EUROPEAN COOPERATION

ECSS

FOR SPACE STANDARDIZATION

Space engineering

SpaceWire Protocols

This ECSS is a draft standard circulated for xxxxxxxxxxx. It is therefore subject to change without notice and may not be referred to as an ECSS Standard until published as such.

xxxxxxxxxxx ends on XX XXXXXX 2007

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

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.

The formulation of this Standard takes into account the existing ISO 9000 family of documents.

This Standard has been prepared by the SpaceWire Working Group, reviewed by the ECSS Executive Secretariat and approved by the ECSS Technical Authority.

Disclaimer

ECSS does not provide any warranty whatsoever, whether expressed, implied, or statutory, including, but not limited to, any warranty of merchantability or fitness for a particular purpose or any warranty that the contents of the item are error-free. In no respect shall ECSS incur any liability for any damages, including, but not limited to, direct, indirect, special, or consequential damages arising out of, resulting from, or in any way connected to the use of this standard, whether or not based upon warranty, contract, tort, or otherwise; whether or not injury was sustained by persons or property or otherwise; and whether or not loss was sustained from, or arose out of, the results of, the item, or any services that may be provided by ECSS.

Published by: ESA Requirements and Standards Division
ESTEC, P.O. Box 299,
2200 AG Noordwijk
The Netherlands
ISSN: 1028-396X
Price: € 30
Printed in: The Netherlands.

Change log

Table of contents

Figures

Tables

Introduction

xxx

1. Scope

This xxxxx

2. Normative references

The following dated normative documents are called by the requirements of this ECSS Standard and therefore constitute requirements to it. Subsequent amendments to, or revisions of any of these publications do not apply.

NOTE However, parties to agreements based on this ECSS Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated be-low.

ECSS-P-001B	ECSS – Glossary of terms
ECSS-E50-12A	SpaceWire – links, nodes, routers and networks

3.This heading is here because of problems with a clause not being a heading

3. Terms, definitions and abbreviated terms

3.1. Terms defined in other standards

For the purpose of this Standard, the terms and definitions from ECSS-P-001B apply.

3.2. General

In this document hexadecimal numbers are written with the prefix 0x, for example 0x34 and 0xDF15. Binary numbers are written with the prefix 0b, for example 0b01001100 and 0b01.

Decimal numbers have no prefix.

3.3. Terms specific to the present standard

3.3.1. byte

8-bits where bit 7 is the most-significant bit

3.3.2. command

an instruction to a SpaceWire node (target) to perform some action

EXAMPLE write data to memory

3.3.3. command packet

a packet that contains a command

3.3.4. confirmation

a primitive passed from a service provider to a service user to indicate the success or otherwise of a previous service request

3.3.5. data character

a SpaceWire symbol containing 8-bits of user information

3.3.6. EEP

Error End of Packet marker of a Packet which indicates that the Packet was terminated prematurely

3.3.7. EOP

End Of Packet marker of the Packet

3.3.8. extender protocol identifier

two data characters following a protocol identifier which has value 0x00 that identify a particular protocol being used for communication

3.3.9. logical address

identifier of a initiator or target which can be used to route a Packet to the target or, if path addressing is being used, to confirm that the final target is the correct one i.e. that the logical address of the target matches the logical address in the packet

3.3.10. indication

a primitive passed from a service provider to a service user to provide information or status to the service user

3.3.11. initiator

SpaceWire node that starts a transaction by sending a command to a SpaceWire node

3.3.12. initiator user application

an application in an initiator that is using the SpaceWire protocol services

3.3.13. memory

any type of addressable storage element including random access memory, registers, FIFO, mailboxes

3.3.14. packet

a SpaceWire packet

3.3.15. path address

sequence of one or more SpaceWire data characters that defines the route to a target by specifying, for each router encountered on the way to the target, the output port that a Packet is forwarded through

3.3.16. protocol identifier

data character that identifies a particular protocol being used for communication

3.3.17. reply

a response sent by a target to the initiator or some other node expecting the reply to provide the required information or to indicate that some commanded action has been completed by the target

3.3.18. reply packet

packet containing a reply

3.3.19. request

a primitive passed from a service user to a service provider to request a service

3.3.20. response

a primitive passed from a service user to a service provider in response to an indication from the service provider

3.3.21. target

SpaceWire node that responds to a command sent by an initiator

3.3.22. target user application

an application in a target that is using the SpaceWire protocol services

3.3.23. transaction

an interaction between an initiator and a target

3.3.24. word

multiple bytes held in a single memory location

3.4. Abbreviated terms

The following abbreviations are defined and used within this standard:

Abbreviation	**Meaning**
CRC	Cyclic Redundancy Code
DMA	Direct Memory Access
EEP	Error End of Packet
EOP	End Of Packet
FCT	Flow Control Token
FIFO	First In First Out
ID	Identifier
Inc	Increment
Len	Length
LS	Least-Significant
LSB	Least-Significant Bit
MS	Most-Significant
MSB	Most-Significant Bit
RMAP	Remote Memory Access Protocol
RMW	Read-Modify-Write
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

4.This heading is here because of problems with clause not being a heading

4. Principles

4.1. SpaceWire Protocols

This standard contains several protocols that can be used in conjunction with the SpaceWire protocols defined in ECSS-E50-12A.

To distinguish between the various protocols a protocol identifier is used which is described in clause 5. The protocols that operate over SpaceWire are then described one per clause from clause 6 onwards.

4.2. Remote Memory Access Protocol (RMAP)

The aim of RMAP is to support reading from and writing to memory in a remote SpaceWire node. RMAP can be used to configure a SpaceWire network, control SpaceWire nodes, and to transfer data to and from SpaceWire nodes. RMAP is defined in clause 6.

4.3. CCSDS Packet Encapsulation Protocol

The aim of the CCSDS Packet Encapsulation Protocol is to transfer CCSDS Packets across a SpaceWire network. It does this by encapsulating the CCSDS Packet in a SpaceWire packet, transferring it across the SpaceWire network and then extracting the CCSDS Packet at the target. The CCSDS Packet Encapsulation Protocol is defined in clause 7.

5.This heading is here because of problems with clause not being a heading

5. Protocol identification

5.1. Overview

The protocol identification scheme enables many different protocols to operate concurrently over a SpaceWire network without them interfering with each other. To achieve this, an identifier is given to each protocol. Nodes receiving packets process and respond to them according to the protocol specified by the Protocol Identifier in the packet. If a packet arrives with a particular Protocol Identifier that is not supported by a node then it is ignored.

5.2. Protocol identification

5.2.1. Addressing

a. A packet containing a Protocol Identifier shall start with a single byte logical address when it arrives at the target.

NOTE See Figure 5-1.

NOTE When sent by the initiator the packet can have one or more leading path or logical address bytes which are stripped off (SpaceWire Address) on the way through the SpaceWire network leaving the single logical address byte when it arrives at the target.

b. The logical address 254 (0xFE) shall be used as a default value when the target does not have another value specified for its logical address.

NOTE When the initiator does not know the logical address of the target the default logical address 254 (0xFE) can be used.

c. A target may choose to ignore packets with logical address 254 (0xFE).

NOTE If a packet with a logical address is ignored then the target can record and make available a count of the number of packets it received and ignored with logical address 254 (0xFE).

d. A target may accept packets with one or more different logical address values.

EXAMPLE A node accepting packets with logical addresses 60, 61 or 254.

5.2.2. Protocol Identifier

a. A Protocol Identifier shall comprise a single byte immediately following the logical address.

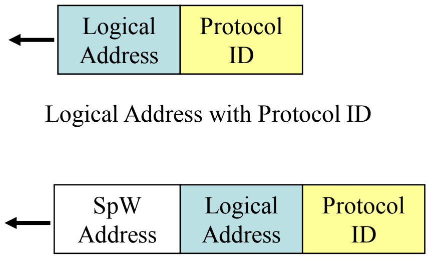
NOTE See Figure 5-1.

b. A value of zero shall be used to identify an Extended Protocol Identifier.

NOTE The value of zero in the Protocol Identifier byte is reserved for extension of the Protocol Identifier, as specified in sub-clause 5.2.3.

c. A Protocol Identifier with a value of 255 (0xFF) shall not be used.

NOTE It is reserved for future use.



Logical Address with Protocol ID

SpaceWire Address and Logical Address with Protocol ID

Figure 5-1 Protocol Identifier position

5.2.3. Extended Protocol Identifier

a. If an Extended Protocol Identifier is supported, the following shall apply:

1. Protocol Identifier has the value zero (0x00)
2. The two bytes following the reserved Protocol Identifier (zero) form a 16-bit Extended Protocol Identifier

NOTE This allows up to 65535 protocols to be carried over a SpaceWire network.

NOTE An Extended Protocol Identifier need not be implemented.

NOTE See Figure 5-2.

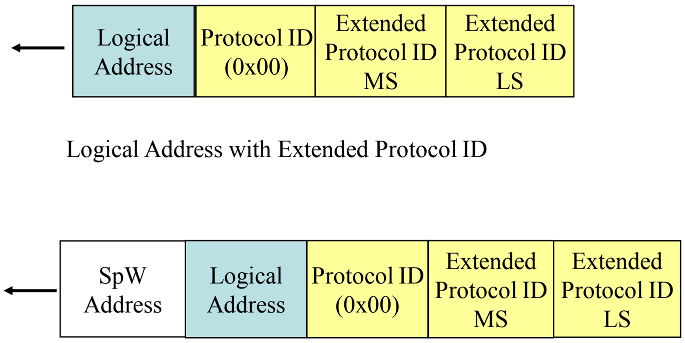
b. If an Extended Protocol Identifier is not supported, then a packet with a Protocol Identifier with the value zero (reserved Protocol Identifier) shall be discarded when received.

NOTE If a target ignores the Extended Protocol Identifier then it can record and make available a count of the number of packets it received with an Extended Protocol Identifier.

c. Extended Protocol Identifiers with values in the range 0x0000 to 0x00FF are reserved and shall not be used.

d. A packet with an Extended Protocol Identifier with a value in the range 0x0000 to 0x00FF shall be discarded when received.

NOTE These values are reserved for future use.



Logical Address with Extended Protocol ID

SpaceWire Address and Logical Address with Extended Protocol ID

Figure 5-2 Extended Protocol Identifier

5.2.4. Ignoring unknown protocols

If a packet arrives with a Protocol Identifier or Extended Protocol Identifier that is not supported (unknown) by that target then the packet shall be discarded.

NOTE The target can count the number of packets that arrive at a target with unknown Protocol Identifier or Extended Protocol Identifier can be kept and made available by the target.

5.2.5. Protocol Identifier and Extended Protocol Identifier Allocation

a. Protocol Identifiers in the range 1 to 239 (0x01 to 0xEF) that shall be used are those listed in Table 5-1.

Table 5-1 Protocol Identifier Allocation

Protocol Identifier	Protocol
0	Extended Protocol Identifier
1	Remote Memory Access Protocol
2	CCSDS Packet Encapsulation Protocol
239	Serial Transfer Universal Protocol

NOTE These identifiers have been assigned by the SpaceWire working group. The protocols starting at number 1 and working upwards as defined in this standard document define the current set of approved SpaceWire protocols and their Protocol Identifiers. The protocols starting at 239 and working downwards are legacy protocols and are not covered by this standard document.

NOTE The reader is advised to consult any amendment sheets for the latest set of Protocol Identifiers and Extended Protocol Identifiers. The amendment sheets are to be found on the ECSS website.

b. Protocol Identifiers in the range 240 to 254 (0xF0 to 0xFE) shall be assigned by the project.

NOTE Developers can use these Protocol Identifiers but it is important to note that they can clash with protocols being developed by other users. Concurrent operation of different protocols is only assured for Protocol Identifiers in the range 1 to 239 (0x01 to 0xEF).

NOTE Proven protocols can be recommended for adoption by the SpaceWire working group and then be included in future revisions or extensions to this SpaceWire Protocols standard. Once adopted they are given a unique Protocol Identifier in the range 1 to 239.

NOTE No Extended Protocol Identifiers have been allocated.

6.This heading is here because of problems with clause not being a heading

6.
Remote memory access protocol

6.1. Overview

6.1.1. Purpose

The remote memory access protocol (RMAP) has been designed to support a wide range of SpaceWire applications. Its primary purposes however are to configure a SpaceWire network, to control SpaceWire nodes and to gather data and status information from those nodes. RMAP can operate alongside other communication protocols running over SpaceWire.

RMAP can be used to configure SpaceWire routing switches, setting their operating parameters and routing table information. It can also be used to monitor the status of those routing switches. RMAP can be used to configure and read the status of nodes on the SpaceWire network. For example, the operating data rate of a node can be set to 100 Mbits/s and the interface can be set to auto-start mode. RMAP can also be used to download and debug software on a remote processor.

For simple SpaceWire units without an embedded processor, RMAP can be used to set application configuration registers, to read status information and to read from or write data to memory in the unit.

For intelligent SpaceWire units RMAP can provide the basis for a wide range of communication services. Configuration, status gathering and data transfer to and from memory or mailboxes can be supported.

6.1.2. Guide to clause 6

Specification of the fields used in RMAP commands and replies is given in sub-clause 6.2. The CRC used by RMAP is specified in sub-clause 6.3. The write command is defined in sub-clause 6.4, the read command in sub-clause 6.5 and the read-modify-write command in sub-clause 6.6. The error codes that are used in RMAP replies are listed in sub-clause 6.7. The way in which partial implementations of RMAP can be implemented is described in sub-clause 6.8. Sub-clause 6.9 specifies the conformance statements i.e. sub-clauses that are implemented and the ancillary information that is provided, in order for a supplier to claim conformance to the SpaceWire RMAP standard. Example VHDL and C-code for the 8-bit CRC used by RMAP is given in 6.10.

6.1.3. RMAP operations

RMAP is used to write to and read from memory, registers, FIFO memory, mailboxes, etc, in a target on a SpaceWire network. Input/output registers, control/status registers and FIFOs are memory-mapped and therefore are accessed as memory. Mailboxes are indirect memory areas that are referenced using a memory address.

All read and write operations defined in the RMAP protocol are posted operations i.e. the initiator does not wait for a reply to be received. This means that many read and write commands can be outstanding at any time. There is no timeout mechanism implemented in RMAP for missing replies. If a reply timeout mechanism is used, it is implemented in the initiator user application.

6.1.3.1. Write commands

The write command provides a means for one node, the initiator, to write zero or more bytes of data into a specified area of memory in another node, the target on a SpaceWire network.

Write commands can be acknowledged or not acknowledged by the target when they have been received correctly. If the write command is to be acknowledged and there is an error with the write command, the target replies with an error/status code to the initiator (or other node) that sent the command. The error/status code can only be sent to the initiator if the write command header was received intact, so that a target that detected an error knows where to send the reply. If no reply is requested then the fact that an error occurred can be stored in a status register in the target.

Write commands can perform the write operation after verifying that the data has been transferred to the target without error, or it can write the data without verification. Verification on the data can be performed only by buffering in the target to store the data while it is being verified, before it is written. The amount of buffering is likely to be limited so verified writes can only be performed for a relatively small amount of data that fits into the available buffer at the target. Verified writes are normally used when writing to configuration or control registers. Larger amounts of data can be written but without verification prior to writing. Verification in this case is done after the data has been written.

The acknowledged/non-acknowledged and verified/non-verified options to the write command result in four different write operations:

- **Write non-acknowledged, non-verified** – writes zero or more bytes to memory in a target. The command header is checked using a CRC before the data is written, but the data itself is not checked before it is written. No reply is sent to the initiator of the write command. This command is typically used for writing large amounts of data to a target where it can be safely assumed that the write operation completed successfully. For example the writing of camera data to a temporary working buffer.

- **Write non-acknowledged, verified** – writes zero or more bytes to memory in a target. Both the command header and data are checked using CRCs before the data is written. This limits the amount of data that can be transferred in a single write operation, but writing erroneous data to memory is unlikely. No reply is sent to the initiator of the write command. This command is typically used for writing command registers and small amounts of data to a target where it can be safely assumed that the write operation completed successfully. For example writing many commands to different configuration registers in a device and then checking for an error using a status register.

