID shall be assigned by the organisation manufacturing or selling the device.

g. The Device Vendor and Product ID field shall be encoded in the Device Vendor and Product ID field as shown in Table 5-3.

Table 5-3: Device Vendor and Product ID field

31 — 16	15 — 0
Vendor ID	Product ID

h. The Device Vendor and Product ID field shall be read-only.

i. The value of the Device Vendor and Product ID field shall be static.

NOTE This means that the value of the field may not change during the operation of the device.

5.3.3.2.2 Version field

a. The Version field shall identify the version of the device type.

b. The value of the Version field shall be assigned by the organisation manufacturing or selling the device.

c. The Version field shall comprise:

1. an 8-bit major version number;
2. an 8-bit minor version number;
3. an 8-bit patch or build number; and
4. an 8-bit reserved field.

d. The Version field shall be encoded as shown in Table 5-4.

Table 5-4: Version field

31 — 24	23 — 16	15 — 8	7 — 0
Major Version	Minor Version	Patch/Build Number	Reserved

e. The Version field shall be read-only.

f. The value of the Version field shall be static.

NOTE This means that the value of the field may not change during the normal operation of the device.

5.3.3.2.3 Device Status field

a. The Device Status field shall indicate the operational status of the device.

b. The Device Status field shall comprise an 8-bit device status and a 24-bit reserved field.

c. A device status value of zero shall indicate nominal operation.

d. A device status value of non-zero shall indicate an error failure condition.

e. The meaning of non-zero device status values shall be assigned by the organisation manufacturing or selling the device.

f. The Device Status field shall be encoded as shown in Table 5-5.

Table 5-5: Device Status field

31 … 8	7 … 0
Reserved	Device Status

g. The Device Status field shall be read-only.

5.3.3.2.4 Active Links field

a. The Active Links field shall indicate which links on the device are running.

b. Each bit in the Active Links field shall correspond to the link with the same number as the bit.

NOTE For example, bit 5 shall correspond to link 5.

c. Bit 0 shall be reserved.

d. If a link is running, the corresponding bit shall be set to 0b1.

e. If a link is not running, the corresponding bit shall be cleared to 0b0.

f. The Active Links field shall be read-only.

5.3.3.2.5 Link Information field

a. The Link Information field shall indicate information regarding the number of, and use of, device links.

b. The Link Information field shall include a 5-bit Link Count field indicating the number of links on the device.

c. The Link Information field shall include a 1-bit Node/Routing Switch field indicating if the device is a node or a routing switch.

d. The Node/Routing Switch field shall be set to 0b1 to indicate a routing switch and cleared to 0b0 to indicate a node.

e. The Link Information field shall include a 1-bit Unit Information field indicating if the device supplies valid unit identification information (0b1) or if the device does not supply valid unit identification information (0b0).

NOTE The unit identification information is contained in the Unit Vendor and Product ID field and the Unit Serial Number field.

f. The Link Information field shall include a 5-bit Return Link field indicating the number of the link through which the reply to the current read command will be transmitted.

NOTE As specified in clause 5.2.3.9, a peripheral device always sends replies out of the link on which the command arrived. This field indicates which link is being used to send the reply.

g. The Link Information field shall include a 5-bit Owner Link field indicating the link which was used for the last successful operation to set the value of the Device ID field.

NOTE The control device which sets the Device ID field is the device owner. This field therefore indicates through which link the owner may be reached. Together with the Owner Logical Address and Owner Address fields, this provides sufficient information to be able to identify and contact the owner of a device.

h. On reset the Owner Link field shall contain zero (0b00000).

i. The Link Information field shall include an 8-bit Owner Logical Address field indicating the Reply Logical Address used by the last successful operation to set the value of the Device ID field.

j. On reset the Owner Logical Address field shall be zero (0x00).

k. The Link Information field shall include a 2-bit Owner Address Length field indicating the number of fields being used to represent the Owner Address.

NOTE The Owner Address may be represented using between 0 and 3 fields.

l. The Link Information field shall include two 1-bit reserved fields and a 3-bit reserved field.

m. The Link Information field shall be encoded as shown in Table 5-6, where

1. 'OAL' indicates the Owner Address Length field;
2. 'R' indicates a 1-bit reserved field;
3. 'T' indicates the Node/Routing Switch field; and
4. 'U' indicates the Unit Information field.

Table 5-6: Link Information field

31 … 24	23 22	21	20 … 16	15 … 13	12 … 8	7	6	5	4 … 0
Owner Logical Address	OAL	R	Owner Link	Rsvd	Return Link	T	U	R	Link Count

n. The Link Information field shall be read-only.

5.3.3.2.6 Owner Address fields

a. There shall be three Owner Address fields identified as Owner Address 0, Owner Address 1 and Owner Address 2.

b. The Owner Address fields shall indicate the Reply Address used by the last successful operation to set the value of the Device ID field.

c. The Owner Address fields shall be encoded as specified for the Reply Address field of an RMAP command packet in clause 5.1.6 of ECSS-E-ST-50-52C.

d. If there is no Reply Address, all three Owner Address fields shall be reserved.

e. If the Reply Address is four bytes long, Owner Address 0 shall contain complete Reply Address.

f. If the Reply Address is four bytes long, the Owner Address 1 and Owner Address 2 fields shall be reserved.

g. If the Reply Address is eight bytes long, Owner Address 0 shall contain bytes 0-3 of the Reply Address and Owner Address 1 shall contain bytes 4-7 of the Reply Address.

h. If the Reply Address is eight bytes long, the Owner Address 2 field shall be reserved.

i. If the Reply Address is twelve bytes long, Owner Address 0 shall contain bytes 0-3 of the Reply Address, Owner Address 1 shall contain bytes 4-7 of the Reply Address and Owner Address 2 shall contain bytes 8-11 of the Reply Address.

j. On reset, the value of all three Owner Address fields shall be zero.

k. All three Owner Address fields shall be read-only.

5.3.3.2.7 Device ID field

a. The Device ID field shall be used to assign the device an identifier.

b. The Device ID field shall be a 32-bit field of which all bits may be set.

c. It shall be possible to read the Device ID field using a read operation.

d. It shall be possible to set the Device ID field using a compare-and-swap operation.

e. It shall not be possible to modify the Device ID field using a write operation.

f. When the Device ID field is set, the Owner Link and Owner Logical Address fields of the Link Information field (see clause 5.3.3.2.5), and the Owner Address fields (see clause 5.3.3.2.6) shall be updated to reflect the owner address information.

g. On reset, the Device ID field shall be zero.

h. If the link identified by the Owner Link field of the Link Information field disconnects, the Device ID field shall be reset to zero.

5.3.3.2.8 Unit Vendor and Product ID field

a. If the Unit Information field of the Link Information field indicates the presence of unit information, the Unit Vendor and Product ID field shall identify the vendor and model of the unit.

b. If the Unit Information field of the Link Information field indicates that unit information is not present, the Unit Vendor and Product ID field shall be reserved.

c. The vendor shall be identified by the Vendor ID.

d. The Vendor ID shall be a 16-bit identifier uniquely identifying the organisation manufacturing or selling the unit.

e. The Vendor ID shall be assigned by the SpaceWire Working Group.

f. The Product ID shall be a 16-bit identifier uniquely identifying the unit model for a given organisation manufacturing or selling the unit.

g. The Product ID shall be assigned by the organisation manufacturing or selling the unit.

h. The Unit Vendor and Product ID field shall be encoded in the Unit Vendor and Product ID field as shown in Table 5-7.

Table 5-7: Unit Vendor and Product ID field

31 16	15 0
Vendor ID	Product ID

i. The Unit Vendor and Product ID field shall be read-only.

j. The value of the Unit Vendor and Product ID field shall be static.

NOTE This means that the value of the field may not change during the operation of the device.

5.3.3.2.9 Unit Serial Number field

a. If the Unit Information field of the Link Information field indicates the presence of unit information, the Unit Serial Number field shall be used to assist the unique identification of the unit to which the device belongs.

b. If the Unit Information field of the Link Information field indicates that unit information is not present, the Unit Serial Number field shall be reserved.

c. The Unit Serial Number field shall contain a 32-bit serial number.

d. The serial number shall be assigned to the unit by the organisation identified by the unit vendor ID in the Unit Vendor and Product ID field.

e. The serial number shall be unique within all units which have identical values in the Unit Vendor and Product ID fields.

NOTE Therefore the combination of the values in the Unit Vendor and Product ID field and the Unit Serial Number field uniquely identify the unit.

f. The Unit Serial Number field shall be read-only.

g. The value of the Unit Serial Number ID field shall be static.

NOTE This means that the value of the field may not change during the operation of the device.

5.3.3.2.10 Vendor String Length field

a. The Vendor String Length field shall identify the length of the Vendor String, in bytes.

b. The Vendor String Length field shall comprise a 15-bit string length field and a 17-bit reserved field.

c. The Vendor String Length field shall be encoded as shown in Table 5-8.

Table 5-8: Vendor String Length field

31 … 15	14 … 0
Reserved	Vendor String Length

d. The Vendor String Length field shall be read-only.

e. The value of the Vendor String Length field shall be static.

NOTE This means that the value of the field may not change during the operation of the device.

5.3.3.2.11 Vendor String fields

a. The Vendor String fields shall contain a human-readable string describing the organisation manufacturing or selling the device.

b. The string contained by the Vendor String fields shall be encoded as UTF-8 as specified by Unicode-6.1.

c. The first byte of the string shall be contained in the most significant byte of the first Vendor String field (bits 24-31).

d. Subsequent bytes shall be contained in the next most significant byte, using additional fields as required such that the bytes of the string are arranged contiguously.

e. The maximum string length shall be 32,764 bytes.

f. The minimum string length shall be 0 bytes.

g. The length of the string shall be contained in the Vendor String Length field (see clause 5.3.3.2.8).

h. There shall be 8,191 Vendor String fields.

i. Unused bytes in the Vendor String fields shall be reserved.

j. The Vendor String fields shall be read-only.

k. The value of the Vendor String fields shall be static.

NOTE This means that the value of the fields may not change during the operation of the device.

5.3.3.2.12 Product String Length field

a. The Product String Length field shall identify the length of the Product String, in bytes.

b. The Product String Length field shall comprise a 15-bit string length field and a 17-bit reserved field.

c. The Product String Length field shall be encoded as shown in Table 5-9.

Table 5-9: Product String Length field

31 … 15	14 … 0
Reserved	Product String Length

d. The Product String Length field shall be read-only.

e. The value of the Product String Length field shall be static.

NOTE This means that the value of the field may not change during the operation of the device.

5.3.3.2.13 Product String fields

a. The Product String fields shall contain a human-readable string describing the type of the device.

b. The string contained by the Product String fields shall be encoded as UTF-8 as specified by Unicode-6.1.

c. The first byte of the string shall be contained in the most significant byte of the first Product String field (bits 24-31).

d. Subsequent bytes shall be contained in the next most significant byte, using additional fields as required such that the bytes of the string are arranged contiguously.

e. The maximum string length shall be 32,764 bytes.

f. The minimum string length shall be 0 bytes.

g. The length of the string shall be contained in the Product String Length field (see clause 5.3.3.2.12).

h. There shall be 8,191 Product String fields.

i. Unused bytes in the Product String fields shall be reserved.

j. The Product String fields shall be read-only.

k. The value of the Product String fields shall be static.

NOTE This means that the value of the fields may not change during the operation of the device.

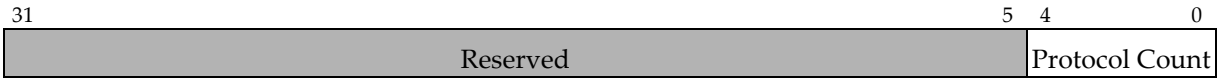
5.3.3.2.14 Protocol Count field

a. The Protocol Count field shall identify the number of protocols supported by the device which may be managed using SpaceWire plug-and-play.

b. The Protocol Count field shall comprise a 5-bit protocol count field and a 27-bit reserved field.

c. The Protocol Count field shall be encoded as shown in Table 5-10.

Table 5-10: Protocol Count field



d. The Protocol Count field shall be read-only.

e. The value of the Protocol Count field shall be static.

NOTE This means that the value of the field may not change during the operation of the device

5.3.3.2.15 Protocol Support fields

a. The Protocol Support fields shall describe the protocols supported by the device which may be managed using SpaceWire plug-and-play.

b. Each supported protocol shall be identified by a single Protocol Identification field.

c. The first supported protocol shall be described by the first Protocol Identification field.

d. Second and subsequent supported protocols shall be described by the second and subsequent Protocol Identification fields such that all supported protocols are described by contiguous fields.

e. The Protocol Identification field shall comprise a 16-bit Vendor ID and a 16-bit Protocol ID.

f. The Protocol Identification field shall be encoded as shown in Table 5-11.

Table 5-11: Protocol Identification field

31 — 16	15 — 0
Vendor ID	Protocol ID

g. If the Vendor ID is non-zero then it shall identify the organisation responsible for standardising the protocol.

h. Non-zero Vendor IDs shall be the same as those used in the Vendor ID field of the Vendor and Product ID field, see clause 5.3.3.2.1.

i. If the Vendor ID is non-zero then the meaning of the Protocol ID shall be assigned by the organisation identified by the Vendor ID.

j. If the Vendor ID is zero then the Protocol ID shall identify a SpaceWire protocol standardised by ECSS in ECSS-E-ST-50-51.

k. There shall be 31 Protocol Identification fields.

l. Unused Protocol Identification fields shall be reserved.

m. Protocol Support fields shall be read-only.

n. The value of Protocol Support fields shall be static.

NOTE This means that the value of the fields may not change during the operation of the device

5.3.3.2.16 Application Count field

a. The Application Count field shall identify the number of applications supported by the device which may be managed using SpaceWire plug-and-play.

b. The Application Count field shall comprise an 8-bit application count field and a 24-bit reserved field.

c. The Application Count field shall be encoded as shown in Table 5-12.

Table 5-12: Application Count field

31 … 8	7 … 0
Reserved	Application Count

d. The Application Count field shall be read-only.

e. The value of the ApplicationCount field shall be static.

NOTE This means that the value of the field may not change during the operation of the device

5.3.3.2.17 Application Support fields

a. The Application Support fields shall describe the applications supported by the device which may be managed using SpaceWire plug-and-play.

b. Each supported application shall be identified by an Application Identification field and a Protocol Use field.

c. The first supported application shall be described by the first Application Identification field and the first Protocol Use field.

d. Second and subsequent supported applications shall be described by the second and subsequent Application Identification and Protocol Use fields such that all supported applications are described by contiguous fields.

e. The Application Identification field shall comprise a 16-bit Vendor ID and a 16-bit Application ID.

f. The Application Identification field shall be encoded as shown in Table 5-13.

Table 5-13: Application Identification field

31 … 16	15 … 0
Vendor ID	Application ID

g. If the Vendor ID is non-zero then it shall identify the organisation responsible for standardising the application.

h. Non-zero Vendor IDs shall be the same as those used in the Vendor ID field of the Vendor and Product ID field, see clause 5.3.3.2.1.

i. If the Vendor ID is non-zero then the meaning of the Application ID shall be assigned by the organisation identified by the Vendor ID.

j. If the Vendor ID is zero then the Application ID shall identify a SpaceWire application standardised by ECSS.

k. There shall be 255 Application Identification fields

l. Unused Application Identification fields shall be reserved.

m. The Protocol Use field shall identify the protocols used by the associated application.

n. Each bit in the Protocol Use field shall correspond to the Protocol Identification field with the same index as the bit number.

NOTE For example, bit number 5 shall correspond to the protocol identified by the 5th Protocol Identification field.

o. Bit zero of the Protocol Use field shall be reserved.

p. There shall be 255 Protocol Use fields

q. Unused Protocol Use fields shall be reserved.

r. Application Support fields shall be read-only.

s. The value of Application Support fields shall be static.

NOTE This means that the value of the fields may not change during the operation of the device

5.3.3.3 Field identification

5.3.3.3.1 Device Information provision

a. All peripheral devices shall provide Device Information fields.

5.3.3.3.2 Protocol and application indices

a. Device Information fields shall be identified by using a Protocol_Index parameter of zero and an Application_Index parameter of zero when applying a read, write or read-modify-write operation.

5.3.3.3.3 Field sets

a. Device Information fields shall be grouped into four field sets, as follows:

1. the Device Identification field set;
2. the Vendor/Product String field set;
3. the Protocol Support field set; and
4. the Application Support field set.

b. Device Information field sets shall be identified as shown in Table 5-14.

Table 5-14: Device Information field sets

Field Set Identifier	Field Set
0	Device Identification
1	Vendor/Product String
2	Protocol Support
3	Application Support

5.3.3.3.4 Device Identification field set

a. The Device Identification field set shall comprise the following fields:

1. the Device Vendor and Product ID field;
2. the Version field;
3. the Device Status field;

4. the Link Information field;
5. the Owner Address fields;
6. the Device ID field;
7. the Unit Vendor and Product ID field; and
8. the Unit Serial Number field.

b. Device Identification field set fields shall be identified as shown in Table 5-15.