- **Write acknowledged, non-verified** – writes zero or more bytes to memory in a target. The command header is checked using a CRC before the data is written, but the data itself is not checked before it is written. A reply to indicate the command execution status is sent to the initiator of the write command. This command is typically used for writing large amounts of data to a target where it can be safely assumed that the write operation completed successfully, but an acknowledgement is required. For example writing sensor data to memory.

- **Write acknowledged, verified** – writes zero or more bytes to memory in a target. Both the command header and data are checked using CRCs before the data is written. This limits the amount of data that can be transferred in a single write operation, but writing erroneous data to memory is unlikely. A reply to indicate the command execution status is sent to the initiator of the write command. This command is typically used for writing small amounts of data to a target where it is important to have confirmation that the write operation was executed successfully. For example writing to configuration registers.

6.1.3.2. Read commands

The read command provides a means for one node, the initiator, to read zero or more bytes of data from a specified area of memory in another node, the target on a SpaceWire network. The data read is returned in a reply packet which normally goes back to the initiator.

6.1.3.3. Read-modify-write

The read-modify-write command provides a means for one node, the initiator, to read a memory location in another node, the target, modify the value read in some way and then write the new value back to the same memory location. The original value read from memory is returned in a reply packet to the initiator.

6.2. RMAP command and reply fields

6.2.1. Target SpaceWire Address field

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

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

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

c. The Target SpaceWire Address field shall not be used when a single logical address is being used for routing the command to the target.

NOTE In this case the command is routed to the target by the Target Logical Address contained in the Target Logical Address field.

6.2.2. Target Logical Address field

Target Logical Address field shall be an 8-bit field that contains a logical address of the target.

NOTE The Target Logical Address field is normally set to a logical address recognised by the target.

NOTE If the target does not have a specific logical address then the Target Logical Address field can be set to the default value 254 (0xFE).

NOTE A target can have more than one logical address.

6.2.3. 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 1 (0x01) which is the Protocol Identifier for the Remote Memory Access Protocol.

6.2.4. Instruction field

6.2.4.1. General

The Instruction field shall be an 8-bit composite field that comprises the packet type, command and Reply Address length fields.

6.2.4.2. Packet type field

a. The Packet Type field shall be a 2-bit field that determines the type of RMAP packet i.e. a command (0b01) or reply (0b00).

b. The other possible values (0b10 and 0b11) of the packet type field are reserved.

6.2.4.3. Command field

a. Command field shall be:

1. A 4-bit field in an RMAP command that specifies the type of command, or

2. A 4-bit field in an RMAP reply that specifies the type of command that caused the reply.

b. The command codes shall have the meanings listed in Table 6-1.

Table 6-1 RMAP Command Codes

Bit 5	Bit 4	Bit 3	Bit 2	Command Field
Write/ Read	Verify Data Before Write	Reply	Increment Address	Function
0	0	0	0	Invalid
0	0	0	1	Invalid
0	0	1	0	Read single address
0	0	1	1	Read incrementing addresses
0	1	0	0	Invalid
0	1	0	1	Invalid
0	1	1	0	Invalid
0	1	1	1	Read-Modify-Write incrementing addresses
1	0	0	0	Write, single address, don’t verify before writing, no reply
1	0	0	1	Write, incrementing addresses, don’t verify before writing, no reply
1	0	1	0	Write, single address, don’t verify before writing, send reply
1	0	1	1	Write, incrementing addresses, don’t verify before writing, send reply
1	1	0	0	Write, single address, verify before writing, no reply
1	1	0	1	Write, incrementing addresses, verify before writing, no reply
1	1	1	0	Write, single address, verify before writing, send reply
1	1	1	1	Write, incrementing addresses, verify before writing, send reply

6.2.4.4. Reply Address length field

The Reply Address Length field shall be:

1. A 2-bit field in an RMAP command that determines the number of bytes in the Reply Address field of a command.
2. A 2-bit field in an RMAP reply that is a copy of the 2-bit Reply Address Length field in the command that caused the reply.

6.2.5. Key field

The Key field shall be an 8-bit field that contains a key which is matched by the target user application in order for the RMAP command to be authorised.

NOTE The Key is only used for command authorisation. It is not used for other purposes.

6.2.6. Reply Address field

a. The Reply Address field shall be a 0, 4, 8 or 12-byte field in a command that contains the SpaceWire address for the reply to the command.

b. The size of the Reply Address field shall depend on the value of the Reply Address Length field as detailed in Table 6-2.

Table 6-2 Reply Address field Size

Value of Reply Address Length Field	Size of Reply Address field
0b00	0
0b01	4 bytes
0b10	8 bytes
0b11	12 bytes

NOTE The Reply Address is not needed if logical addressing is being used. The Reply Address is normally used by the target to send replies or data back to the initiator that requested a write or read operation using path addressing. The Reply Address allows path addressing and regional logical addressing to be used to specify the node that is to receive the reply (normally the initiator).

c. Leading bytes with the value 0x00 in the Reply Address field shall be ignored.

d. If the Reply Address Length field is not zero and the Reply Address bytes are all zero (0x00), a single zero value data character shall be sent as part of the Reply SpaceWire Address field.

NOTE This is so that a Reply SpaceWire Address comprising a single zero (0x00) data character is possible.

e. Any characters in the Reply Address field after the leading bytes with the value 0x00 shall form the Reply SpaceWire Address.

f. SpaceWire path addressing and regional addressing shall be used to form the Reply Address field.

EXAMPLE Some examples of the mapping between the contents of the Reply Address field and the Reply SpaceWire Address are listed in Table 6-3.

Table 6-3 Example Reply Address field to Reply SpaceWire Address mappings

Reply Address field	Resulting Reply SpaceWire Address
0x00 0x00 0x00 0x00	0x00
0x00 0x00 0x01 0x02	0x01 0x02
0x00 0x01 0x00 0x02	0x01 0x00 0x02
0x00 0x01 0x02 0x00	0x01 0x02 0x00
0x00 0x00 0x00 0x01 0x02 0x03 0x04 0x05	0x01 0x02 0x03 0x04 0x05
0x00 0x00 0x66 0x05	0x66 0x05
0x00 0x54 0x08 0x00	0x54 0x08 0x00

g. The Reply Address field shall not be used when a single logical address is used for routing the reply to its initiator (or other node).

NOTE In this case the reply is routed to the initiator by the Initiator Logical Address.

6.2.7. Initiator Logical Address field

The Initiator Logical Address field shall be an 8-bit field that contains either:

- The logical address of the initiator of a command packet, if the initiator has a logical address, or
- 254 (0xFE) otherwise.

NOTE The value 254 (0xFE) is the default logical address (see 5.2.1).

NOTE An initiator can have more than one logical address.

6.2.8. Transaction Identifier field

a. The Transaction Identifier field shall be a 16-bit field used to associate replies with the command that caused the reply.

b. The Transaction Identifier in a reply shall have the same value as the Transaction Identifier in the command that caused the reply.

c. The most significant byte of the Transaction Identifier shall be sent first.

NOTE Typically Transaction Identifiers are an incrementing integer sequence, with each successive RMAP transaction being given the next number in the sequence. The intention of the Transaction Identifier is to uniquely identify a transaction.

6.2.9. Extended Address field

The Extended Address field shall be an 8-bit field that contains the most-significant 8-bits of the memory address extending the 32-bit memory address to 40-bits.

6.2.10. Address field

a. The Address field shall be a 32-bit field that contains the least-significant 32-bits of the memory address.

b. The most significant byte of the Address field shall be sent first.

6.2.11. Data Length field

a. The Data Length field shall be a 24-bit field that contains the length in bytes of the data field or data and mask field in a command or reply.

b. The most significant byte of the Data Length field shall be sent first.

6.2.12. Header CRC field

The Header CRC field shall be an 8-bit field that contains an 8-bit Cyclic Redundancy Code (CRC) covering each byte in the header, starting with the Target Logical Address and ending with the byte before the Header CRC in a command and starting with the Initiator Logical Address and ending with the byte before the Header CRC in a reply.

6.2.13. Data field

The Data field shall be a variable length field containing the data bytes that are written in a write command or the data bytes that are read in a read reply, or read and written in a read-modify-write command and reply.

NOTE The order of the bytes in the data field is up to the specific implementation and is defined in the target product characteristic table (see sub-clause 6.9). This is a change from draft F, which stated that “when writing to memory organised in words (e.g. 32-bit words) then the first byte sent is the most significant byte of the word.”

6.2.14. Mask field

The Mask field shall be a variable length field containing the mask in a read-modify-write command.

6.2.15. Data CRC field

The Data CRC field shall be an 8-bit field that contains an 8-bit Cyclic Redundancy Code (CRC) covering each byte in the data and mask field starting with the byte after the Header CRC and ending with the byte before the Data CRC.

6.2.16. Reply SpaceWire Address field

The Reply SpaceWire Address field shall be a variable length field formed from the contents of the Reply Address field of a command which is used to route a reply back to the initiator or other intended destination for the reply.

6.2.17. Status field

The Status field shall be an 8-bit field in a reply containing a status/error code as defined in sub-clause 6.7.

6.3. Cyclic Redundancy Code

a. The same method of calculating the CRC shall be used for both the Header CRC and the Data CRC.

b. The CRC calculation procedures shall:

1. use modulo 2 arithmetic for polynomial coefficients;
2. use a systematic binary ($n+8$, n) block code, where $n+8$ is the number of bits of the codeword $c(x)$ and n is divisible by 8; n is the number of bits covered by the CRC;
3. use the following generating polynomial:

   $g(x) = x^8 + x^2 + x + 1$
4. use byte format as input and output, for which the bits are represented as:

   $b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$

   where b_7 is the most significant bit and b_0 is the least significant bit;

c. The CRC generation procedure shall behave as follows:

1. The procedure accepts an n-bit input which is used to construct $m(x)$, where:
   - the n-bit input is defined to be the set of bits $B_{i,j}$ grouped into $n/8$ bytes where $i=\{0, 1, \ldots, n/8\text{-}1\}$ is the byte index and $j=\{7,6, \ldots, 0\}$ is the bit index;
   - the $n/8$ input bytes correspond to the RMAP fields covered by the CRC excluding the CRC byte; the first byte transmitted has index i=0; the last byte transmitted has index $i=n/8$-1;
   - $m(x)$ is a polynomial $m_{n-1}x^{n-1} + m_{n-2}x^{n-2} + \ldots + m_0x^0$ having binary coefficients m_i;
   - $m(x)$ can be represented as an n-bit vector where coefficient m_{n-1} of the highest power of x is the most significant bit and coefficient m_0 of the lowest power of x is the least significant bit;
   - the bit vector representation of $m(x)$ is formed by concatenating the $n/8$ bytes of the input in transmission order, where the least significant bit b_0 of each byte is taken first and the most significant bit b_7 of each byte is taken last:

     $m_{n-1}=B_{0,0}$, $m_{n-2}=B_{0,1}$, $m_{n-3}=B_{0,2}$, …, $m_{n-7}=B_{0,6}$, $m_{n-8}=B_{0,7}$,

     $m_{n-9}=B_{1,0}$, $m_{n-10}=B_{1,1}$, $m_{n-11}=B_{1,2}$, …, $m_{n-15}=B_{1,6}$, $m_{n-16}=B_{1,7}$,

     …,

     $m_7=B_{n/8\text{-}1,0}$, $m_6=B_{n/8\text{-}1,1}$, $m_5=B_{n/8\text{-}1,2}$, …, $m_1=B_{n/8\text{-}1,6}$, $m_0=B_{n/8\text{-}1,7}$
2. The procedure generates the remainder polynomial $r(x)$ given by the equation:

   $r(x) = [m(x) \cdot x^8]$ modulo $g(x)$

   where $r(x) = r_7x^7 + r_6x^6 + \ldots + r_0x^0$ and r_i are binary coefficients;
3. The Header and Data CRC are formed from the 8-bit vector representation of $r(x)$; the least significant bit b_0 of the CRC byte is coefficient r_7 of the highest power of x, while the most significant bit b_7 of the CRC byte is coefficient r_0 of the lowest power of x:

   $b_7=r_0$, $b_6=r_1$, $b_5=r_2$, $b_4=r_3$, $b_3=r_4$, $b_2=r_5$, $b_1=r_6$, $b_0=r_7$

NOTE The codeword $c(x) = m(x) \cdot x^8 + r(x)$ is formed by concatenating the bit vector representations of $m(x)$ and $r(x)$.

NOTE When a Galois version of a Linear Feedback Shift Register is used for CRC generation, its initial value is zero.

NOTE Example VHDL and C-code implementations of this CRC algorithm are included in sub-clause 6.10.

d. If the CRC generation procedure is applied to the bytes covered by the CRC *excluding* the CRC byte then the generated CRC can be compared directly with the expected CRC byte. If the generated and expected CRC bytes are equal then no errors have been detected; if they are different then an error has been detected.

e. If the CRC generation procedure is applied to the bytes covered by the CRC *including* the CRC byte then the output of the CRC generation procedure will be zero if no errors have been detected and non-zero if an error has been detected.

NOTE When the codeword $c^*(x)$ is input to the CRC generator then the remainder is the syndrome: $s(x) = [c^*(x) \cdot x^8]$ modulo $g(x)$. The codeword $c^*(x)$ is the concatenation of the Header or Data bytes covered by the CRC, followed by the CRC byte.

f. If the value of the data length field is zero, then the Data CRC shall be 0x00.

NOTE Read commands and write replies have no Data CRC field.

g. The CRC shall be calculated on the byte stream not the serial bit stream, since the RMAP protocol operates above the SpaceWire packet level (see ECSS-E50-12A).

NOTE The equivalent bit serial version takes the least-significant bit of each byte first and does not include data/control or parity bits, NULL, FCT or other non-data characters.

NOTE See sub-clause 6.10 for some examples of how the CRC is implemented along with some test patterns.

6.4. Write Command

6.4.1. Write command format

6.4.1.1. Fields

The write command shall contain the fields shown in Figure 6-1.

First byte transmitted

	Target SpW Address	....	Target SpW Address
Target Logical Address	Protocol Identifier	Instruction	Key
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Initiator Logical Address	Transaction Identifier (MS)	Transaction Identifier (LS)	Extended Address
Address (MS)	Address	Address	Address (LS)
Data Length (MS)	Data Length	Data Length (LS)	Header CRC
Data	Data	Data	Data
Data	...	...	Data
Data	Data CRC	EOP	

Last byte transmitted

Bits in Instruction Field

MSB						LSB
Reserved = 0	Command = 1	Write = 1	Verify data(1) Don't Verify (0)	Reply (1)/ No reply (0)	Increment (1)/ No inc (0)	Reply Address Length
Packet Type		Command				Reply Address Length

Figure 6-1 Write Command Format

6.4.1.2. Target SpaceWire Address field

The Target SpaceWire Address field shall be as defined in sub-clause 6.2.1.

6.4.1.3. Target Logical Address field

The Target Logical Address field shall be as defined in sub-clause 6.2.2.

6.4.1.4. Protocol Identifier field

The Protocol Identifier field shall be as defined in sub-clause 6.2.3.

6.4.1.5. Instruction field

6.4.1.5.1. Instruction field format

The Instruction field format shall be as defined in sub-clause 6.2.4.

6.4.1.5.2. Packet type field

The Packet Type field shall be 0b01 to indicate that this is a command.

6.4.1.5.3. Command field

a. The Write/Read bit shall be set (1) for a write command.

b. The Verify-Data-Before-Write bit shall be:

   1. Set (1) if the data is to be checked before it is written to memory, and
   2. Clear (0) otherwise.

c. The Reply bit shall be:

   1. Set (1) if a reply to the write command is required, and
   2. Clear (0) otherwise.

d. The Increment/No increment Address bit shall be:

1. Set (1) if data is written to sequential memory addresses.
2. Clear (0) if data is written to a single memory address.

6.4.1.5.4. Reply Address length field

The Reply Address Length field shall be set to the smallest number of 32-bit words that is able to contain the Reply SpaceWire Address from the target, back to the initiator of the command packet or some other node that is to receive the reply.

EXAMPLE If three Reply SpaceWire Address bytes are used then the Reply Address Length field is set to one (0b01).

6.4.1.6. Key field

The Key field shall be as defined in sub-clause 6.2.5.

6.4.1.7. Reply Address field

The Reply Address field shall be as defined in sub-clause 6.2.6.