Table 5-15: Device Identification field set fields

Field Identifier	Field
0	Device Vendor and Product ID
1	Version
2	Device Status
3	Active Links
4	Link Information
5	Owner Address 0
6	Owner Address 1
7	Owner Address 2
8	Device ID
9	Unit Vendor and Product ID
10	Unit Serial Number

5.3.3.3.5 Vendor/Product Strings field set

a. The Vendor/Product Strings field set shall comprise the following fields:
1. the Vendor String Length field;
2. the Vendor String fields;
3. the Product String Length field; and
4. the Product String fields.

b. Vendor/Product Strings field set fields shall be identified as shown in Table 5-16.

Table 5-16: Vendor/Product Strings field set fields

Field Identifier	Field
0	Vendor String Length
1 – 8,191	Vendor String
8,192	Product String Length
8,193 – 16,383	Product String

5.3.3.3.6 Protocol Support field set

a. The Protocol Support field set shall comprise the following fields:

   1. the Protocol Count field; and

   2. the Protocol Support fields.

b. Protocol Support field set fields shall be identified as shown in Table 5-17.

Table 5-17: Protocol Support field set fields

Field Identifier	Field
0	Protocol Count
Protocol Index (1 – 31)	Protocol Identification

5.3.3.3.7 Application Support field set

a. The Application Support field set shall comprise the following fields:

   1. the Application Count field; and

   2. the Application Support fields.

b. Application Support field set fields shall be identified as shown in Table 5-18.

Table 5-18: Application Support field set fields

Field Identifier	Field
0	Application Count
([Application Index (1 – 255)] x 2)	Application Identification
([Application Index (1 – 255)] x 2) + 1	Protocol Use

5.3.4 SpaceWire protocol

5.3.4.1 Overview

The SpaceWire protocol fields provide the capability to manage SpaceWire devices in a standard way.

5.3.4.2 Fields

5.3.4.2.1 Time-Code Counter field

a. The Time-Code Counter field shall provide the value of the device time-code counter, as defined by clause 8.12 of ECSS-E-ST-50-12C, and the ability to reset the counter.

b. The Time-Code Counter field shall comprise a 6-bit field containing the current value of the device time-code counter and a 26-bit reserved field.

c. The Time-Code Counter field shall be encoded as shown in Table 5-19.

Table 5-19: Time-Code Counter field

31 … 6	5 … 0
Reserved	Time-Code

d. Writing a value of zero to the time-code counter field shall reset the value of the time-code counter to zero.

e. Writing any other value to the Time-Code Counter field shall be ignored.

5.3.4.2.2 Base Transmit Rate field

a. The Base Transmit Rate field shall indicate the basic frequency from which Run state link transmit rates are derived.

b. The Base Transmit Rate field shall contain a 16-bit field specifying transmit rate measured in Megabits per second (Mbps).

c. The Base Transmit Rate field shall contain a 16-bit reserved field.

d. The Base Transmit Rate field shall be encoded as shown in Table 5-20.

Table 5-20: Base Transmit Rate field

31 … 16	15 … 0
Reserved	Base Transmit Rate

e. The Base Transmit Rate field shall be read-only.

f. If the determination of link transmit rates is not supported, the Base Transmit Rate field shall be reserved.

NOTE A Base Transmit Rate value of zero indicates that neither the determination nor the configuration of link transmit rates is supported.

5.3.4.2.3 Reference Transmit Rate Range field

a. The Reference Transmit Rate Range field shall identify the maximum and minimum valid values for the Reference Transmit Rate Divider field (see clause 5.3.4.2.4).

b. The Reference Transmit Rate Range field shall include a 16-bit field specifying the maximum valid value for the Reference Transmit Rate Divider field (see clause 5.3.4.2.4).

c. The Reference Transmit Rate Range field shall include a 16-bit field specifying the minimum valid value for the Reference Transmit Rate Divider field (see clause 5.3.4.2.4).

d. The Reference Transmit Rate Range field shall be encoded as shown in Table 5-21.

Table 5-21: Reference Transmit Rate Range field

31 … 16	15 … 0
Maximum Reference Transmit Divider	Minimum Reference Transmit Divider

e. The Reference Transmit Rate Range field shall be read-only.

f. If determination of link transmit rates is not supported, the Reference Transmit Rate Range field shall be reserved.

NOTE If an implementation is able to support the determination of the reference transmit rate, but not configuration, this may be conveyed by setting the minimum and maximum reference transmit dividers to the same value.

5.3.4.2.4 Reference Transmit Rate Divider field

a. The Reference Transmit Rate Divider field shall define the value by which the Base Transmit Rate (see clause 5.3.4.2.2) should be divided to generate the reference transmit rate.

b. The Reference Transmit Rate Divider field shall include a 16-bit field containing the reference transmit rate divider value.

c. The Reference Transmit Rate Divider field shall include a 16-bit reserved field.

d. The Reference Transmit Rate Divider field shall be encoded as shown in Table 5-22.

Table 5-22: Reference Transmit Rate Divider field

31 … 16	15 … 0
Reserved	Reference Transmit Rate Divider

e. If a value lower than the permitted minimum value is written, the Reference Transmit Rate Divider shall be set to the minimum value.

f. If a value higher than the permitted maximum value is written, the Reference Transmit Rate Divider shall be set to the maximum value.

g. If a value is written which is not supported by the implementation, the Reference Transmit Rate Divider shall be set to the highest supported value below the written value.

h. If determination of link transmit rates is not supported, the Reference Transmit Rate Divider field shall be reserved.

5.3.4.2.5 Base Watchdog Rate field

a. The Base Watchdog Rate field shall indicate the basic frequency from which link timeout watchdog rates are derived.

b. The Base Watchdog Rate field shall contain a 16-bit field specifying the watchdog rate measured in kilo Hertz (kHz).

c. The Base Watchdog Rate field shall contain a 16-bit reserved field.

d. The Base Watchdog Rate field shall be encoded as shown in Table 5-23.

Table 5-23: Base Watchdog Rate field

31 … 16	15 … 0
Reserved	Base Watchdog Rate

e. The Base Watchdog Rate field shall be read-only.

f. If determination of link watchdog rates is not supported, the Base Watchdog Rate field shall be reserved.

NOTE A Base Watchdog Rate value of zero indicates that neither the determination nor configuration of link watchdog rates is supported.

5.3.4.2.6 Minimum Reference Watchdog Divider field

a. The Minimum Reference Watchdog Divider field shall identify the minimum valid value for the Reference Watchdog Rate Divider field (see clause 5.3.4.2.8).

b. The Minimum Reference Watchdog Rate field shall comprise a 32-bit field specifying the minimum valid value for the Reference Watchdog Rate Divider field (see clause 5.3.4.2.8).

c. The Minimum Reference Watchdog Rate field shall be read-only.

d. If determination of link watchdog rates is not supported, the Minimum Reference Watchdog Divider field shall be reserved.

NOTE If an implementation is able to support the determination of the reference watchdog rate, but not configuration, this may be conveyed by setting the minimum and maximum reference watchdog divider fields to the same value.

5.3.4.2.7 Maximum Reference Watchdog Divider field

a. The Maximum Reference Watchdog Divider field shall identify the maximum valid value for the Reference Watchdog Rate Divider field (see clause 5.3.4.2.8).

b. The Maximum Reference Watchdog Rate field shall comprise a 32-bit field specifying the maximum valid value for the Reference Watchdog Rate Divider field (see clause 5.3.4.2.8).

c. The Maximum Reference Watchdog Rate field shall be read-only.

d. If determination of link watchdog rates is not supported, the Maximum Reference Watchdog Divider field shall be reserved.

NOTE If an implementation is able to support the determination of the reference watchdog rate, but not configuration, this may be conveyed by setting the minimum and maximum reference watchdog divider fields to the same value.

5.3.4.2.8 Reference Watchdog Rate Divider field

a. The Reference Watchdog Rate Divider field shall define the value by which the Base Watchdog Rate (see clause 5.3.4.2.5) should be divided to generate the reference watchdog rate.

b. The Reference Watchdog Rate Divider field shall comprise a 32-bit field containing the reference watchdog rate divider value.

c. If a value lower than the permitted minimum value is written, the Reference Watchdog Rate Divider shall be set to the minimum value.

d. If a value higher than the permitted maximum value is written, the Reference Watchdog Rate Divider shall be set to the maximum value.

e. If a value is written which is not supported by the implementation, the Reference Watchdog Rate Divider shall be set to the highest supported value below the written value.

f. If determination of link watchdog rates is not supported, the Reference Watchdog Rate Divider field shall be reserved.

5.3.4.2.9 Link Status fields

a. A Link Status field shall describe the type and status of a link.

b. A Link Status field shall include a 1-bit field indicating if the link should be used for network discovery (0b1) or ignored during network discovery (0b0).

c. A Link Status field shall include a 1-bit field indicating if the link is a SpaceWire link (0b1) or any other type of link (0b0).

d. A Link Status field shall include a 3-bit field indicating the state of the link encoded as shown in Table 5-24, with states as defined by clause 8.5 of ECSS-E-ST-50-12C.

Table 5-24: Link state encoding

Link State Value	Link State
0b000	Error Reset
0b001	Error Wait
0b010	Ready
0b011	Started
0b100	Connecting
0b101	Run
0b110	Reserved
0b111	Reserved

e. If a link is not a SpaceWire link then the only values that shall be used by the corresponding link state in the Link Status field are 0b000 to indicate that the link is not connected and 0b101 to indicate that the link is running

f. A Link Status field shall include a 1-bit field indicating if a character sequence error (as defined in clause 8.9.2.5 of ECSS-E-ST-50-12C) has occurred (0b1) or if no character sequence error has occurred (0b0).

g. A Link Status field shall include a 1-bit field indicating if a credit error (as defined in clause 8.9.2.4 of ECSS-E-ST-50-12C) has occurred on the link (0b1) or if no credit error has occurred (0b0).

h. A Link Status field shall include a 1-bit field indicating if an escape error (as defined in clause 8.9.2.3 of ECSS-E-ST-50-12C) has occurred on the link (0b1) or if no escape error has occurred (0b0).

i. A Link Status field shall include a 1-bit field indicating if a parity error (as defined in clause 8.9.2.2 of ECSS-E-ST-50-12C) has occurred on the link (0b1) or if no parity error has occurred (0b0).

j. A Link Status field shall include a 1-bit field indicating if a disconnect error (as defined in clause 8.9.2.1 of ECSS-E-ST-50-12C) has occurred on the link (0b1) or if no disconnect error has occurred (0b0).

k. A Link Status field shall include a 1-bit field indicating if a watchdog time-out error has occurred on the link (0b1) or if no watchdog time-out error has occurred (0b0).

l. A Link Status field shall include a 1-bit field indicating if a packet address error (as defined in clause 10.5.4 of ECSS-E-ST-50-12C) has occurred on the link (0b1) or if no packet address error has occurred (0b0).

m. A Link Status field shall include an 11-bit reserved field, an 8-bit reserved field and a 1-bit reserved field.

n. If an error is not supported, or cannot be generated, by the underlying link implementation then the corresponding bit of the Link Status field shall always be clear.

o. Link Status fields shall be encoded as shown in Table 5-25, where the following abbreviations are used:

1. ‘N’ indicates the network discovery bit;
2. ‘T’ indicates the link type bit;
3. ‘S’ indicates the character sequence error bit;
4. ‘C’ indicates the credit error bit;
5. ‘E’ indicates the escape error bit;
6. ‘P’ indicates the parity error bit;
7. ‘D’ indicates the disconnect error bit;
8. ‘W’ indicates the watchdog time-out error bit;
9. ‘A’ indicates the packet address error bit; and
10. ‘R’ indicates the reserved bit.

Table 5-25: Link Status field

31	30	29 … 19	18 … 16	15 … 8	7	6	5	4	3	2	1	0
N	T	Reserved	State	Reserved	S	C	E	P	D	W	A	R

p. On reset, the bits corresponding to errors reported by a Link status field shall be clear (0b0).

q. Once any error reported by a Link Status field occurs, the corresponding bit shall remain set until cleared or the device is reset.

r. Invoking a write or compare-and-swap operation to set the value of a Link Status field to zero shall result in the error bits being cleared.

s. Operations attempting to set any other value in a Link Status field shall be ignored.

5.3.4.2.10 Link Control fields

a. A Link Control field shall permit control over the state of a link.

b. A Link Control field shall include a 1-bit field to specify if the link should be automatically started if there is a packet waiting to be transmitted (0b1) or if the link should not be automatically started if there is a packet waiting to be transmitted.

NOTE This function is known as 'start-on-request'.

c. If the start-on-request function is not implemented for the link then the start-on-request bit shall be reserved.

d. A Link Control field shall include a 1-bit field to specify if time-codes may be transmitted (as defined in clause 8.12 of ECSS-E-ST-50-12C) over the link (0b1) or if time-codes may not be transmitted over the link (0b0).

e. A Link Control field shall include a 1-bit field to specify if the watchdog time-out should be applied to packets being transmitted or received on the link (0b1) or if the watchdog timeout should not be applied (0b0).

f. If watchdog timeouts are not implemented for the link then the watchdog timeout bit shall be reserved.

g. A Link Control field shall include a 1-bit field to specify the value of the LinkDisabled flag as specified in clause 8.6 of ECSS-E-ST-50-12C.

h. A Link Control field shall include a 1-bit field to specify the value of the LinkStart flag as specified in clause 8.6 of ECSS-E-ST-50-12C.

i. A Link Control field shall include a 1-bit field to specify the value of the AutoStart flag as specified in clause 8.6 of ECSS-E-ST-50-12C.

j. A Link Control field shall include a 27-bit reserved field.

k. Link Control fields shall be encoded as shown in Table 5-26, where the following abbreviations are used:

1. 'R' indicates the start-on-request bit;
2. 'T' indicates the time-code transmission bit;
3. 'W' indicates the watchdog time-out bit;
4. 'D' indicates the LinkDisabled flag bit;
5. 'S' indicates the LinkStart flag bit;
6. 'A' indicates the AutoStart flag bit.

Table 5-26: Link Control field

31 … 6	5	4	3	2	1	0
Reserved	R	T	W	D	S	A

5.3.4.2.11 Link Debug Information fields

a. A Link Debug Information field shall contain debugging information about a link.

b. A Link Debug Information field shall include a 1-bit field to indicate if the receive buffer empty flag is supported (0b1) or if the receive buffer empty flag is not supported (0b0).

c. A Link Debug Information field shall include a 1-bit field to indicate if the receive credit count field is supported (0b1) or if the receive credit count field is not supported (0b0).

d. If a Link Debug Information field indicates that the receive buffer empty flag is supported, the Link Debug Information field shall include a 1-bit field indicating if the receive buffer (as defined in clause 8.3 of ECSS-E-ST-50-12C) is empty (0b1) or if the receive buffer is non-empty (0b0).

e. If a Link Debug Information field indicates that the receive buffer empty flag is not supported, the 1-bit receive buffer empty field shall be reserved.

f. If a Link Debug Information field indicates that the receive credit count is supported, the Link Debug Information field shall include a 5-bit field containing the current receive credit count as defined in clause 8.3 of ECSS-E-ST-50-12C.

g. If a Link Debug Information field indicates that the receive credit count is not supported, the 5-bit receive credit count field shall be reserved.

h. A Link Debug Information field shall include a 1-bit field to indicate if the transmit buffer empty full is supported (0b1) or if the transmit buffer full flag is not supported (0b0).

i. A Link Debug Information field shall include a 1-bit field to indicate if the transmit credit count field is supported (0b1) or if the transmit credit count field is not supported (0b0).

j. If a Link Debug Information field indicates that the transmit buffer full flag is supported, the Link Debug Information field shall include a 1-bit field indicating if the transmit buffer (as defined in clause 8.3 of ECSS-E-ST-50-12C) is full (0b1) or if the transmit buffer is non-full (0b0).

k. If a Link Debug Information field indicates that the transmit buffer full flag is not supported, the 1-bit transmit buffer full field shall be reserved.

l. If a Link Debug Information field indicates that the transmit credit count is supported, the Link Debug Information field shall include a 5-bit field containing the current transmit credit count as defined in clause 8.3 of ECSS-E-ST-50-12C.

m. If a Link Debug Information field indicates that the transmit credit count is not supported, the 5-bit transmit credit count field shall be reserved.

n. A Link Debug Information field shall include a 16-bit reserved field.

o. Link Debug Information fields shall be encoded as shown in Table 5-27, where the following abbreviations are used:

   1. ‘A’ indicates the receive buffer empty flag support;

2. ‘R’ indicates the receive credit count support;
3. ‘E’ contains the receive buffer empty flag;
4. ‘Rx Credit’ contains the receive credit count;
5. ‘B’ indicates the transmit buffer full flag support;
6. ‘T’ indicates the transmit credit count support;
7. ‘F’ contains the transmit buffer full flag; and
8. ‘Tx Credit’ contains the transmit credit count.

Table 5-27: Link Debug Information field

31 … 16	15	14	13	12 … 8	7	6	5	4 … 0
Reserved	A	R	E	Rx Credit	B	T	F	Tx Credit

p. Link Debug Information fields shall be read-only.