6.4.1.8. Initiator Logical Address field

The Initiator Logical Address field shall be as defined in sub-clause 6.2.7.

6.4.1.9. Transaction Identifier field

The Transaction Identifier field format shall be as defined in sub-clause 6.2.8.

6.4.1.10. Extended Address field

a. The Extended Address field shall be as defined in sub-clause 6.2.9.

b. The Extended Address field shall hold the most-significant 8-bits of the starting memory address to be written to.

6.4.1.11. Address field

a. The Address field format shall be as defined in sub-clause 6.2.10.

b. The Address field shall hold the least-significant 32-bits of the starting memory address to which the data in a write command is written.

6.4.1.12. Data Length field

The Data Length field format shall be as defined in sub-clause 6.2.11.

NOTE This gives a maximum Data Length of 16 Megabytes -1 in a single write command. If a single byte is being written this field is set to one. If set to zero then no bytes are written to memory which can be used as a test transaction depending upon the implementation.

6.4.1.13. Header CRC field

The Header CRC field shall contain an 8-bit CRC as defined in sub-clauses 6.2.12 and 6.3.

6.4.1.14. Data field

The Data field shall contain zero or more bytes of data that are written into the memory of the target as defined in sub-clause 6.2.13.

6.4.1.15. Data CRC field

The Data CRC shall contain an 8-bit CRC as defined in sub-clauses 6.2.15 and 6.3.

6.4.1.16. EOP character

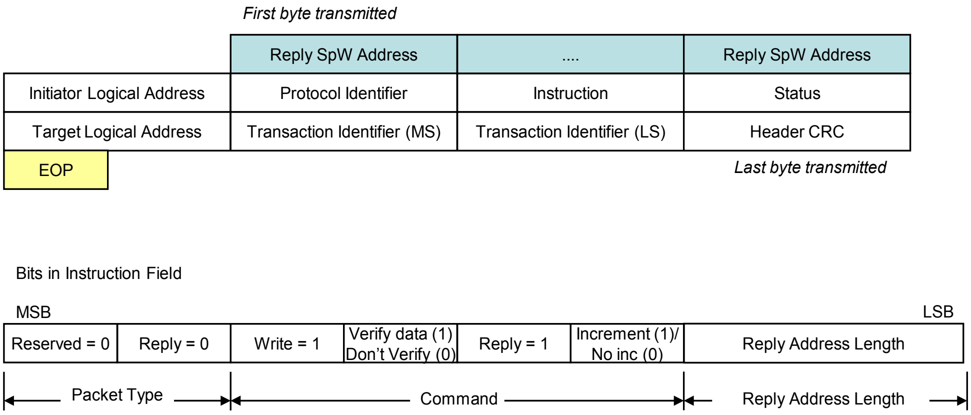
The end of the packet containing the write command shall be indicated by an EOP character.

6.4.2. Write reply format

6.4.2.1. Format

The format of the reply to a write command shall contain the fields shown in Figure 6-2.

NOTE A reply is sent by the target back to initiator of the write command or to some other node as defined by the Reply Address field if requested in the write command. The reply indicates the success or failure of the write command by the value in the Status field.



First byte transmitted

	Reply SpW Address	....	Reply SpW Address
Initiator Logical Address	Protocol Identifier	Instruction	Status
Target Logical Address	Transaction Identifier (MS)	Transaction Identifier (LS)	Header CRC
EOP			

Last byte transmitted

Bits in Instruction Field

MSB LSB

Reserved = 0	Reply = 0	Write = 1	Verify data (1) Don't Verify (0)	Reply = 1	Increment (1)/ No inc (0)	Reply Address Length
Packet Type		Command				Reply Address Length

Figure 6-2 Write Reply Format

6.4.2.2. Reply SpaceWire Address field

a. The Reply SpaceWire Address field shall comprise zero or more data characters which define how the reply is routed to the initiator or some other node.

b. The SpaceWire address in the Reply SpaceWire Address field shall be constructed from the Reply Address field in the command as detailed in sub-clause 6.2.6.

6.4.2.3. Initiator Logical Address field

The Initiator Logical Address field shall be as defined in sub-clause 6.2.7.

6.4.2.4. Protocol Identifier field

The Protocol Identifier field shall be as defined in sub-clause 6.2.3.

6.4.2.5. Instruction field

a. The Instruction field format shall be as defined in sub-clause 6.2.4.

b. The Packet Type field shall be 0b00 to indicate that the RMAP packet is a reply.

c. The Command field shall be set to the same value as in the Command field of the write command, sub-clause 6.4.1.5.3.

d. The Reply Address Length field shall be set to the same value as in the Reply Address Length field of the write command, sub-clause 6.4.1.5.4.

6.4.2.6. Status field

a. The Status field format shall be as defined in sub-clause 6.2.17.

b. The Status field shall contain:

1. 0x00 if the command executed successfully
2. A non-zero error code if there was an error with the write command as specified in sub-clause 6.7.

6.4.2.7. Target Logical Address field

The Target Logical Address field shall be set to either of:

a. The value of the Target Logical Address field of the write command, see sub-clause 6.4.1.3, or

b. A logical address of the target.

NOTE Normally these are the same.

6.4.2.8. Transaction Identifier field

The Transaction Identifier field shall be set to the same value as the Transaction Identifier in the write command, see sub-clause 6.4.1.9.

NOTE This is so that the initiator of the write command can associate the reply with the original write command.

6.4.2.9. Header CRC field

The Header CRC field shall contain an 8-bit CRC as defined in sub-clauses 6.2.12 and 6.3.

6.4.2.10. EOP character

The end of the Packet containing the write reply shall be indicated by an EOP character.

6.4.3. Write action

6.4.3.1. Overview

The normal sequence of actions for a write command is illustrated in Figure 6-3.

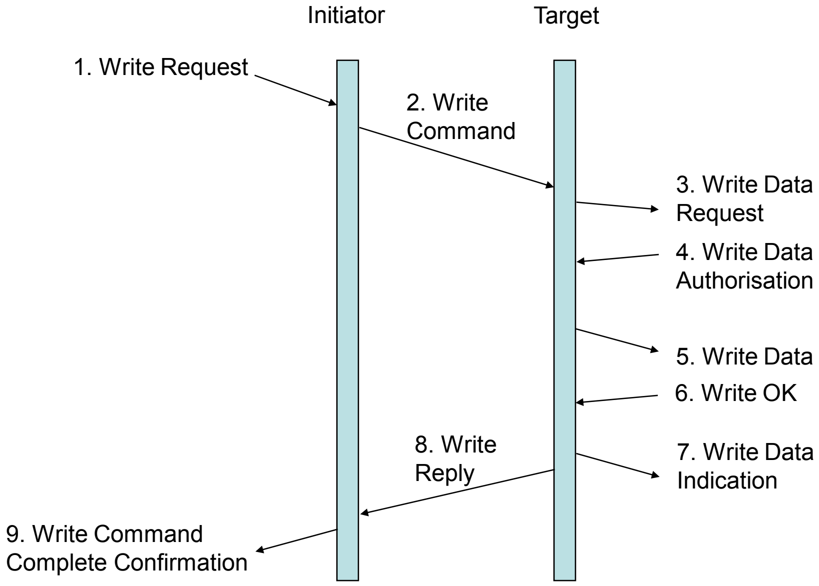


Figure 6-3 Write Command/Reply Sequence

6.4.3.2. Write request

a. The write command sequence shall begin when an initiator user application requests to perform a write operation (Write Request).

b. The initiator user application shall pass the following information to the initiator:

1. Target SpaceWire Address
2. Target Logical Address
3. Write command options
4. Key
5. Reply Address (if needed)
6. Initiator Logical Address
7. Transaction Identifier
8. Extended Address
9. Memory address
10. Data Length
11. Data

6.4.3.3. Write command

In response to the write request the initiator shall construct the write command including the Header CRC and Data CRC and send it across the SpaceWire network to the target (Write Command).

NOTE The Target SpaceWire Address and Target Logical Address are used to route the command packet to the target.

6.4.3.4. Write data request

a. When a Packet is received at the target and the Protocol Identifier field is 0x01 the packet shall be regarded as an RMAP packet.

b. If an EEP or EOP is received before the complete header including header CRC has been received:

1. The entire packet shall be discarded

2. The error information should be updated to reflect the "EEP" or "Early EOP" error if the target supports error information gathering,

3. A reply packet shall not be sent.

c. If an EEP is received immediately after the complete header including header CRC has been received:

1. The entire packet shall be discarded

2. The error information should be updated to reflect the "EEP" error if the target supports error information gathering,

3. A reply packet should not be sent.

d. When an RMAP packet is received at the target the Header CRC shall be checked.

e. When checking the Header CRC indicates an error in the header:

1. The entire packet shall be discarded

2. The error information should be updated to reflect the "Header CRC" error if the target supports error information gathering,

3. A reply packet shall not be sent.

NOTE The sequence of events that occurs when there is a CRC error in the header of the write command is illustrated in Figure 6-4.

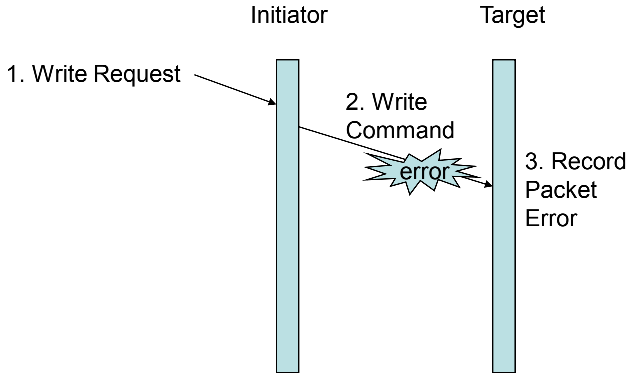


Figure 6-4 Write Command Header Error

f. When checking the Header CRC indicates no error present in the header:

1. If the Instruction field contains an unused packet type (0b10 or 0b11), the target:

(a) Shall discard the command packet,

(b) Should update the error information to reflect the "unused RMAP packet type or command code" error if the target supports error information gathering,

(c) Shall not send a reply.

(d) May send a reply containing an "unused RMAP packet type or command code" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields, if a reply has been requested (Reply bit set).

2. If the Instruction field contains an invalid command code as specified in Table 6-1, the target:

(a) Shall discard the command packet,

(b) Should update the error information to reflect the "unused RMAP packet type or command code" error if the target supports error information gathering,

(c) Shall return an "unused RMAP packet type or command code" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields, if a reply has been requested (Reply bit set).

3. If the Instruction field contains a write command (packet type 0b01 and a write command code) the target shall pass the following information to the target user application:

   (a) Target Logical Address

   (b) Instruction

   (c) Key

   (d) Initiator Logical Address

   (e) Transaction Identifier

   (f) Extended Address

   (g) Memory address

   (h) Data Length

6.4.3.5. Write data authorisation

a. The target user application shall be asked to authorise the write operation.

b. If the value of the Key is not the value expected by the target user application, the target:

   1. Shall discard the command packet,

   2. Should update the error information to reflect the "invalid key" error if the target supports error information gathering,

   3. Shall return an "invalid key" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

c. If the Target Logical Address is not a logical address recognised by the target user application, the target:

   1. Shall discard the command packet,

   2. Should update the error information to reflect the "invalid Target Logical Address" error if the target supports error information gathering,

   3. Shall return an "invalid Target Logical Address" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

d. If the command is not accepted by the target user application for any other reason, the target:

   1. Shall discard the command packet,

   2. Should update the error information to reflect the "RMAP command not implemented or not authorised" error if the target supports error information gathering,

   3. Shall return an "RMAP command not implemented or not authorised" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

   NOTE The target user application can reject the command for any reason it likes. For example the address is not 32-bit aligned, the Data Length is not a multiple of 4-bytes, or the address range falls partially or completely outside an acceptable memory address region.

   NOTE The sequence of events that occurs when a write command is not authorised is illustrated in Figure 6-5.

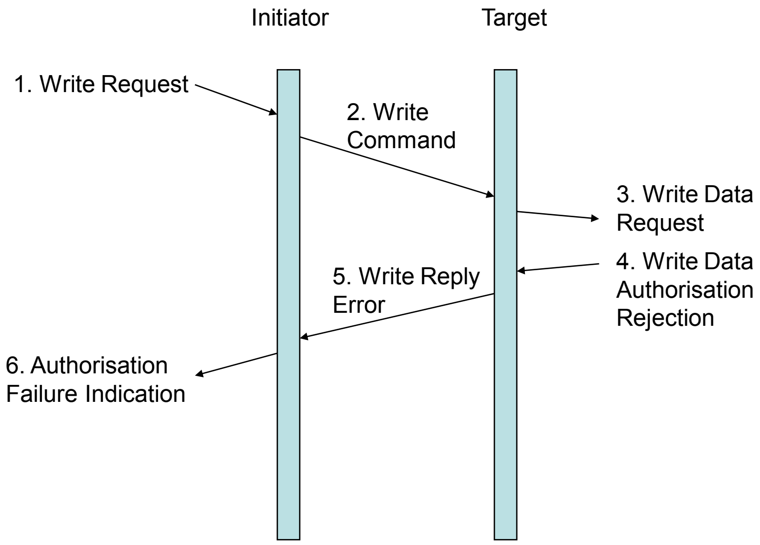


Figure 6-5 Write Data Authorisation Rejection

6.4.3.6. Write data

a. If authorisation is given by the target user application, the data contained in the write command shall be written into the memory location in the target specified by the Extended Address and Address fields (Write Data in Figure 6-3).

b. If the Verify-Data-Before-Write bit is set (1) in the command field of the header:

1. The data shall be buffered and checked using the Data CRC before it is written to memory.

   NOTE The size of the Verify-Data-Before-Write data buffer is implementation dependent.

2. If the data exceeds the available buffer space, the target:

   (a) Shall not write data to memory,

   (b) Should update the error information to reflect the "verify buffer overrun" error if the target supports error information gathering,

   (c) Shall return a "verify buffer overrun" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

3. If the Data CRC is correct and the amount of data matches the value of the data length field, the data shall be written from the buffer into memory.

4. If the Data CRC is in error the target:

   (a) Shall not write data to memory,

   (b) Should update the error information to reflect the "invalid Data CRC" error if the target supports error information gathering,

   (c) Shall return an "invalid Data CRC" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

5. If there is less data in the data field than specified in the Data Length field of the write command header when the EOP is reached, the target:

   (a) Shall not write data into memory,

   (b) Should indicate that an insufficient data error has occurred to the user application in the target,

(c) Should update the error information to reflect the insufficient data error if the target supports error information gathering,

(d) Shall return an "early EOP" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1),

6. If there is more data in the data field than specified in the Data Length field of the write command header, the target:

(a) Shall not write data into memory,

(b) Shall discard the rest of the packet,

(c) Should update the error information to reflect "too much data" error if the target supports error information gathering,

(d) Shall return a "too much data" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

7. If the packet ends in an EEP, the target:

(a) Shall not write data into memory,

(b) Should indicate that an "EEP" error has occurred to the user application in the target,

(c) Should update the error information to reflect the "EEP" error if the target supports error information gathering,

(d) Shall return an "EEP" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

c. If the Verify-Data-Before-Write bit is clear (0) in the command field of the header:

1. The data shall be written directly to memory without necessarily buffering and checking of the Data CRC beforehand.

2. If there is a Data CRC error the target shall:

(a) Update the error information to reflect the "invalid Data CRC" error if the target supports error information gathering,

(b) Return an "invalid Data CRC" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

NOTE If verify before write bit is clear (0) then the Data CRC error is reported after the data has been transferred to target memory.

NOTE The sequence of events that occurs when the Data CRC detects an error in the data field is illustrated in Figure 6-6.

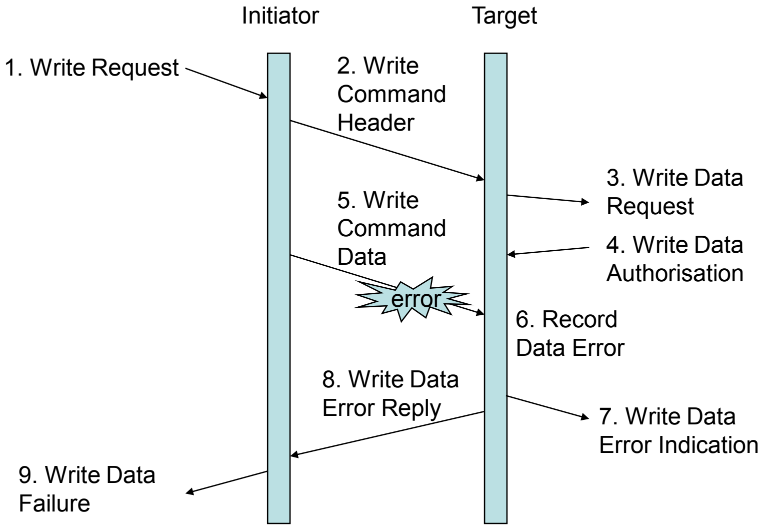


Figure 6-6 Write Command Data Error

3. If there is less data in the data field than specified in the Data Length field of the write command header when the EOP is reached, the target:

   (a) Shall stop transferring into target memory,

   (b) Should indicate that an "insufficient data" error has occurred to the user application in the target,

   (c) Should update the error information to reflect the "insufficient data" error if the target supports error information gathering,

   (d) Shall return an "early EOP" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1),

   NOTE If there is a Data CRC in the packet prior to the EOP then it can be incorrectly transferred into memory at the end of the data.

4. If there is more data in the data field than specified in the Data Length field of the write command header, the target:

   (a) Shall transfer the amount of data specified by the Data Length field of the write command header to memory,

   (b) Shall discard the rest of the packet,

   (c) Should update the error information to reflect "too much data" error if the target supports error information gathering,

   (d) Shall return a "too much data" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

5. If the packet ends in an EEP, the target:

   (a) Shall stop transferring data into target memory,

   (b) Should indicate that an EEP error has occurred to the user application in the target,

   (c) Should update the error information to reflect the "EEP" error if the target supports error information gathering,

   (d) Shall return an "EEP" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

d. If the Increment bit is clear (0) in the command field of the header, the memory address written to in the target shall remain constant i.e. all data in the write command is written to the same memory location.

e. If the Increment bit is set (1) in the command field of the header, the memory address written to in the target shall be incremented as determined by the target user application in order to access sequential memory locations i.e. the data in the write command is written to sequential memory locations.

NOTE The width of the memory locations is determined by the target user application. Byte addressing is not necessarily implied.

6.4.3.7. Write data indication

a. Once data has been written to memory the target user application should be informed that a write operation has taken place (Write Data Indication).

b. If data is not written to memory after authorisation has been given for the write to memory, the target user application should be informed that an error occurred.

6.4.3.8. Write reply

a. If the Reply bit in the command field is set (1) requesting a reply and the write command was executed successfully, the target shall send a reply packet with the status field set to 0x00 indicating that there was no error to the node specified by the Reply Address and Initiator Logical Address fields of the write command (Write Reply).

b. If the Reply bit in the command field is clear (0), the target shall not send a reply.

6.4.3.9. Write command complete confirmation

a. When the write reply is received at the initiator (or other node specified by the Reply Address and Initiator Logical Address), successful completion of the write request or its failure shall be indicated to the user application on that node (Write Complete Confirmation).

b. The Transaction Identifier shall be used to relate the reply to the command that caused the reply.

6.4.3.10. Write not OK

If the write operation to memory fails, the target:

a. Should stop writing to memory as soon as the memory error is detected,

b. Should update the error information to reflect the memory access error if the target supports error information gathering,

c. Shall return a “General” error as specified in sub-clause 6.7 to the node specified in the Reply Address field and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

NOTE This is a functional change to draft F to cope with the case when RMAP is connected to a memory bus where transactions can fail for any reason.

6.4.3.11. Corrupted write reply

If the write reply is corrupted or does not reach the initiator (or other node specified by the Reply Address) intact the initiator:

a. Shall discard the reply,

b. Should update the error information to reflect the invalid reply error, if the initiator or other node receiving the invalid reply supports error information gathering,

c. Should indicate an error to the user application in the node receiving the reply.

NOTE The sequence of events that occurs when a write reply error occurs is illustrated in Figure 6-7.

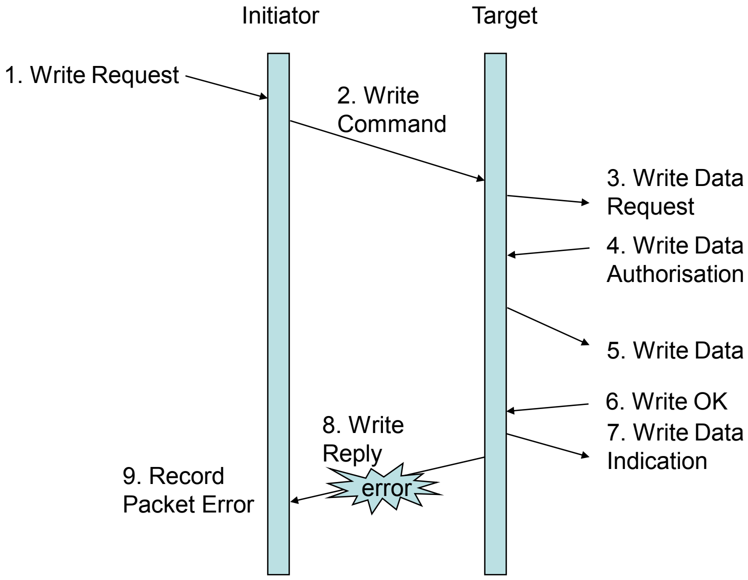


Figure 6-7 Write Reply Error

NOTE The data has been written into target memory and the target user application has been informed. The initiator application is informed when a write reply is received. It is not informed when no reply is received.

6.4.3.12. Invalid reply

When a reply is received by the initiator (or other node specified by the Reply Address) with the reserved bit in the instruction field set (1) or with the command/reply bit clear (0), the initiator:

a. Shall discard the reply,

b. Should update the error information to reflect the invalid reply error, if the initiator or other node receiving the invalid reply supports error information gathering.

6.5. Read Command

6.5.1. Read command format

6.5.1.1. Fields

The read command shall contain the fields shown in Figure 6-8.

First byte transmitted

	Target SpW Address	....	Target SpW Address
Target Logical Address	Protocol Identifier	Instruction	Key
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Initiator Logical Address	Transaction Identifier (MS)	Transaction Identifier (LS)	Extended Address
Address (MS)	Address	Address	Address (LS)
Data Length (MS)	Data Length	Data Length (LS)	Header CRC
EOP			

Last byte transmitted

Bits in Instruction Field

MSB						LSB
Reserved = 0	Command = 1	Read = 0	Verify data = 0	Reply = 1	Increment (1) / No inc (0)	Reply Address Length
Packet Type		Command				Reply Address Length

Figure 6-8 Read Command Format

6.5.1.2. Target SpaceWire Address field

The Target SpaceWire Address field shall be as defined in sub-clause 6.2.1.

6.5.1.3. Target Logical Address field

The Target Logical Address field shall be as defined in sub-clause 6.2.2.

6.5.1.4. Protocol Identifier field

The Protocol Identifier field shall be as defined in sub-clause 6.2.3.

6.5.1.5. Instruction field

6.5.1.5.1. Instruction field format

The Instruction field format shall be as defined in sub-clause 6.2.4.

6.5.1.5.2. Packet type field

The Packet Type field shall be 0b01 to indicate that this is a command.

6.5.1.5.3. Command field

a. The Write/Read bit shall be clear (0) for a read command.

b. The Verify-Data-Before-Write bit shall be clear (0) for a read command.

c. The Reply bit shall be set (1) for a read command.

d. The Increment/No increment Address bit shall be:

1. Set (1) if data is read from sequential memory addresses.
2. Clear (0) if data is read from a single memory address.

6.5.1.5.4. Reply Address length field

The Reply Address Length field shall be set to the smallest number of 32-bit words that is able to contain the Reply SpaceWire Address from the target, back to the initiator of the command packet or some other node that is to receive the reply.

EXAMPLE If six Reply SpaceWire Address bytes are used then the Reply Address Path Length field is set to two (0b10).

6.5.1.6. Key field

The Key field shall be as defined in sub-clause 6.2.5.

6.5.1.7. Reply Address field

The Reply Address field shall be as defined in sub-clause 6.2.6.

6.5.1.8. Initiator Logical Address field

The Initiator Logical Address field shall be as defined in sub-clause 6.2.7.

6.5.1.9. Transaction Identifier field

The Transaction Identifier field format shall be as defined in sub-clause 6.2.8.

6.5.1.10. Extended Address field

a. The Extended Address field shall be as defined in sub-clause 6.2.9.

b. The Extended Address field shall hold the most-significant 8-bits of the starting memory address to be read from.

6.5.1.11. Address field

a. The Address field format shall be as defined in sub-clause 6.2.10.

b. The Address field shall hold the least-significant 32-bits of the starting memory address from which data is read.

6.5.1.12. Data Length field

The Data Length field format shall be as defined in sub-clause 6.2.11.

NOTE This gives a maximum Data Length of 16 Megabytes - 1 in a single read command. If a single byte is being read this field is set to one. If set to zero then no bytes are read from memory which can be used as a test transaction depending upon the implementation.

6.5.1.13. Header CRC

The Header CRC field shall contain an 8-bit CRC as defined in sub-clauses 6.2.12 and 6.3.

6.5.1.14. EOP character

The end of the Packet containing the read command shall be indicated by an EOP character.

6.5.2. Read reply format

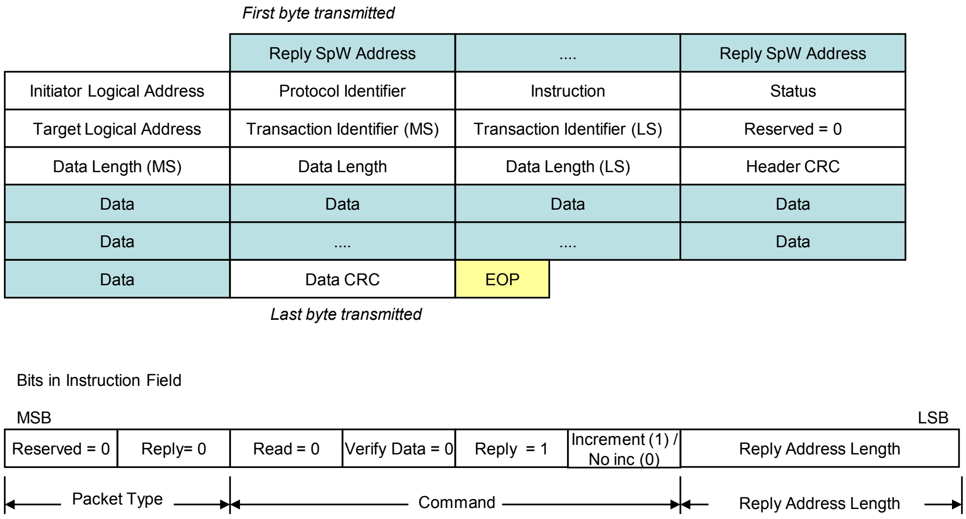
6.5.2.1. General

The read reply shall contain either:

1. the data that was read from the target, or
2. an error code indicating why data was not read, or
3. both data and an error code.

6.5.2.2. Format

The format of the reply to a read command shall be as in Figure 6-9.



First byte transmitted

	Reply SpW Address	....	Reply SpW Address
Initiator Logical Address	Protocol Identifier	Instruction	Status
Target Logical Address	Transaction Identifier (MS)	Transaction Identifier (LS)	Reserved = 0
Data Length (MS)	Data Length	Data Length (LS)	Header CRC
Data	Data	Data	Data
Data	....	....	Data
Data	Data CRC	EOP	

Last byte transmitted

Bits in Instruction Field

MSB						LSB
Reserved = 0	Reply= 0	Read = 0	Verify Data = 0	Reply = 1	Increment (1) / No inc (0)	Reply Address Length

Figure 6-9 Read Reply Format

6.5.2.3. Reply SpW Address

a. The Reply SpaceWire Address field shall comprise zero or more data characters which define how the reply is routed to the initiator or some other node.

b. The SpaceWire address in the Reply SpaceWire Address field shall be constructed from the Reply Address field in the command as detailed in sub-clause 6.2.6.

6.5.2.4. Initiator Logical Address field

The Initiator Logical Address field shall be as defined in sub-clause 6.2.7.

6.5.2.5. Protocol Identifier field

The Protocol Identifier field shall be as defined in sub-clause 6.2.3.

6.5.2.6. Instruction field

a. The Instruction field format shall be as defined in sub-clause 6.2.4.

b. The Packet Type field shall be 0b00 to indicate that RMAP packet is a reply.

c. The Command field shall be set to the same value as in the Command field of the read command, sub-clause 6.5.1.5.3.

d. The Reply Address Length field shall be set to the same value as in the Reply Address Length field of the read command, sub-clause 6.5.1.5.4.

6.5.2.7. Status field

a. The Status field format shall be as defined in sub-clause 6.2.17.

b. The Status field shall contain:

1. 0x00 if the command executed successfully
2. A non-zero error code if there was an error with the read command as specified in sub-clause 6.7.

6.5.2.8. Target Logical Address field

The Target Logical Address field shall be set to either of:

a. The value of the Target Logical Address field of the read command, see sub-clause 6.4.1.3, or

b. A logical address of the target.

NOTE Normally these are the same.

6.5.2.9. Transaction Identifier field

The Transaction Identifier field shall be set to the same value as the Transaction Identifier of the read command, see sub-clause 6.5.1.9.

NOTE This is so that the initiator of the read command can associate the reply and data in the reply with the original read command when the reply is sent to the initiator.

6.5.2.10. Data Length field

The Data Length field format shall be as defined in sub-clause 6.2.11.

6.5.2.11. Header CRC field

The Header CRC field shall contain a CRC as defined in sub-clauses 6.2.12 and 6.3.

6.5.2.12. Data field

a. The Data field shall contain the data that has been read from the memory of the target as defined in sub-clause 6.2.13.

b. The number of data bytes in the reply may be a different value from that indicated in the Data Length field in the command and reply, if fewer bytes are returned than requested.

6.5.2.13. Data CRC field

The Data CRC shall contain an 8-bit CRC as defined in sub-clauses 6.2.15 and 6.3.

6.5.2.14. EOP character

The end of the Packet containing the read reply shall be indicated by an EOP character.

6.5.3. Read action

6.5.3.1. Overview

The normal sequence of actions for a read command is illustrated in Figure 6-10.

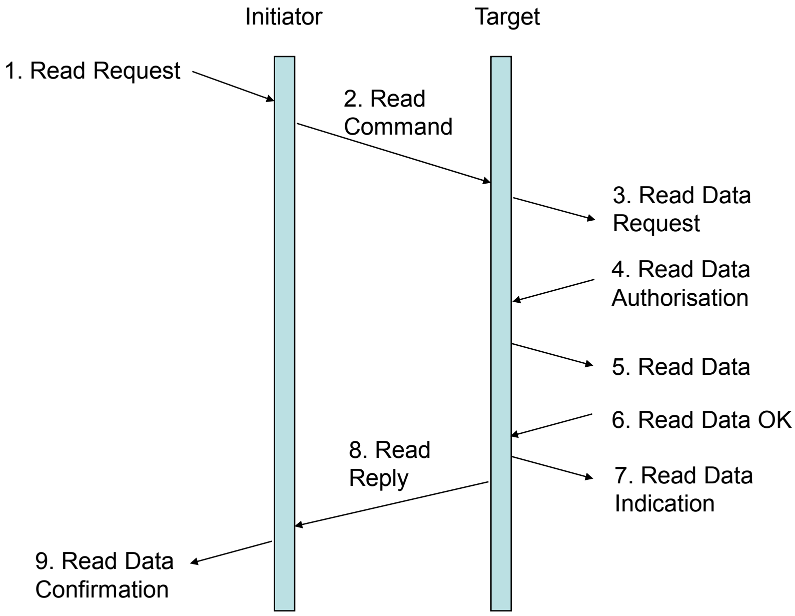


Figure 6-10 Read Command/Reply Sequence

6.5.3.2. Read Request

a. The read command sequence shall begin when an initiator user application requests to perform a read operation (Read Request).

b. The initiator user application shall pass the following information to the initiator:

1. Target SpaceWire Address
2. Target Logical Address
3. Read command options
4. Key
5. Reply Address
6. Initiator Logical Address
7. Transaction Identifier
8. Extended Address
9. Memory address
10. Data Length

6.5.3.3. Read command

In response to the read request the initiator shall construct the read command including the Header CRC and send it across the SpaceWire network to the target (Read Command).