5.3.4.2.12 Link Transmit Rate Range fields

a. A Link Transmit Rate Range field shall identify the maximum and minimum valid values for the corresponding Link Transmit Rate Divider field (see clause 5.3.4.2.13).

b. A Link Transmit Rate Range field shall include a 16-bit field specifying the maximum valid value for the corresponding Link Transmit Rate Divider field (see clause 5.3.4.2.13).

c. A Link Transmit Rate Range field shall include a 16-bit field specifying the minimum valid value for the corresponding Link Transmit Rate Divider field (see clause 5.3.4.2.13).

d. Link Transmit Rate Range fields shall be encoded as shown in Table 5-28.

Table 5-28: Link Transmit Rate Range field

31 … 16	15 … 0
Maximum Transmit Divider	Minimum Transmit Divider

e. Link Transmit Rate Range fields shall be read-only.

f. If determination of link transmit rates is not supported, Link Transmit Rate Range fields shall be reserved.

NOTE If an implementation is able to support the determination of the link transmit rate, but not configuration, this may be conveyed by setting the minimum and maximum link transmit dividers to the same value.

5.3.4.2.13 Link Transmit Rate Divider fields

a. A Link Transmit Rate Divider field shall define the value by which the reference transmit rate (see clause 5.3.4.2.4) should be divided to generate the transmit rate for a link when it is in the Run state.

NOTE 1 The transmit rate for a link may therefore be calculated as the Base Transmit Rate divided by

the Reference Transmit Rate Divider (to generate the reference transmit rate) divided by the relevant Link Transmit Rate Divider.

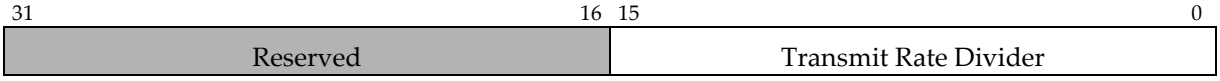
NOTE 2 The link transmit rate only applies when the link is in the Run state. No provision is made to configure the link startup rate.

b. A Link Transmit Rate Divider field shall include a 16-bit field containing the transmit rate divider value.

c. A Link Transmit Rate Divider field shall include a 16-bit reserved field.

d. Link Transmit Rate Divider fields shall be encoded as shown in Table 5-29.

Table 5-29: Link Transmit Rate Divider field



e. If a value lower than the permitted minimum value is written, the Link Transmit Rate Divider shall be set to the minimum value.

f. If a value higher than the permitted maximum value is written, the Link Transmit Rate Divider shall be set to the maximum value.

g. If a value is written which is not supported by the implementation, the Link Transmit Rate Divider shall be set to the highest supported value below the written value.

h. If determination of link transmit rates is not supported, Link Transmit Rate Divider fields shall be reserved.

5.3.4.2.14 Minimum Watchdog Divider fields

a. A Link Minimum Watchdog Divider field shall identify the minimum valid value for the corresponding Link Watchdog Rate Divider field (see clause 5.3.4.2.16).

b. A Link Minimum Watchdog Rate field shall comprise a 32-bit field specifying the minimum valid value for the corresponding Link Watchdog Rate Divider field (see clause 5.3.4.2.16).

c. Link Minimum Watchdog Rates field shall be read-only.

d. If determination of link watchdog rates is not supported, Link Minimum Watchdog Divider fields shall be reserved.

NOTE If an implementation is able to support the determination of the link watchdog rate, but not configuration, this may be conveyed by setting the minimum and maximum link watchdog divider fields to the same value.

5.3.4.2.15 Maximum Watchdog Divider fields

a. A Link Maximum Watchdog Divider field shall identify the maximum valid value for the corresponding Link Watchdog Rate Divider field (see clause 5.3.4.2.16).

b. A Link Maximum Watchdog Rate field shall comprise a 32-bit field specifying the maximum valid value for the corresponding Link Watchdog Rate Divider field (see clause 5.3.4.2.16).

c. Link Maximum Watchdog Rate fields shall be read-only.

d. If determination of link watchdog rates is not supported, Link Maximum Watchdog Divider fields shall be reserved.

NOTE If an implementation is able to support the determination of the link watchdog rate, but not configuration, this may be conveyed by setting the minimum and maximum link watchdog divider fields to the same value.

5.3.4.2.16 Link Watchdog Rate Divider fields

a. A Link Watchdog Rate Divider field shall define the value by which the reference watchdog rate (see clause 5.3.4.2.8) should be divided to generate the link watchdog rate.

NOTE 1 The watchdog rate for a link may therefore be calculated as the Base Watchdog Rate divided by the Reference Watchdog Rate Divider (to generate the reference watchdog rate) divided by the relevant Link Watchdog Rate Divider.

NOTE 2 The watchdog period for a link is the reciprocal of the watchdog rate.

b. A Link Watchdog Rate Divider field shall comprise a 32-bit field containing the watchdog rate divider value.

c. If a value lower than the permitted minimum value is written, the Link Watchdog Rate Divider shall be set to the minimum value.

d. If a value higher than the permitted maximum value is written, the Link Watchdog Rate Divider shall be set to the maximum value.

e. If a value is written which is not supported by the implementation, the Link Watchdog Rate Divider shall be set to the highest supported value below the written value.

f. If determination of link watchdog rates is not supported, Link Watchdog Rate Divider fields shall be reserved.

5.3.4.2.17 Routing Control field

a. The Routing Control field shall permit control of the routing behaviour of a routing switch.

b. The Routing Control field shall include a 1-bit field to determine if self-addressing is permitted (0b1) or not permitted (0b0).

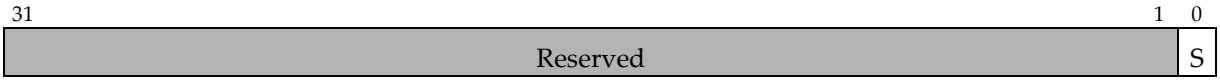
NOTE Self-addressing is where the routing of an incoming packet attempts to select the port at which the packet was received as the transmission port either by path or logical addressing.

c. The Routing Control field shall include a 31-bit reserved field.

d. The Routing Control field shall be encoded as shown in Table 5-30, where the following abbreviations are used:

1. 'S' indicates the self-addressing bit.

Table 5-30: Routing Control field



5.3.4.2.18 Port Association fields

a. A Port Association field shall define the routing switch ports associated with a SpaceWire address.

b. Each bit of a Port Association field shall correspond to a routing switch port of the same number as the bit number.

NOTE For example, bit 5 corresponds to port 5.

c. Bit 0 of a Port Association field shall be reserved.

d. The value of a bit of a Port Association field shall indicate the association between the address and the corresponding port with:

1. a set bit (0b1) indicating that the port is to be used for routing the address; and

2. a clear bit (0b0) indicating that the port is not to be used for routing the address.

e. If the implementation supports port groups for the address it shall be possible to set multiple bits in a Port Association field.

f. It shall be permissible for an implementation to fix the value of one or more bits in a Port Association field.

NOTE It is likely that an implementation will want to fix the value of a Port Association field associated with a path address such that the association always includes the port with the same numeric value as the path address. This behaviour is required by the SpaceWire standard, see clause 10.2.3 of ECSS-E-ST-50-12C.

5.3.4.2.19 Address Control fields

a. An Address Control field shall define the operation of a SpaceWire address during routing.

b. An Address Control field shall include an 8-bit field specifying the arbitration priority (see clause 10.2.5 of ECSS-E-ST-50-12C) associated with the address.

c. If the implementation supports fewer than 8-bits to specify the arbitration priority then the bits supported by the implementation shall be aligned

with the most significant bit of the 8-bit arbitration priority field of the Address Control fields with the remaining bits being reserved.

d. If the implementation does not support arbitration priority then the 8-bit arbitration priority field of the Address Control fields shall be reserved.

e. An Address Control field shall include a 1-bit field specifying the action to be taken during routing of packets with the address when a group of ports are defined in the corresponding Port Association field, with:

   1. the field being set (0b1) to indicate group adaptive routing (see clause 10.2.6 of ECSS-E-ST-50-12C); and

   2. the field being clear (0b0) to indicate packet distribution (see clause 10.2.7 of ECSS-E-ST-50-12C).

f. An Address Control field shall include a 1-bit field specifying if header deletion (see clause 10.2.3 of ECSS-E-ST-50-12C) will be applied to packets being routed with the address, with:

   1. the field being set (0b1) to indicate that header deletion will be applied; and

   2. the field being clear (0b0) to indicate that header deletion will not be applied.

g. If the implementation supports group adaptive routing but not packet distribution then the group action bit of the Address Control fields shall be:

   1. set (0b1); and

   2. read-only.

h. If the implementation supports packet distribution but not group adaptive routing then the group action bit of the Address Control fields shall be:

   1. clear (0b0); and

   2. read-only.

i. If the implementation supports neither packet distribution nor group adaptive routing then the group action bit of the Address Control fields shall be:

   1. set (0b1); and

   2. read-only.

j. An Address Control field shall include a 1-bit field indicating if the address is enabled for routing, with:

   1. the field being set (0b1) to indicate that the address is enabled for routing; and

   2. the field being clear (0b0) to indicate that the address is not enabled for routing and all packets arriving at the routing switch with the address should be discarded and a packet address error generated.

k. Address Control fields shall be encoded as shown in Table 5-31, where the following abbreviations are used:

1. 'G' indicates the group action bit;
2. 'D' indicates the header deletion bit; and
3. 'E' indicates the address enabled bit.