NOTE The Target SpaceWire Address and Target Logical Address are used to route the command packet to the target.

6.5.3.4. Read data request

a. When a Packet is received at the target and the Protocol Identifier field is 0x01 the packet shall be regarded as an RMAP packet.

b. If an EEP or EOP is received before the complete header including header CRC has been received:

1. The entire packet shall be discarded,
2. The error information should be updated to reflect the “EEP” or “Early EOP” error if the target supports error information gathering,
3. A reply packet shall not be sent.

c. If an EEP is received immediately after the complete header including header CRC has been received:

1. The entire packet shall be discarded,
2. The error information should be updated to reflect the “EEP” error if the target supports error information gathering,
3. A reply packet should not be sent.

d. When an RMAP packet is received at the target the Header CRC shall be checked.

e. When checking the Header CRC indicates an error in the header:

1. The entire packet shall be discarded,
2. The error information shall be updated to reflect the “Header CRC” error if the target supports error information gathering,
3. A reply packet shall not be sent.

NOTE The sequence of events that occurs when there is a CRC error in the header of the read command is illustrated in Figure 6-11.

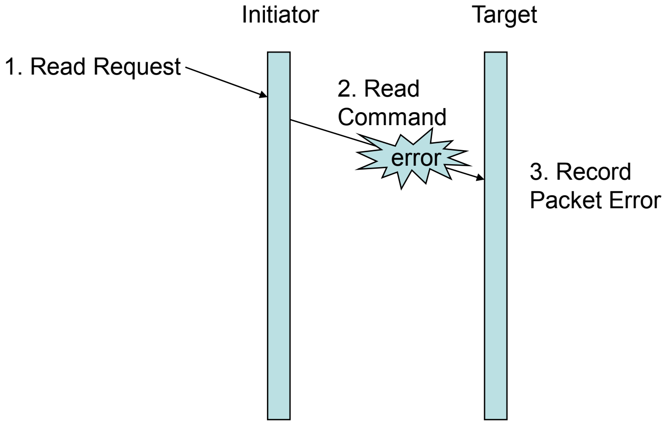


Figure 6-11 Read Command Header Error

f. When checking the Header CRC indicates no error present in the header:

1. If the Instruction field contains an unused packet type (0b10 or 0b11) the target:
   (a) Shall discard the command packet
   (b) Should update the error information to reflect the “unused RMAP packet type of command code” error if the target supports error information gathering,
   (c) Shall not send a reply.
   (d) May send a reply containing an “unused RMAP packet type or command code” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.
2. If the Instruction field contains an invalid command code as specified in Table 6-1, the target:
   (a) Shall discard the command packet,
   (b) Should update the error information to reflect the “unused RMAP packet type or command code” error if the target supports error information gathering,

(c) Shall return an "unused RMAP packet type or command code" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields, if a reply has been requested (Reply bit set).

3. If the Instruction field contains a read command (packet type 0b01 and a read command code) and if one or more data characters are received immediately after the complete header including header CRC the target:

   (a) Shall discard the remainder of the packet,

   (b) Shall not execute the read command,

   (c) Should update the error information to reflect the "too much data" error if the target supports error information gathering,

   (d) Shall return a "too much data" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

4. If the Instruction field contains a read command (packet type 0b01 and a read command code) the target shall pass the following information to the target user application:

   (a) Target Logical Address

   (b) Instruction

   (c) Key

   (d) Initiator Logical Address

   (e) Transaction Identifier

   (f) Extended Address

   (g) Memory address

   (h) Data Length

6.5.3.5. Read data authorisation

a. The target user application shall be asked to authorise the read operation.

b. If the value of the Key is not the value expected by the target user application, the target:

   1. Shall discard the command packet

   2. Should update the error information to reflect the "invalid key" error if the target supports error information gathering,

   3. Shall return an "invalid key" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).

c. If the Target Logical Address is not a logical address recognised by the target user application, the target:

   1. Shall discard the command packet

   2. Should update the error information to reflect the "invalid Target Logical Address" error if the target supports error information gathering,

   3. Shall return an "invalid Target Logical Address" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

d. If the command is not accepted by the target user application for any other reason, the target:

   1. Shall discard the command packet

   2. Should update the error information to reflect the "RMAP command not implemented or not authorised" error if the target supports error information gathering,

   3. Shall return an "RMAP command not implemented or not authorised" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

NOTE The target user application can reject the command for any reason it likes. For example the address is not 32-bit aligned, the Data Length is not a multiple of 4-bytes, or the address range falls partially or completely outside an acceptable memory address region.

NOTE The sequence of events that occurs when a read command is not authorised is illustrated in Figure 6-12.

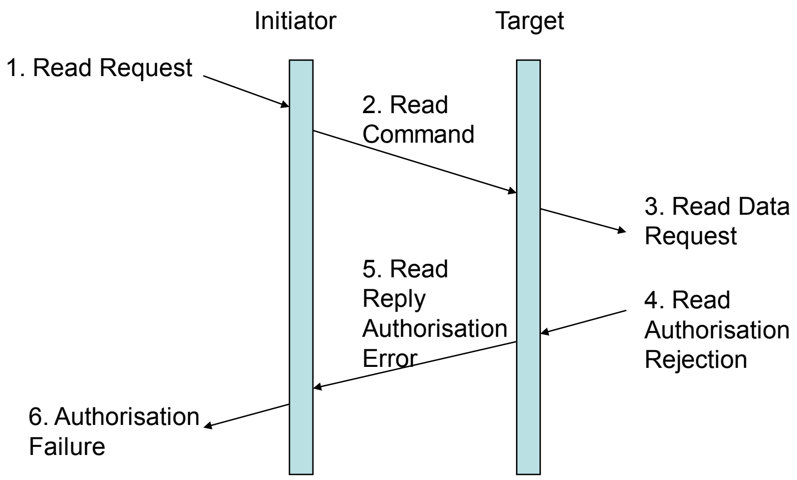


Figure 6-12 Read Authorisation Rejection

6.5.3.6. Read data

a. If authorisation is given by the target user application, data shall be read from the memory location in the target specified by the Extended Address and Address fields (Read Data).

b. If the Increment bit is clear (0) in the command field of the header, the memory address read from in the target shall remain constant i.e. all data is read from the same memory location.

c. If the Increment bit is set (1) in the command field of the header, the memory address read in the target shall be incremented as determined by the target user application in order to access sequential memory locations i.e. the data is read from sequential memory locations.

NOTE The width of the memory locations is determined by the target user application. Byte addressing is not necessarily implied.

6.5.3.7. Read data indication

a. Once data has been read from memory the target user application should be informed that a read operation has taken place (Read Data Indication).

b. If data is not read from memory after authorisation has been given for the read from memory, the target user application shall be informed that an error occurred.

6.5.3.8. Read reply

a. If the read command was executed successfully, the target shall send a reply packet to the node specified by the Reply Address and Initiator Logical Address fields of the read command (Read Reply).

b. The reply to a successful read command shall have:

1. The status field set to 0x00 indicting that there was no error
2. The Data Length field set to the amount of data read in bytes
3. The data field filled with the data read from the target memory.

6.5.3.9. Read data confirmation

a. When the read reply is received at the initiator (or other node specified by the Reply Address), successful completion of the read request shall be indicated to the user application on that node (Read Data Confirmation).

b. The Transaction Identifier shall be used to relate the reply to the command that caused the reply.

NOTE It is the responsibility of the initiator user application to read the data in the read reply once it has been informed that the data has been received.

6.5.3.10. Read not OK

If the read memory operation memory fails, the target:

a. Should stop reading from memory as soon as the memory error is detected,

b. Should update the error information to reflect the memory access error if the target supports error information gathering,

c. Shall either:

1. Append an EEP to the end of the data already sent in the reply to the initiator, or
2. Append an appropriate data CRC byte covering the data already sent in the reply to the initiator, followed by an EOP.

6.5.3.11. Read reply header error

If the reply from the read command arrives at the initiator (or other node specified by the Reply Address) with a Header CRC error, packet type error, or other error in the header, the receiving node:

a. Shall discard the entire packet containing the corrupted read reply,

b. Should update the error information to reflect the "Packet Error" error if the initiator (or other node receiving the reply) supports error information gathering.

NOTE The response to an error in the header of a read reply is illustrated in Figure 6-13.

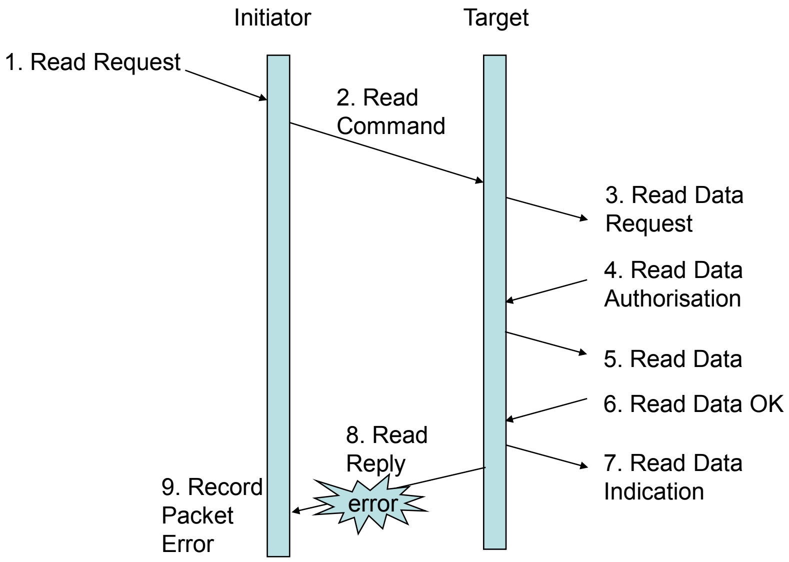


Figure 6-13 Read Reply Header Error

6.5.3.12. Read reply data error

If the header of the read reply packet is received intact by the initiator (or other node specified by the Reply Address) but the data field is corrupted as indicated by an incorrect data field length (too long or too short) or by a Data CRC error, the initiator:

a. Shall discard the reply,

b. Should update the error information to reflect the "invalid reply" error, if the initiator or other node receiving the invalid reply supports error information gathering,

c. Should indicate an error to the user application in the node receiving the reply (Read Data Failure).

NOTE The response to an error in the data field of a read reply is illustrated in Figure 6-14.

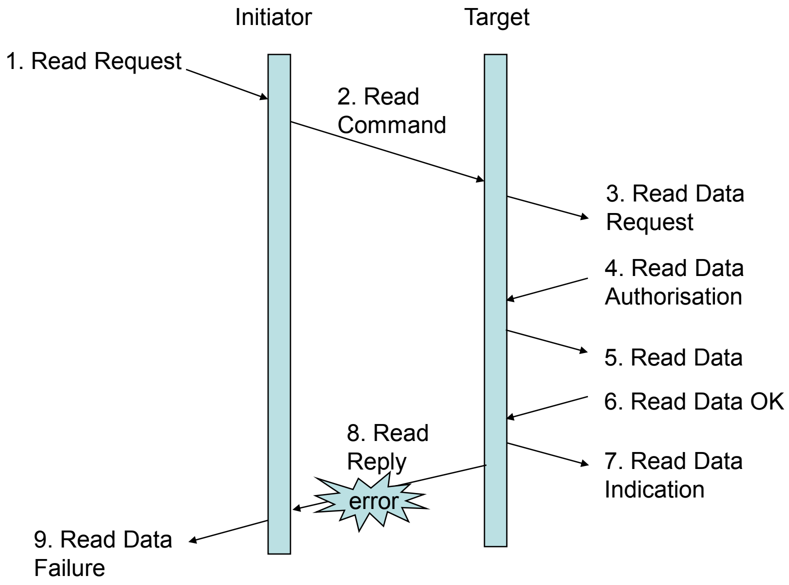


Figure 6-14 Read Reply Data Error

6.5.3.13. Invalid reply

When a reply is received by the initiator (or other node specified by the Reply Address) with the reserved bit in the instruction field set (1) or with the command/reply bit clear (0), the initiator:

a. Shall discard the reply,

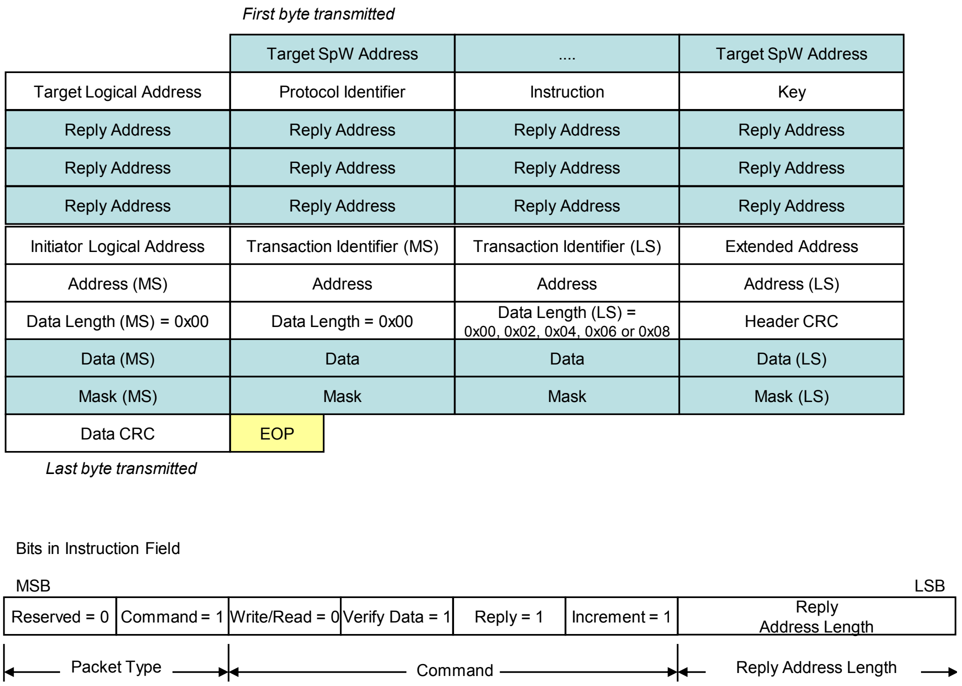
b. Should update the error information to reflect the "invalid reply" error, if the initiator or other node receiving the invalid reply supports error information gathering.

6.6. Read-Modify-Write Command

6.6.1. Read-modify-write command format

6.6.1.1. Fields

The read-modify-write command shall contain the fields shown in Figure 6-15.



First byte transmitted

	Target SpW Address	....	Target SpW Address
Target Logical Address	Protocol Identifier	Instruction	Key
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Initiator Logical Address	Transaction Identifier (MS)	Transaction Identifier (LS)	Extended Address
Address (MS)	Address	Address	Address (LS)
Data Length (MS) = 0x00	Data Length = 0x00	Data Length (LS) = 0x00, 0x02, 0x04, 0x06 or 0x08	Header CRC
Data (MS)	Data	Data	Data (LS)
Mask (MS)	Mask	Mask	Mask (LS)
Data CRC	EOP		

Last byte transmitted

Bits in Instruction Field

MSB						LSB
Reserved = 0	Command = 1	Write/Read = 0	Verify Data = 1	Reply = 1	Increment = 1	Reply Address Length

Figure 6-15 Read-Modify-Write Command Format

6.6.1.2. Target SpaceWire Address field

The Target SpaceWire Address field shall be as defined in sub-clause 6.2.1.

6.6.1.3. Target Logical Address field

The Target Logical Address field shall be as defined in sub-clause 6.2.2

6.6.1.4. Protocol Identifier field

The Protocol Identifier field shall be as defined in sub-clause 6.2.3.

6.6.1.5. Instruction field

6.6.1.5.1. Instruction field format

The Instruction field format shall be as defined in sub-clause 6.2.4.

6.6.1.5.2. Packet type field

The Packet Type field shall be 0b01 to indicate that this is a command.

6.6.1.5.3. Command field

a. The Write/Read bit shall be clear (0) for a read-modify-write command.

b. The Verify-Data-Before-Write bit shall be set (1) for a read-modify-write command .

> NOTE This is so that the data is verified before it is written to memory and also distinguishes a read-modify-write from a read command.

c. The Reply bit shall be set (1) for a read-modify-write command.

> NOTE The reply contains the data initially read from the memory in the target.

d. The "Increment / No Increment Address" bit shall be set (1) for a read-modify-write command.

NOTE This means that when read-modify-write is to be applied to more than one byte, the address is incremented if byte wide memory is being used. Note that the width of the memory word is determined by the target unit and can be any multiple of 8-bits.

6.6.1.5.4. Reply Address length field

The Reply Address Length field shall be set to the smallest number of 32-bit words that is able to contain the Reply SpaceWire Address from the target, back to the initiator of the command packet or some other node that is to receive the reply.

EXAMPLE If ten path Reply SpaceWire Address bytes are used then the Reply Address Length field is set to three (0b11).

6.6.1.6. Key field

The Key field shall be as defined in sub-clause 6.2.5.

6.6.1.7. Reply Address field

The Reply Address field shall be as defined in sub-clause 6.2.6.

6.6.1.8. Initiator Logical Address field

The Initiator Logical Address field shall be as defined in sub-clause 6.2.7

6.6.1.9. Transaction Identifier field

The Transaction Identifier field format shall be as defined in sub-clause 6.2.8.

6.6.1.10. Extended Address field

a. The Extended Address field shall be as defined in sub-clause 6.2.9.

b. The Extended Address field shall hold the most-significant 8-bits of the starting memory address to be read from.

6.6.1.11. Address field

a. The Address field format shall be as defined in sub-clause 6.2.10.

b. The Address field shall hold the least-significant 32-bits of the memory address to which the data in a read-modify-write command is read from and written to.

6.6.1.12. Data Length field

a. The Data Length field format shall be as defined in sub-clause 6.2.11.

b. The Data Length field shall contain the overall length, in bytes, of the data and mask fields i.e. the length of the data field plus the length of the mask field.

c. In a read-modify-write command the Data Length shall specify the size of the data field plus the size of the mask field sent in the command, which is twice the amount of data read and written.

EXAMPLE If a 2-byte word is written, then the Data Length is 0x04. There are two data bytes and two mask bytes in the command. Two bytes are read from memory and returned to the initiator. Two bytes are written combining the read data, the data from the command and the mask.

d. The Data Length shall only take on values of 0x00, 0x02, 0x04, 0x06 or 0x08, which correspond to the reading, modifying and writing of 0, 1, 2, 3, or 4 bytes of data respectively.

6.6.1.13. Header CRC field

The Header CRC field shall contain an 8-bit CRC as defined in sub-clauses 6.2.12 and 6.3.

6.6.1.14. Data field

a. The Data field shall contain the data that is combined with the mask and the data read from memory before the result is written into the memory of the target as defined in sub-clause 6.2.13.

b. The set of 0, 1, 2, 3 or 4 data bytes shall precede the corresponding set of 0, 1, 2, 3, or 4 mask bytes.

6.6.1.15. Mask field

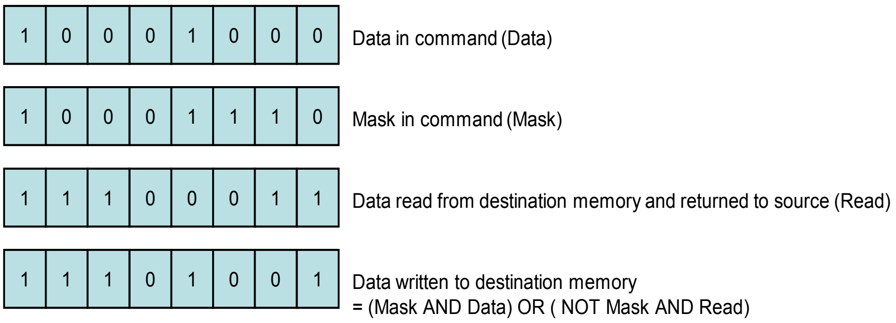
The Mask field shall be used by the target application to define how the data written to memory is formed.

NOTE The way the read data and mask are combined is application dependent.

EXAMPLE Data written can be selected on a bit by bit basis from the data sent in the command when the corresponding mask bit is set (1) or from the data read in the reply when the mask bit is clear (0).

Written Data = (Mask AND Command_Data) OR (NOT Mask AND Read_Data).

This example is illustrated in Figure 6-16. The target user application can implement different schemes for example test and set.



1	0	0	0	1	0	0	0	Data in command (Data)
1	0	0	0	1	1	1	0	Mask in command (Mask)
1	1	1	0	0	0	1	1	Data read from destination memory and returned to source (Read)
1	1	1	0	1	0	0	1	Data written to destination memory = (Mask AND Data) OR (NOT Mask AND Read)

Figure 6-16 Example Operation of Read-Modify-Write Command

6.6.1.16. Data CRC field

a. The Data CRC shall contain an 8-bit CRC as defined in sub-clauses 6.2.15 and 6.3.

b. The Data CRC shall cover both the data and the mask fields.

6.6.1.17. EOP character

The end of the Packet containing the read-modify-write command shall be indicated by an EOP character.

6.6.2. Read-modify-write reply format

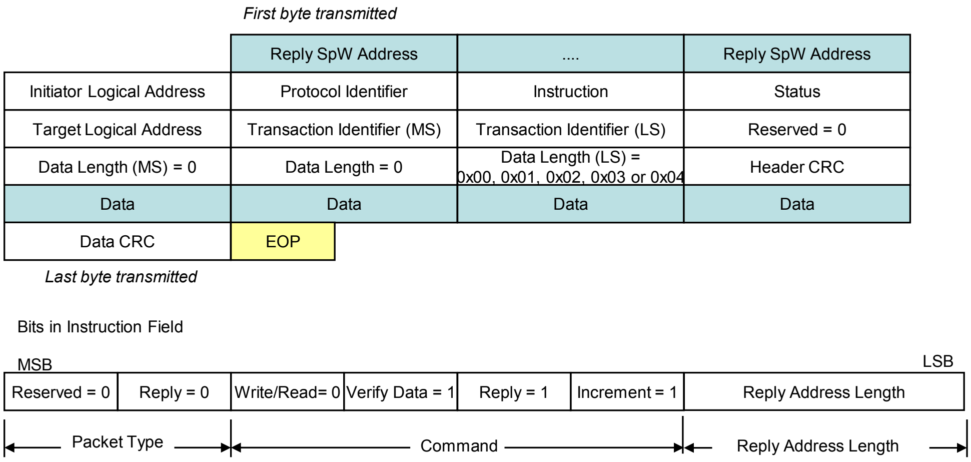
6.6.2.1. General

The read-modify-write reply shall contain either:

1. the data that was read from the target, or
2. an error code indicating why data was not read, or
3. both data and an error code.

6.6.2.2. Format

The format of the reply to a read-modify-write command shall be as in Figure 6-17.



First byte transmitted

	Reply SpW Address	....	Reply SpW Address
Initiator Logical Address	Protocol Identifier	Instruction	Status
Target Logical Address	Transaction Identifier (MS)	Transaction Identifier (LS)	Reserved = 0
Data Length (MS) = 0	Data Length = 0	Data Length (LS) = 0x00, 0x01, 0x02, 0x03 or 0x04	Header CRC
Data	Data	Data	Data
Data CRC	EOP		

Last byte transmitted

Bits in Instruction Field

MSB LSB

Reserved = 0	Reply = 0	Write/Read= 0	Verify Data = 1	Reply = 1	Increment = 1	Reply Address Length

Figure 6-17 Read-Modify-Write Reply Format

6.6.2.3. Reply SpaceWire Address

a. The Reply SpaceWire Address field shall comprise zero or more data characters which define how the reply is routed to the initiator or some other node.

b. The SpaceWire address in the Reply SpaceWire Address field shall be constructed from the Reply Address field in the command as detailed in sub-clause 6.2.6

6.6.2.4. Initiator Logical Address field

The Initiator Logical Address field shall be as defined in sub-clause 6.2.7.

6.6.2.5. Protocol Identifier field

The Protocol Identifier field shall be as defined in sub-clause 6.2.3.

6.6.2.6. Instruction field

a. The Instruction field format shall be as defined in sub-clause 6.2.4.

b. The Packet Type field shall be 0b00 to indicate that RMAP packet is a reply.

c. The Command field shall be set to the same value as in the Command field of the read-modify-write command, 6.6.1.5.3.

d. The Reply Address Length field shall be set to the same value as in the Reply Address Length field of the read-modify-write command, sub-clause 6.6.1.5.3.

6.6.2.7. Status field

a. The Status field format shall be as defined in sub-clause 6.2.17.

b. The Status field shall contain:

1. 0x00 if the command executed successfully
2. A non-zero error code if there was an error with the read-modify-write command as specified in sub-clause 6.7.

6.6.2.8. Target Logical Address field

The Target Logical Address field shall be set to either of:

a. The value of the Target Logical Address field of the write command, see sub-clause 6.4.1.3, or

b. A logical address of the target.

NOTE Normally these are the same.

6.6.2.9. Transaction Identifier field

The Transaction Identifier field shall be set to the same value as the Transaction Identifier of the read-modify-write command, see sub-clause 6.6.1.9.

NOTE This is so that the initiator of the read-modify-write command can associate the reply and data in the reply with the original read-modify-write command.

6.6.2.10. Data Length field

a. The Data Length field format shall be as defined in sub-clause 6.2.11.

b. The Data Length field shall contain the length, in bytes, of the data returned in the reply packet.

c. For a read-modify-write command the Data Length shall be 0, 1, 2, 3 or 4 only

NOTE The Data Length in the reply is a different value to the Data Length in the command since the Data Length in the command includes both data and mask.

6.6.2.11. Header CRC field

The Header CRC field shall contain a CRC as defined in sub-clauses 6.2.12 and 6.3.

6.6.2.12. Data field

The Data field shall contain the data that has been read from the memory of the target as defined in sub-clause 6.2.13.

NOTE The data length field in the reply is different to that in the command.

6.6.2.13. Data CRC field

The Data CRC shall contain an 8-bit CRC as defined in sub-clauses 6.2.15 and 6.3.

6.6.2.14. EOP character

The end of the Packet containing the read-modify-write reply shall be indicated by an EOP character.

6.6.3. Read-modify-write action

6.6.3.1. Overview

The normal sequence of actions for a read-modify-write command is illustrated in Figure 6-18.

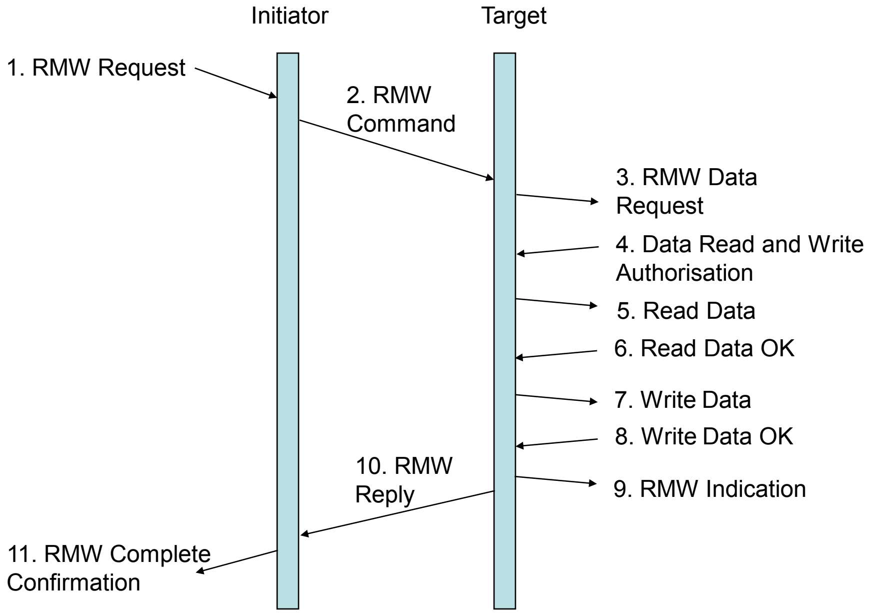


Figure 6-18 Read-Modify-Write Command/Reply Sequence

6.6.3.2. Read-modify-write request

a. The read-modify-write command sequence shall begin when an initiator user application requests to perform a read-modify-write operation (RMW Request).

b. The initiator user application shall pass the following information to the initiator:

1. Target SpaceWire Address
2. Target Logical Address
3. Read-modify-write command options
4. Key
5. Reply Address
6. Initiator Logical Address
7. Transaction Identifier
8. Extended Address
9. Memory address
10. Data Length
11. Data
12. Mask

6.6.3.3. Read-modify-write command

In response to the read-modify-write request the initiator shall construct the read-modify-write command including the Header CRC and Data CRC and send it across the SpaceWire network to the target (RMW Command).

NOTE The Target SpaceWire Address and Target Logical Address are used to route the command packet to the target.

6.6.3.4. Read-modify-write data request

a. When a Packet is received at the target and the Protocol Identifier field is 0x01 the packet shall be regarded as an RMAP packet.

b. If an EEP or EOP is received before the complete header including header CRC has been received:

   1. The entire packet shall be discarded,

   2. The error information should be updated to reflect the an “EEP” or “Early EOP” error if the target supports error information gathering,

   3. A reply packet shall not be sent.

c. If an EEP is received immediately after the complete header including header CRC has been received:

   1. The entire packet shall be discarded

   2. The error information should be updated to reflect the “EEP” error if the target supports error information gathering,

   3. A reply packet should not be sent.

d. When an RMAP packet is received at the target the Header CRC shall be checked.

e. When the Header CRC indicates an error in the header:

   1. The entire packet shall be discarded

   2. The error information should be updated to reflect the “Header CRC” error if the target supports error information gathering,

   3. A reply packet shall not be sent.

   NOTE The sequence of events that occurs when there is a CRC error in the header of the read-modify-write command is illustrated in Figure 6-19.

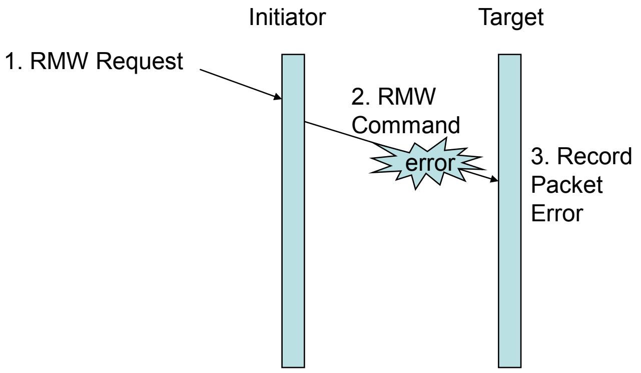


Figure 6-19 Read-Modify-Write Command Header Error

f. When checking the Header CRC indicates no error present in the header:

   1. If the Instruction field contains an unused packet type (0b10 or 0b11), the target:

      (a) Shall discard the command packet

      (b) Should update the error information to reflect the “unused RMAP packet type of command code” error if the target supports error information gathering,

      (c) Shall not send a reply.

      (d) May send a reply containing an “unused RMAP packet type or command code” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

   2. If the Instruction field contains an invalid command code as specified in Table 6-1, the target:

      (a) Shall discard the command packet,

(b) Should update the error information to reflect the “unused RMAP packet type or command code” error if the target supports error information gathering,

(c) Shall return an “unused RMAP packet type or command code” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

3. The data and mask shall be buffered and checked using the Data CRC before command authorisation is requested.

4. If the data exceeds the available buffer space, the target:

   (a) Shall not read data from and write data to memory,

   (b) Should update the error information to reflect the “verify buffer overrun” error if the target supports error information gathering,

   (c) Shall return a “verify buffer overrun” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

5. If the Data CRC is in error the target:

   (a) Shall not read data from and write data to memory,

   (b) Should update the error information to reflect the “invalid Data CRC” error if the target supports error information gathering,

   (c) Shall return an “invalid Data CRC” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

6. If there is less data in the data field than specified in the Data Length field of the read-modify-write command header when the EOP is reached, the target:

   (a) Shall not read data from and write data into memory,

   (b) Should indicate that an “insufficient data” error has occurred to the user application in the target,

   (c) Should update the error information to reflect the “insufficient data” error if the target supports error information gathering,

   (d) Shall return an “early EOP” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

7. If there is more data in the data field than specified in the Data Length field of the read-modify-write command header, then the target:

   (a) Shall not read data from and write data into memory,

   (b) Shall discard the rest of the packet,

   (c) Should update the error information to reflect “too much data” error if the target supports error information gathering,

   (d) Shall return a “too much data” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

8. If the value in the Data Length field is incorrect (i.e. is not 0, 2 ,4, 6 or 8), the target:

   (a) Shall not read data from and write data into memory,

   (b) Should update the error information to reflect “read-modify-write Data Length” error if the target supports error information gathering,

   (c) Shall return a “read-modify-write Data Length” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

   NOTE The sequence of events that occurs when there is an error in the data field of the read-modify-write command is illustrated in Figure 6-20.