Table 5-31: Address Control fields

31 … 16	15 … 8	7 … 3	2	1	0
Reserved	Arbitration Priority	Reserved	G	D	E

5.3.4.2.20 Time-Code Generation Control field

a. The Time-Code Generation Control field shall control the generation of time-codes by the device.

NOTE This field is part of the Time-Code Generation field set (see clause 5.3.4.3.7) which can be provided by a device wishing to offer a time-code generation function using the plug-and-play protocol. Provision of the Time-Code Generation field set is not mandatory.

b. The Time-Code Generation Control field shall include a 6-bit time-code value field containing the value of the next time-code to be generated.

c. The Time-Code Generation Control field shall include a 1-bit field which, if set to 0b1 on a write (or compare-and-swap), causes the write (or compare-and-swap) to set the value of the time-code value field or if set to 0b0 causes the write (or compare-and-swap) to leave the time-code value field unaffected.

d. The Time-Code Generation Control field shall include a 1-bit field to determine if time-codes should be generated periodically (0b1) or if time-codes should not be generated periodically (0b0).

e. If Time-Code Generation Control field is set to generate time-codes periodically then the device shall generate time-codes according to the period specified in the Time-Code Generation Period field (see clause 5.3.4.2.21).

f. The Time-Code Generation Control field shall include a 1-bit field which always reads as 0b0 and if written with 0b1 causes the immediate generation of a time-code.

g. The immediate time-code generation bit (see clause 5.3.4.2.20.f) of the Time-Code Generation Control field shall be ignored if periodic time-code generation is enabled (see clause 5.3.4.2.20.d).

h. If a write (or compare-and-swap) operation specifies that a time-code should be generated immediately (see clause 5.3.4.2.20.f) and the time-code value is not specified as part of the same write (or compare-and-swap) operation (see clauses 5.3.4.2.20.b and c), the value of the time-code shall be the next consecutive time-code.

i. If a write (or compare-and-swap) operation specifies that a time-code should be generated immediately (see clause 5.3.4.2.20.f) and the time-code value is specified as part of the same write (or compare-and-swap) operation (see clauses 5.3.4.2.20.b and c), the value of the time-code shall be the specified value.

j. The Time-Code Generation Control field shall include a 21-bit reserved field.

k. The Time-Code Generation Control field shall include a 2-bit reserved field.

l. The Time-Code Generation Control field shall be encoded as shown in Table 5-32, where the following abbreviations are used:

1. 'V' indicates the set time-code value bit;
2. 'I' indicates the immediate time-code generation bit; and
3. 'P' indicates the periodic time-code generation enabled bit.

Table 5-32: Time-Code Generation Control field

31 … 11	10	9	8	7 … 6	5 … 0
Reserved	P	I	V	Rsvd	Time-Code Value

5.3.4.2.21 Time-Code Generation Period field

a. The Time-Code Generation Period field shall define the period between time-codes generated by the device when periodic time-code generation is enabled in the Time-Code Generation Control field (see clause 5.3.4.2.20).

NOTE This field is part of the Time-Code Generation field set (see clause 5.3.4.3.7) which can be provided by a device wishing to offer a time-code generation function using the plug-and-play protocol. Provision of the Time-Code Generation field set is not mandatory.

b. The Time-Code Generation Period field shall comprise a 32-bit field containing the time-code generation period in microseconds.

c. If a value lower than the permitted minimum value is written, the Time-Code Generation Period shall be set to the minimum value.

d. If a value higher than the permitted maximum value is written, the Time-Code Generation Period shall be set to the maximum value.

e. If a value is written which is not supported by the implementation, the Time-Code Generation Period shall be set to the highest supported value below the written value.

5.3.4.3 Field identification

5.3.4.3.1 Protocol identifier

a. Support for the SpaceWire Protocol fields shall be indicated by specifying a Vendor ID of zero and a protocol ID of zero in the protocol support list (see clause 5.3.3.3.6.).

5.3.4.3.2 Protocol and application indices

a. SpaceWire Protocol fields shall be identified by using an Application_Index parameter of zero and a Protocol_Index corresponding to the index of the entry in the protocol support list specifying support for the SpaceWire Protocol fields (see clause 5.3.4.3.1) when applying a read, write or read-modify-write operation.

5.3.4.3.3 Field sets

a. SpaceWire Protocol fields shall be grouped into four field sets, as follows:
   1. the Device Configuration field set;
   2. the Link Configuration field set;
   3. the Routing Table field set; and
   4. the Time-Code Generation field set.

b. SpaceWire Protocol field sets shall be identified as shown in Table 5-33.

Table 5-33: SpaceWire Protocol field sets

Field Set Identifier	Field Set
0	Device Configuration
1	Link Configuration
2	Routing Table
3	Time-Code Generation

5.3.4.3.4 Device Configuration field set

a. The Device Configuration field set shall comprise the following fields:
   1. the Time-Code Counter field;
   2. the Base Transmit Rate field;
   3. the Reference Transmit Rate Range field;
   4. the Reference Transmit Rate Divider field;
   5. the Base Watchdog Rate field;
   6. the Minimum Reference Watchdog Divider field;
   7. the Maximum Reference Watchdog Divider field; and
   8. the Reference Watchdog Rate Divider field.

b. Device Identification field set fields shall be identified as shown in Table 5-34.

Table 5-34: Device Configuration field set fields

Field Identifier	Field
0	Time-Code Counter
1	Base Transmit Rate
2	Reference Transmit Rate Range
3	Reference Transmit Rate Divider
4	Base Watchdog Rate
5	Minimum Reference Watchdog Divider
6	Maximum Reference Watchdog Divider
7	Reference Watchdog Rate Divider

5.3.4.3.5 Link Configuration field set

a. The Link Configuration field set shall include one of each of the following fields for each link implemented on the device:

1. the Link Status field;
2. the Link Control field;
3. the Link Debug Information field;
4. the Link Transmit Rate Range field;
5. the Link Transmit Rate Divider field;
6. the Link Minimum Watchdog Divider field;
7. the Link Maximum Watchdog Divider field; and
8. the Link Watchdog Rate Divider field.

b. Link Configuration field set fields shall be identified as shown in Table 5-35.

Table 5-35: Link Configuration field set fields

Field Identifier	Field
0 – 7	Reserved
([Link Number (1 – 31)] x 8)	Link Status
([Link Number (1 – 31)] x 8) + 1	Link Control
([Link Number (1 – 31)] x 8) + 2	Link Debug Information
([Link Number (1 – 31)] x 8) + 3	Link Transmit Rate Range
([Link Number (1 – 31)] x 8) + 4	Link Transmit Rate Divider
([Link Number (1 – 31)] x 8) + 5	Link Minimum Watchdog Divider
([Link Number (1 – 31)] x 8) + 6	Link Maximum Watchdog Divider
([Link Number (1 – 31)] x 8) + 7	Link Watchdog Rate Divider

5.3.4.3.6 Routing Table field set

a. Nodes shall not provide the Routing Table field set.

b. The Routing Table field set shall include the Routing Control field.

c. The Routing Table field set shall include one of each of the following fields for each address implemented on the device:

1. the Port Association field; and

2. the Address Control field.

d. Routing Table field set fields shall be identified as shown in Table 5-36.

Table 5-36: Routing Table field set fields

Field Identifier	Field
0	Routing Control
1	Reserved
([Address (1 – 255)] x 2)	Port Association
([Address (1 – 255)] x 2) + 1	Address Control

5.3.4.3.7 Time-Code Generation field set

a. The Time-Code Generation field set shall be provided by any device wishing to expose a time-code generation function using the plug-and-play protocol.

b. If a device does not wish to expose a time-code generation function using the plug-and-play protocol then the Time-Code Generation field set shall be reserved.

c. The Time-Code Generation field set shall comprise the following fields:

1. the Time-Code Generation Control field; and

2. the Time-Code Generation Period field.

d. Time-Code Generation field set fields shall be identified as shown in Table 5-37.