9. If the packet ends in an EEP, the target:

   (a) Shall not write data into memory,

(b) Should indicate that an “EEP” error has occurred to the user application in the target,

(c) Should update the error information to reflect the “EEP” error if the target supports error information gathering,

(d) Shall return an “EEP” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

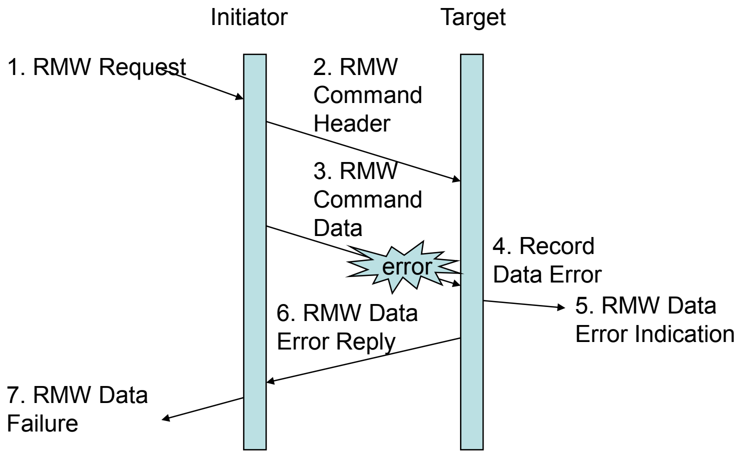


Figure 6-20 Read-Modify-Write Command Data Error

10. If the instruction field contains a RMW command and the Data CRC is correct and the amount of data in the data field is correct, the target shall pass the following information to the target user application for command authorisation:

(a) Target Logical Address

(b) Instruction

(c) Key

(d) Initiator Logical Address

(e) Transaction Identifier

(f) Extended Address

(g) Memory address

(h) Data Length

6.6.3.5. Read-modify-write authorisation

a. The target user application shall be asked to authorise the read-modify-write operation.

b. If the value of the Key is not the value expected by the target user application, the target:

1. Shall discard the command packet
2. Should update the error information to reflect the “invalid key” error if the target supports error information gathering,
3. Shall return an “invalid key” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

c. If the Target Logical Address is not a logical address recognised by the target user application, the target:

1. Shall discard the command packet
2. Should update the error information to reflect the “invalid Target Logical Address” error if the target supports error information gathering,
3. Shall return an “invalid Target Logical Address” error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

d. If the command is not accepted by the target user application for any other reason, the target:

1. Shall discard the command packet
2. Should update the error information to reflect the "RMAP command not implemented or not authorised" error if the target supports error information gathering,
3. Shall return an "RMAP command not implemented or not authorised" error as specified in sub-clause 6.7 to the node specified in the Reply Address and Initiator Logical Address fields.

NOTE The target user application can reject the command for any reason it likes.

EXAMPLE If the read operation is acceptable, but the write operation is not acceptable, due to for instance a write protected memory, then the command is not authorised by the target user application when the read-modify-write data request is made.

NOTE The sequence of events that occurs when a read-modify-write command is not authorised is illustrated in Figure 6-21.

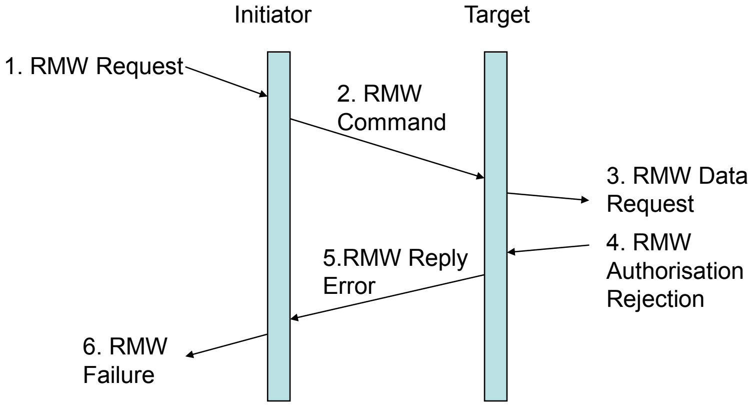


Figure 6-21 Read-Modify-Write Authorisation Rejection

6.6.3.6. Read data

a. If the data to be written does not contain any errors and authorisation is given by the target user application, data shall be read from the memory location in the target specified by the Extended Address and Address fields (Read Data).

b. The memory address read from in the target shall be incremented as determined by the target user application in order to access sequential memory locations i.e. the data is read from sequential memory locations.

NOTE The width of the memory locations is determined by the target user application. Byte addressing is not necessarily implied.

6.6.3.7. Write data

a. The data to be written to the memory locations shall be calculated from the data read from memory and the data and mask fields of the read-modify-write command.

NOTE The way in which the data read from target memory is combined with the data and mask values in the command is application dependent.

b. The new data shall be written to the memory location(s) that was previously read.

6.6.3.8. Read-modify write data indication

a. Once data has been read and written to memory the user application running on the target should be informed that a read-modify-write operation has taken place (RMW Indication).

b. If data is not written to memory after authorisation has been given for the read-modify-write to memory, the target user application should be informed that an error occurred.

6.6.3.9. Read-modify-write reply

a. If the read-modify-write command was executed successfully, the target shall send a reply packet to the node specified by the Reply Address and Initiator Logical Address fields of the read command (RMW Reply).

b. The reply to a successful read-modify-write command shall have:

   1. The status field set to 0x00 indicting that there was no error
   2. The Data Length field set to the amount of data read in bytes
   3. The data field filled with the data read from the target memory.

6.6.3.10. Read-modify-write complete confirmation

a. When the read-modify-write reply is received at the initiator (or other node specified by the Reply Address), successful completion of the read-modify-write request shall be indicated to the user application on that node (RMW Complete Confirmation).

b. The Transaction Identifier shall be used to relate the reply to the command that caused the reply.

NOTE It is the responsibility of the initiator user application to read the data in the read reply once it has been informed that the data has been received.

6.6.3.11. Read and Write not OK

If the read or write operations to memory fails, the target:

a. Should stop reading to or writing from memory as soon as the memory error is detected,

b. Should update the error information to reflect the "memory access" error if the target supports error information gathering,

c. Shall either:

   1. Append an EEP to the end of the data sent in the reply to the initiator, or
   2. Append an appropriate data CRC byte covering the data sent in the reply to the initiator, followed by an EOP.

NOTE In this case the data length field in the reply will contain the amount of data requested which will be different to the amount of data returned in the data field of the reply.

6.6.3.12. Read-modify-write reply header error

If the reply from the read-modify-write command arrives at the initiator (or other node) with a Header CRC error, packet type error, or other error in the header, the receiving node shall:

a. Shall discard the entire packet containing the corrupted read-modify-write reply,

b. Should update the error information to reflect the "Packet Error" error if the initiator (or other node receiving the reply) supports error information gathering.

NOTE The response to an error in the header of a read-modify-write reply is illustrated in Figure 6-22. The data has been correctly read from target memory, modified using the mask information and the result written back into memory. The target application has been informed. The read-modify-write reply that is sent back to the initiator is corrupted.

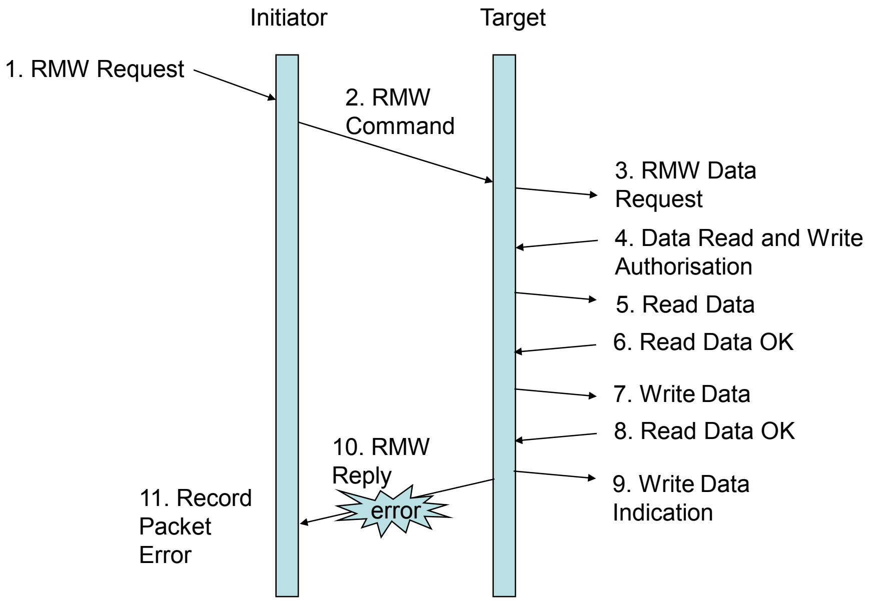


Figure 6-22 Read-Modify-Write Reply Error

6.6.3.13. Read-modify-write reply data error

If the header of the read-modify-write reply packet is received intact by the initiator (or other node specified by the Reply Address) but the data field is corrupted as indicated by an incorrect data field length (too long or too short) or by a Data CRC error, the initiator:

a. Shall discard the reply,

b. Should update the error information to reflect the "invalid reply" error, if the initiator or other node receiving the invalid reply supports error information gathering,

c. Should indicate an error to the user application in the node receiving the reply (Read-Modify-Write Data Failure).

NOTE The response to an error in the data field of a read reply is illustrated in Figure 6-23.

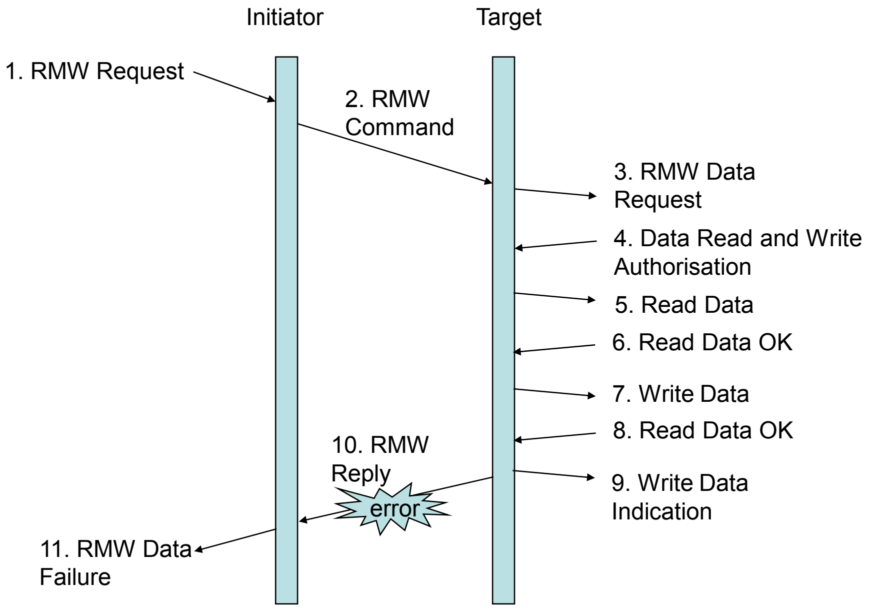


Figure 6-23 RMW Reply Data Error

6.6.3.14. Invalid reply

When a reply is received by the initiator (or other node specified by the Reply Address) with the reserved bit in the instruction field set (1) or with the command/reply bit clear (0), the initiator:

a. Shall discard the reply,

b. Should update the error information to reflect the "invalid reply" error, if the initiator or other node receiving the invalid reply supports error information gathering.

6.7. Error and status codes

6.7.1. Error and status codes

a. The set of error and status codes that shall be used are listed in Table 6-4.

b. If a command executes successfully, then the Error Code 0 shall be used in the Status field of any reply.

c. If there is an error with the command, then a suitable error code as defined in Table 6-4 shall be used in the Status field of any reply.

Table 6-4 Error and Status Codes

Error Code	Error	Error Description	Applicability		
			Write	Read	RMW
0	Command executed successfully		X	X	X
1	General error code	The detected error does not fit into the other error cases or the node does not support further distinction between the errors	X	X	X
2	Unused RMAP Packet Type or Command Code	The Header CRC was decoded correctly but the packet type is reserved or the command is not used by the RMAP protocol.	X	X	X
3	Invalid key	The Header CRC was decoded correctly but the device key did not match that expected by the target user application.	X	X	X
4	Invalid Data CRC	Error in the CRC of the data field	X		X
5	Early EOP	EOP marker detected before the end of the data.	X	X	X
6	Too much data	More than the expected amount of data in a command has been received.	X	X	X
7	EEP	EEP marker detected immediately after the header CRC or during the transfer of data and Data CRC or immediately thereafter. Indicates that there was a communication failure of some sort on the network.	X	X	X
8	Reserved	Reserved			
9	Verify buffer overrun	The verify before write bit of the command was set so that the data field was buffered in order to verify the Data CRC before transferring the data to target memory. The data field was longer than able to fit inside the verify buffer resulting in a buffer overrun. Note that the command is not executed in this case.	X		X
10	RMAP Command not implemented or not authorised	The target user application did not authorise the requested operation. This may be because the command requested has not been implemented.	X	X	X
11	RMW Data Length error	The amount of data in a RMW command is invalid (0x01, 0x03, 0x05, 0x07 or greater than 0x08).			X
12	Invalid Target Logical Address	The Header CRC was decoded correctly but the Target Logical Address was not the value expected by the target.	X	X	X
13-255	Reserved	All unused error codes are reserved			

d. When one or more errors arise that mean more than one error code is applicable it shall be application dependent as to which of the relevant error codes is sent.

e. The specific error returned may vary depending on the specific application.

6.8. Partial Implementations of RMAP

6.8.1. Limited functionality nodes

6.8.1.1. Types of RMAP node

It shall be possible to implement nodes which are:

1. Initiators,
2. Targets, or
3. Both initiators and targets.

6.8.1.2. Initiator only node

a. An initiator shall be able to send RMAP commands and receive RMAP replies.

b. If an initiator only node receives a command, the command shall be discarded.

c. The "Command Received by Initiator" error should be recorded.

6.8.1.3. Target only node

a. A target shall be able to receive RMAP commands and send replies.

b. If a target only node receives a reply, the reply shall be discarded.

c. The "Reply Received by Target" error should be recorded.

6.8.2. Partial implementations

a. Partial implementations of RMAP may be permitted where only some commands or command options are supported.

EXAMPLE A unit that supports write and read command but does not implement the read-modify-write command.

b. If the target user application is passed a command or a command with options that it does not support then it shall not authorise the command.

c. If a reply has been requested then the RMAP command not implemented or not authorised error shall be sent back to the initiator (or other node).

NOTE See sub-clause 6.7

6.9. RMAP Conformance

6.9.1. Overview

Several SpaceWire RMAP compatible subsets can be identified each of which implements only a part of the SpaceWire RMAP standard:

- RMAP Write Command / Target only
- RMAP Read Command / Target only
- RMAP Read-Modify-Write Command / Target only
- RMAP Read and Write / Target only

- RMAP Initiator
- RMAP Initiator and Target

Corresponding subsets of the SpaceWire RMAP standard are defined to which implementations can claim conformance. The form of the conformance statement to use is the one given by the appropriate subset definition in the following sub-clauses.

An RMAP compliant product can implement one or more of these subsets.