Table 5-37: Time-Code Generation field set fields

Field Identifier	Field
0	Time-Code Generation Control
1	Time-Code Generation Period

5.3.5 SpaceWire-PnP protocol

5.3.5.1 Overview

The SpaceWire-PnP protocol fields provide the capability to manage the SpaceWire-PnP protocol implementation on devices.

5.3.5.2 Fields

5.3.5.2.1 Maximum Write Length field

a. The Maximum Write Length field shall specify the maximum number of fields which may be written in a write operation on the peripheral device.

b. The Maximum Write Length field shall include a 15-bit field containing the maximum number of fields which may be written in a write operation on the peripheral device.

c. The maximum write length specified by the Maximum Write Length field shall not be:

1. less than 2 (see clause 5.3.2.4); or

2. greater than 16,384 (0x4000) (see clause 5.2.4.8.4).

d. The Maximum Write Length field shall include a 17-bit reserved field.

e. The Maximum Write Length field shall be encoded as shown in Table 5-38.

Table 5-38: Maximum Write Length field

31 … 15	14 … 0
Reserved	Maximum Write Length

f. The Maximum Write Length field shall be read-only.

g. The value of the Maximum Write Length field shall be static.

NOTE This means that the value of the field may not change during the operation of the device

5.3.5.2.2 Maximum Read Length field

a. The Maximum Read Length field shall specify the maximum number of fields which may be read in a read operation on the peripheral device.

b. The Maximum Read Length field shall include a 15-bit field containing the maximum number of fields which may be read in a read operation on the peripheral device.

c. The maximum write length specified by the Maximum Read Length field shall not be:

1. less than 8 (see clause 5.3.2.4); or

2. greater than 16,384 (0x4000) (see clause 5.2.5.8.4).

d. The Maximum Read Length field shall include a 17-bit reserved field.

e. The Maximum Read Length field shall be encoded as shown in Table 5-39.

Table 5-39: Maximum Read Length field

31 … 15	14 … 0
Reserved	Maximum Read Length

f. The Maximum Read Length field shall be read-only.

g. The value of the Maximum Read Length field shall be static.

NOTE This means that the value of the field may not change during the operation of the device

5.3.5.3 Field identification

5.3.5.3.1 Protocol identifier

a. Support for the SpaceWire-PnP Protocol fields shall be indicated by specifying a Vendor ID of zero and a protocol ID matching that for SpaceWire-PnP (see clause 5.2.3.2) in the protocol support list (see clause 5.3.3.3.6.).

5.3.5.3.2 Protocol and application indices

a. SpaceWire-PnP Protocol fields shall be identified by using an Application_Index parameter of zero and a Protocol_Index corresponding to the index of the entry in the protocol support list specifying support for the SpaceWire-PnP Protocol fields (see clause 5.3.5.3.1) when applying a read, write or read-modify-write operation.

5.3.5.3.3 Field sets

a. SpaceWire-PnP Protocol fields shall be grouped into one field set, the Protocol Information field set.

b. SpaceWire-PnP Protocol field sets shall be identified as shown in Table 5-40.

Table 5-40: SpaceWire-PnP Protocol field sets

Field Set Identifier	Field Set
0	Protocol Information

5.3.5.3.4 Protocol Information field set

a. The Protocol Information field set shall comprise the following fields:

1. the Maximum Write Length field; and
2. the Maximum Read Length field.

b. Protocol Information field set fields shall be identified as shown in Table 5-41.

Table 5-41: Protocol Information field set fields

Field Identifier	Field
0	Maximum Write Length
1	Maximum Read Length

5.3.6 Network management service

5.3.6.1 Overview

The Network Management Service fields provide the capability to manage the Network Management Service implementation on devices.

5.3.6.2 Fields

5.3.6.2.1 Service Status field

a. The Service Status field shall indicate the operational status of the service.

b. The Service Status field shall comprise an 8-bit service status and a 24-bit reserved field.

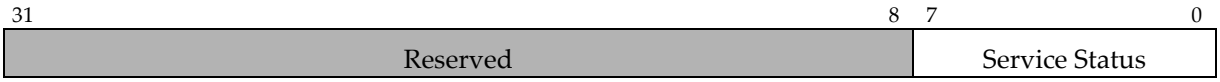
c. A service status value of zero shall indicate nominal operation.

d. A service status value of non-zero shall indicate an error failure condition.

e. The meaning of non-zero service status values shall be assigned by the organisation manufacturing or selling the device.

f. The Service Status field shall be encoded as shown in Table 5-42.

Table 5-42: Service Status field



g. The Service Status field shall be read-only.

5.3.6.3 Field identification

5.3.6.3.1 Application identifier

a. Support for the Network Management Service fields shall be indicated by specifying a Vendor ID of zero and an Application ID of 1 in the application support list (see clause 5.3.3.3.7.).

5.3.6.3.2 Protocol and application indices

a. Network Management Service fields shall be identified by using a Protocol_Index parameter of zero and an Application_Index corresponding to the index of the entry in the application support list specifying support for the Network Management Service fields (see clause 5.3.6.3.1) when applying a read, write or read-modify-write operation.

5.3.6.3.3 Field sets

a. Network Management Service fields shall be grouped into one field set, the Service Information field set.

b. Network Management Service field sets shall be identified as shown in Table 5-43.

Table 5-43: Network Management Service field sets

Field Set Identifier	Field Set
0	Service Information

5.3.6.3.4 Service Information field set

a. The Service Information field set shall comprise the Service Status field only.

b. The Service Status field shall be identified using a field Identifier of 0.

Annex A (informative) Network Discovery

This informative annex illustrates methods for discovering SpaceWire networks using SpaceWire-PnP.

A.1 Example network

A diagram of an example SpaceWire network is shown in Figure A-1. This network has been deliberately chosen to be very simple in order to serve as an example for network discovery. It is not necessarily representative of a practical SpaceWire network.

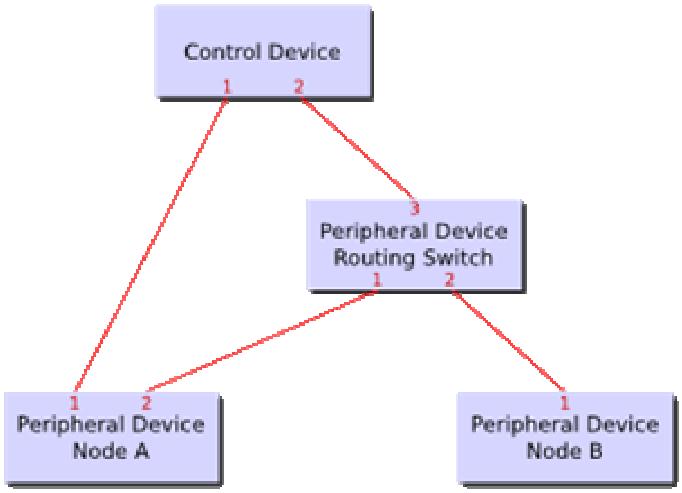


Figure A-1: Example SpaceWire network

Each box in the diagram represents a device. The device at the top of the diagram is a control device and will be conducting the network discovery. The devices are interconnected with SpaceWire links. The number positioned where a link meets a device indicates the number of that link on the device.

A.2 Identifying a device

To identify a peripheral device the control device issues a plug-and-play read operation on all fields of the Device Identification field set. This will inform the control device of:

1. the device type and version information;

2. the number of links on the device;
3. which links on the device are active;
4. whether the device has already been identified by a control device;
5. the device identifier, if one has been assigned.

This information is sufficient to be able to identify the device.

A.3 Discovering the network

In order to discover the network, the control device should attempt to read device identification information from the device attached to each active link in the network. This is much like exploring a tree structure and so may be carried out either depth- or breadth-first, or any combination of the two. In this example we choose to explore depth-first.

Firstly, the control device will identify which of its own links are active. In this example, the control device has two active links, numbered 1 and 2.

The control device will then attempt to identify the device at the end of link 1 by reading the device identification information of Node A. This information will indicate that:

1. the device is a node;
2. it has not already been identified (there is no assigned identifier);
3. it has two active links, links 1 and 2;
4. the control device is attached to link 1.

The control device then assigns a device identifier of 1 (for example) to Node A by applying a compare-and-swap operation to the Device ID field. The knowledge of the network held by the control device now corresponds to the network shown in Figure A-2.

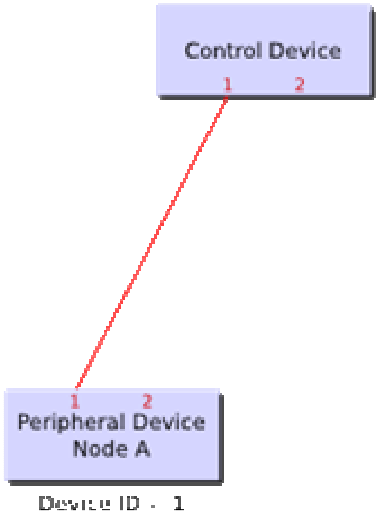


Figure A-2: Network discovered by the control device after the first step

There are no further devices to discover using link 1 on the control device so the process is repeated through link 2. The control device attempts to identify the device at the end of link 2 by reading the device identification information of the routing switch. This information will indicate that:

1. the device is a routing switch;
2. it has not already been identified (there is no assigned identifier);
3. it has three active links, links 1, 2 and 3;
4. the control device is attached to link 3.

The control device then assigns a device identifier of 2 to the routing switch by applying a compare-and-swap operation to the Device ID field. The knowledge of the network held by the control device now corresponds to the network shown in Figure A-3.

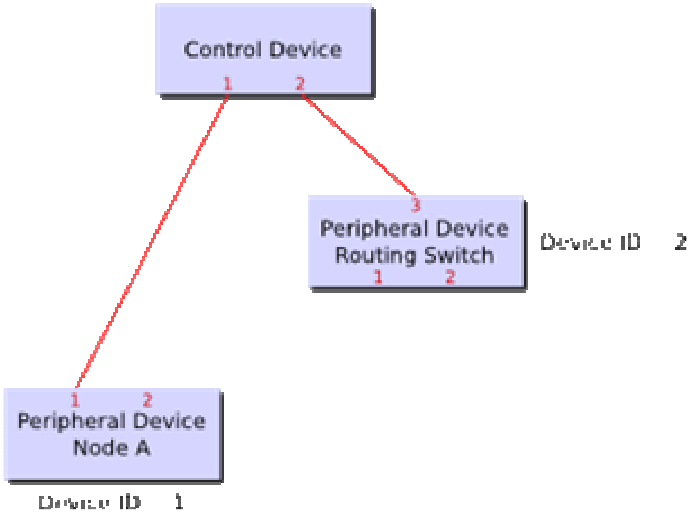


Figure A-3: Network discovered by the control device after the second step

As the device just discovered is a routing switch, discovery may progress further. The control device will therefore attempt to identify the devices attached to all active links on the routing switch. In this case there are three active links, but the connection for link 3 is already known: it is attached to the control device. The control device then continues the process by attempting to identify the device at the end of link 1 of the routing switch. This is done, as before, by reading the device identification information. This information will indicate that:

1. the device is a node;
2. it has already been identified and has an assigned identifier of 1;
3. it has two active links, links 1 and 2;
4. the routing switch device is attached to link 2.

The knowledge of the network held by the control device now corresponds to the network shown in Figure A-4.

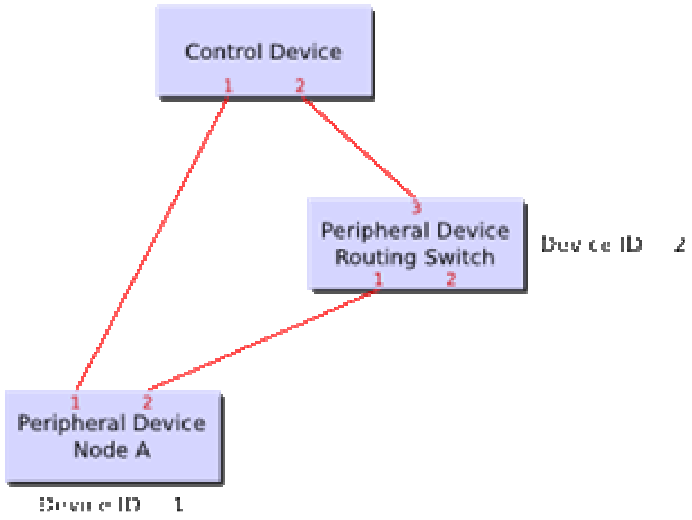


Figure A-4: Network discovered by the control device after the third step

As the discovered device was a node, discovery can continue no further through link 1 of the routing switch so the process is repeated through link 2. The control device attempts to identify the device at the end of link 2 by reading the device identification information of Node B. This information will indicate that:

1. the device is a node;
2. it has not already been identified (there is no assigned identifier);
3. it has one active link, link 1;
4. the routing switch is attached to link 1.

The control device then assigns a device identifier of 3 to Node B by applying a compare-and-swap operation to the Device ID field. As there are no further links to discover, the knowledge of the network held by the control device is complete, as shown in Figure A-5 (matching that shown in Figure A-1).

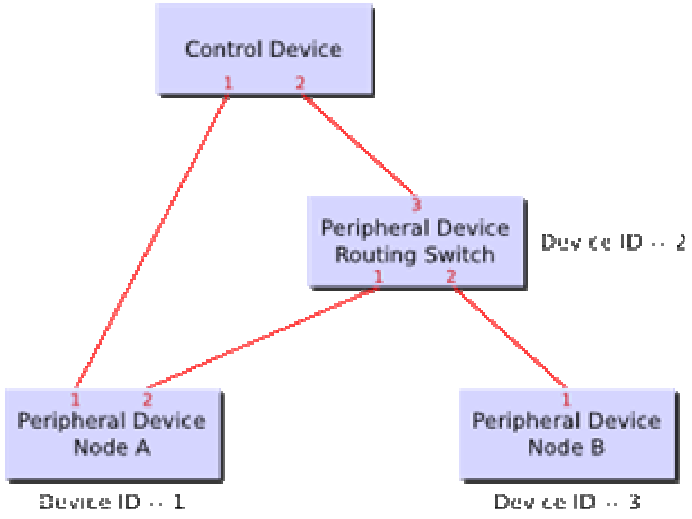


Figure A-5: Network discovered by the control device after the fourth, and final, step

A.4 Network changes

If a single device were to be disconnected and then reconnected to the network, without resetting the device, the Device ID field would automatically be cleared.

If several interconnected devices, a device subnetwork, are disconnected and then reconnected to the network, without resetting any device, the Device ID fields in some devices will not be cleared. However, this situation can be detected by a control device as a device on the subnetwork can easily be identified, despite the non-zero Device ID field, as it can only be accessed via a routing switch with a Device ID field of zero.

A.5 Support for multiple control devices

The control device which assigned the Device ID can be identified using the Owner Link, Owner Address and Owner Logical Address fields. This information can be used by a control device to determine if it already owns the device and if the current owner of the device is valid. The validity of the current owner can only be determined if the owning control device:

1. is also a peripheral device;
2. is reachable.

An example of an unreachable control device is shown in Figure A-6. In this figure, neither control device is reachable from the other. This means that the validity of the ownership information in the peripheral device cannot be confirmed by the control device which does not own the device.

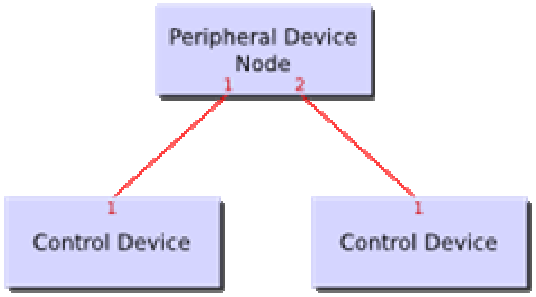


Figure A-6: An unreachable control device

Where a control device meets both of these conditions, its Device Status flag can be read to determine validity. Should the Device Status flag not indicate nominal operation, or no reply is received in a suitable time, the control device can be considered as invalid.

The ability for a control device to correctly determine the owner of a device on a subnetwork which has been disconnected and reconnected without resetting any device depends on the addressing scheme used by the control devices. If all control devices use path addressing then it is possible to identify the route from any device to its owner. If this route is broken by a device with a cleared Device ID, or the route is invalid, then the current owner can be considered as invalid. If, on the other hand, logical addressing is used, the ability to determine owner

validity is not guaranteed. It will depend on many different factors including logical address usage and the network topology before and after the change.

If a control device wishes to take ownership of a device which has a current valid owner then there is competition for ownership. This is best resolved using an algorithm which is known to be common between all control devices on the network. The algorithm can easily be based on the value of the assigned Device ID or the value of the Owner Link field versus the Return Link field. These fields identify the links to which the two competing owners are attached. If control devices always ensure that they own all routing switches on the path to the control device then these two fields are guaranteed to be different. This could then be used as a last resort for conflict resolution.

Ultimately, the policies governing the way networks are designed and discovered, and the way that devices are claimed, must be decided appropriately for the target mission or application scenario.

Bibliography

ECSS-S-ST-00	ECSS system – Description, implementation and general requirements
CCSDS RD20 850.0-G-R1.1	CCSDS Spacecraft Onboard Interface Services – Green Book
http://spacewire.esa.int	SpaceWire website