6.9.2. RMAP Partial implementations

6.9.2.1. Target Only

6.9.2.2. Initiator Only

6.9.2.3. RMAP Write Command

a. A product that uses the following conformance statement *“This product conforms to the SpaceWire RMAP Write specification of the ESA SpaceWire Protocols Standard (ECSS-E-50-11A).”* shall meet the RMAP Write specifications listed in Table 6-5.

Table 6-5: SpaceWire RMAP Write Command

Relevant clauses or sub-clauses	Title
5	Protocol Identifier
a	Write Command
6.7	Error Codes

b. The supplier of the RMAP equipment shall provide a table detailing the write characteristics of the RMAP implementation.

NOTE An example of the required table is given in Table 6-6.

Table 6-6 Example of Write Command Product Characteristics

Write Command			
Action	Supported	Maximum Data Length (bytes)	Non-aligned access accepted
8-bit write	No	-	-
16-bit write	No	-	-
32-bit write	Yes	12	No
64-bit write	No	-	-
Verified write	Yes	4	No
Word or byte address	Word address 32-bit aligned		
Endian order	Little endian i.e. first byte received goes in least significant byte of memory location		
Accepted logical addresses	0xFE at power-on 0x42-0x51 after initialisation		
Target logical address in reply	What was in command		
Accepted keys	0x20		
Accepted address ranges	0x00 0000 0000 – 0x00 0000 001C		
Address incrementation	Incrementing address only		
Status codes returned	All		

6.9.2.4. RMAP Read Command

a. A product that uses the following conformance statement *"This product conforms to the SpaceWire RMAP Read specification of the ESA SpaceWire Protocols Standard (ECSS-E-50-11A)."* shall meet the RMAP Read specifications listed in Table 6-7.

Table 6-7: SpaceWire RMAP Read Command

Relevant clauses or sub-clauses	Title
5	Protocol Identifier
6.6	Read Command
6.7	Error Codes

b. The supplier of the RMAP equipment shall provide a table detailing the read characteristics of the RMAP implementation.

NOTE An example of the required table is given in Table 6-8.

Table 6-8 Example Read Command Product Characteristics

Read Command			
Action	Supported	Maximum Data Length (bytes)	Non-aligned access accepted
8-bit read	No	-	-
16-bit read	No	-	-
32-bit read	Yes	12	No
64-bit read	No	-	-
Word or byte address	Word address 32-bit aligned		
Endian order	Big endian i.e. the most significant byte of the memory location is returned as the first byte		
Accepted logical addresses	0xFE at power-on 0x42 after initialisation		
Target logical address in reply	Logical address of target		
Accepted keys	0x20		
Accepted address ranges	0x00 0000 0000 – 0x00 0000 001C 0x00 0000 0020 – 0x00 0000 003C		
Address incrementation	Incrementing address only		
Status codes returned	All		

6.9.2.5. RMAP Read-Modify-Write Command

a. A product that uses the following conformance statement *“This product conforms to the SpaceWire RMAP Read-Modify-Write specification of the ESA SpaceWire Protocols Standard (ECSS-E-50-11).”* shall meet the RMAP Read-ModifyWrite specification listed in Table 6-9.

Table 6-9: SpaceWire RMAP Read-Modify-Write Command

Relevant clauses or sub-clauses	Title
5	Protocol Identifier
6.5	Read-Modify-Write Command
6.7	Error Codes

b. The supplier of the RMAP equipment should provide a table detailing the characteristics of the RMAP implementation.

NOTE An example of the required table is given in Table 6-10.

Table 6-10 Example Read-Modify-Write Command Product Characteristics

Read-Modify-Write Command			
Action	Supported	Maximum Data Length (bytes)	Non-aligned access accepted
8-bit read-modify-write	No	-	-
16-bit read-modify-write	No	-	-
32-bit read-modify-write	Yes	4	No
64-bit read-modify-write	No	-	-
Word or byte address	Byte address 32-bit word		
Endian order	Little endian i.e. first byte received goes in least significant byte of memory location		
Accepted logical addresses	0xFE at power-on 0x42 after initialisation		
Target logical address in reply	What was in command		
Accepted keys	0x20		
Accepted address ranges	0x00 0000 0000 – 0x00 0000 001C		
Status codes returned	All		

6.10. Annex RMAP CRC Implementation (informative)

In this annex example implementations of the CRC used by RMAP are provide in VHDL and C-code.

6.10.1. VHDL implementation of RMAP CRC

```
-----------------------------------------------------------------------------
   -- Cyclic Redundancy Code (CRC) for Remote Memory Access Protocol (RMAP)
   -----------------------------------------------------------------------------
   -- Purpose:
   --    Given an intermediate SpaceWire RMAP CRC byte value and an RMAP header
   --    or data byte, return an updated RMAP CRC byte value.
   --
   -- Parameters:
   --    INCRC(7:0)  - The intermediate  RMAP CRC byte value.
   --    INBYTE(7:0) - The RMAP Header or Data byte.
   --
   -- Return value:
   --    OUTCRC(7:0) - The updated RMAP CRC byte value.
   --
   -- Description:
   --    One-to-many implementation: Galois version of LFSR (reverse CRC).
   --
   --         +---+   +---+   +---+   +---+   +---+   +---+    +---+    +---+
   -- out <-+-| 7 |<--| 6 |<--| 5 |<--| 4 |<--| 3 |<--| 2 |<-X-| 1 |<-X-| 0 |<-+
   --       | +---+   +---+   +---+   +---+   +---+   +---+  ^ +---+  ^ +---+  |
   --       |                                                |        |        |
   --       v                                                |        |        |
   -- in -->X------------------------------------------------+--------+--------+
   --    x**8     x**7    x**6    x**5    x**4    x**3    x**2     x**1     x**0
   --
   --    Generator polynomial: g(x) = x**8 + x**2 + x**1 + x**0
   --
   -- Notes:
   --    The INCRC input CRC value must have all bits zero for the first INBYTE.
   --
   --    The first INBYTE must be the first Header or Data byte covered by the
   --    RMAP CRC calculation. The remaining bytes must be supplied in the RMAP
   --    transmission/reception byte order.
   --
   --    If the last INBYTE is the last Header or Data byte covered by the RMAP
   --    CRC calculation then the OUTCRC output will be the RMAP CRC byte to be
   --    used for transmission or to be checked against the received CRC byte.
   --
   --    If the last INBYTE is the Header or Data CRC byte then the OUTCRC
   --    output will be zero if no errors have been detected and non-zero if
--    an error has been detected.
--
--    Each byte is inserted in or extracted from a SpaceWire packet without
--    the need for any bit reversal or similar manipulation. The SpaceWire
--    packet transmission and reception procedure does the necessary bit
--    ordering when sending and receiving Data Characters (see ECSS-E-50-12A
--    clause 7.2).
--
--    SpaceWire data is sent/received Least Significant Bit (LSB) first:
--       INBYTE(0) is the LSB of SpaceWire data byte (sent/received first)
--       INCRC(0)  is the LSB of SpaceWire data byte (sent/received first)
--
-----------------------------------------------------------------------------
function RMAP_CalculateCRC (
   constant INCRC:   in Std_Logic_Vector(7 downto 0);
   constant INBYTE:  in Std_Logic_Vector(7 downto 0))
   return               Std_Logic_Vector is  -- Same range as the two inputs

   -- This variable is to hold the output CRC value.
   variable OUTCRC:     Std_Logic_Vector(7 downto 0);

   -- Internal Linear Feedback Shift Register (LFSR). Note that the
   -- vector indices correspond to the powers of the Galois field
   -- polynomial g(x) which are NOT the same as the indices of the
   -- SpaceWire data byte.
   variable LFSR:       Std_Logic_Vector(7 downto 0);
begin
   -- External to internal bit-order reversal to match indices.
   for i in 0 to 7 loop
      LFSR(7-i) := INCRC(i);
   end loop;

   -- Left shift LFSR eight times feeding in INBYTE bit 0 first (LSB).
   for j in 0 to 7 loop
      LFSR(7 downto 0) := (LFSR(6 downto 2)) &
                          (INBYTE(j) xor LFSR(7) xor LFSR(1)) &
                          (INBYTE(j) xor LFSR(7) xor LFSR(0)) &
                          (INBYTE(j) xor LFSR(7));
   end loop;

   -- Internal to external bit-order reversal to match indices.
   for i in 0 to 7 loop
      OUTCRC(7-i) := LFSR(i);
   end loop;

   -- Return the updated RMAP CRC byte value.
   return OUTCRC;
end function RMAP_CalculateCRC;
```

6.10.2. C-code implementation of RMAP CRC

```
/*
 * The local look-up table used to calculate the updated RMAP CRC
 * byte from the intermediate CRC byte and the input byte.
 */
static const unsigned char RMAP_CRCTable[] = {
   0x00, 0x91, 0xe3, 0x72, 0x07, 0x96, 0xe4, 0x75,
   0x0e, 0x9f, 0xed, 0x7c, 0x09, 0x98, 0xea, 0x7b,
   0x1c, 0x8d, 0xff, 0x6e, 0x1b, 0x8a, 0xf8, 0x69,
   0x12, 0x83, 0xf1, 0x60, 0x15, 0x84, 0xf6, 0x67,
   0x38, 0xa9, 0xdb, 0x4a, 0x3f, 0xae, 0xdc, 0x4d,
   0x36, 0xa7, 0xd5, 0x44, 0x31, 0xa0, 0xd2, 0x43,
   0x24, 0xb5, 0xc7, 0x56, 0x23, 0xb2, 0xc0, 0x51,
   0x2a, 0xbb, 0xc9, 0x58, 0x2d, 0xbc, 0xce, 0x5f,
   0x70, 0xe1, 0x93, 0x02, 0x77, 0xe6, 0x94, 0x05,
   0x7e, 0xef, 0x9d, 0x0c, 0x79, 0xe8, 0x9a, 0x0b,
   0x6c, 0xfd, 0x8f, 0x1e, 0x6b, 0xfa, 0x88, 0x19,
   0x62, 0xf3, 0x81, 0x10, 0x65, 0xf4, 0x86, 0x17,
   0x48, 0xd9, 0xab, 0x3a, 0x4f, 0xde, 0xac, 0x3d,
   0x46, 0xd7, 0xa5, 0x34, 0x41, 0xd0, 0xa2, 0x33,
   0x54, 0xc5, 0xb7, 0x26, 0x53, 0xc2, 0xb0, 0x21,
   0x5a, 0xcb, 0xb9, 0x28, 0x5d, 0xcc, 0xbe, 0x2f,
   0xe0, 0x71, 0x03, 0x92, 0xe7, 0x76, 0x04, 0x95,
   0xee, 0x7f, 0x0d, 0x9c, 0xe9, 0x78, 0x0a, 0x9b,
   0xfc, 0x6d, 0x1f, 0x8e, 0xfb, 0x6a, 0x18, 0x89,
   0xf2, 0x63, 0x11, 0x80, 0xf5, 0x64, 0x16, 0x87,
   0xd8, 0x49, 0x3b, 0xaa, 0xdf, 0x4e, 0x3c, 0xad,
   0xd6, 0x47, 0x35, 0xa4, 0xd1, 0x40, 0x32, 0xa3,
   0xc4, 0x55, 0x27, 0xb6, 0xc3, 0x52, 0x20, 0xb1,
   0xca, 0x5b, 0x29, 0xb8, 0xcd, 0x5c, 0x2e, 0xbf,
   0x90, 0x01, 0x73, 0xe2, 0x97, 0x06, 0x74, 0xe5,
   0x9e, 0x0f, 0x7d, 0xec, 0x99, 0x08, 0x7a, 0xeb,
   0x8c, 0x1d, 0x6f, 0xfe, 0x8b, 0x1a, 0x68, 0xf9,
   0x82, 0x13, 0x61, 0xf0, 0x85, 0x14, 0x66, 0xf7,
   0xa8, 0x39, 0x4b, 0xda, 0xaf, 0x3e, 0x4c, 0xdd,
   0xa6, 0x37, 0x45, 0xd4, 0xa1, 0x30, 0x42, 0xd3,
   0xb4, 0x25, 0x57, 0xc6, 0xb3, 0x22, 0x50, 0xc1,
   0xba, 0x2b, 0x59, 0xc8, 0xbd, 0x2c, 0x5e, 0xcf
};


/*
-----------------------------------------------------------------------------
-- Cyclic Redundancy Code (CRC) for Remote Memory Access Protocol (RMAP)
-----------------------------------------------------------------------------
-- Purpose:
--    Given an intermediate SpaceWire RMAP CRC byte value and an RMAP Header
--    or Data byte, return an updated RMAP CRC byte value.
--
```

```
-- Parameters:
--    INCRC  - The intermediate RMAP CRC byte value.
--    INBYTE - The RMAP Header or Data byte.
--
-- Return value:
--    OUTCRC - The updated RMAP CRC byte value.
--
-- Description:
--    Table look-up version: uses the XOR of the intermediate CRC byte with the
--    header/data byte to obtain the updated CRC byte from a look-up table.
--
--    Generator polynomial: g(x) = x**8 + x**2 + x**1 + x**0
--
-- Notes:
--    The INCRC input CRC value must have all bits zero for the first INBYTE.
--
--    The first INBYTE must be the first Header or Data byte covered by the
--    RMAP CRC calculation. The remaining bytes must be supplied in the RMAP
--    transmission/reception byte order.
--
--    If the last INBYTE is the last Header or Data byte covered by the RMAP
--    CRC calculation then the OUTCRC output will be the RMAP CRC byte to be
--    used for transmission or to be checked against the received CRC byte.
--
--    If the last INBYTE is the Header or Data CRC byte then the OUTCRC
--    output will be zero if no errors have been detected and non-zero if
--    an error has been detected.
--
--    Each byte is inserted in or extracted from a SpaceWire packet without
--    the need for any bit reversal or similar manipulation. The SpaceWire
--    packet transmission and reception procedure does the necessary bit
--    ordering when sending and receiving Data Characters (see ECSS-E-50-12A
--    clause 7.2).
-----------------------------------------------------------------------------
 */
unsigned char RMAP_CalculateCRC(unsigned char INCRC, unsigned char INBYTE)
   { return RMAP_CRCTable[INCRC ^ INBYTE]; }
```

6.10.3. RMAP CRC test patterns

The following test patterns are based on complete SpaceWire RMAP commands and replies. The data and CRC values are read from top to bottom and are represented as bytes in hexadecimal notation.

Each byte is inserted in a SpaceWire packet without the need for any bit reversal or similar manipulation. The SpaceWire packet transmission and reception procedure does the necessary bit ordering when sending and receiving Data Characters (see ECSS-E-50-12A clause 7.2).

Prerequisites:

Writeable and readable memory at location 0xA0000000 through 0xA0000020.

Key:	0x00
Target Logical Address:	0xFE
Initiator Logical Address:	0x67
Target SpaceWire Address:	0x11 0x22 0x33 0x44 0x55 0x66 0x77, or 0x11 0x22 0x33 0x44
Initiator SpaceWire Address:	0x99 0xAA 0xBB 0xCC 0xDD 0xEE 0x00, or 0x99 0xAA 0xBB 0xCC

Note that when the number of bytes in an Initiator SpaceWire Address is not divisible by four, it requires that the Reply SpaceWire Address field in the corresponding command is padded with one or more leading bytes with value 0x00.

-- RMAP non-verified incrementing write-with-reply command - without SpaceWire addresses:

Target Logical Address:	0xFE
Protocol Identifier:	0x01
Instruction:	0x6C
Key:	0x00
Initiator Logical Address:	0x67
Transaction Identifier MS:	0x00
Transaction Identifier LS:	0x00
Extended Address:	0x00
Address MS:	0xA0
Address:	0x00
Address:	0x00
Address LS:	0x00
Data Length MS:	0x00
Data Length:	0x00
Data Length LS:	0x10
Header CRC:	0x9F
Data:	0x01
Data:	0x23
Data:	0x45
Data:	0x67
Data:	0x89
Data:	0xAB
Data:	0xCD
Data:	0xEF
Data:	0x10
Data:	0x11
Data:	0x12
Data:	0x13

Data:	0x14
Data:	0x15
Data:	0x16
Data:	0x17
Data CRC:	0x56

-- Expected RMAP successful write reply to previous command - without SpaceWire addresses:

Initiator Logical Address:	0x67
Protocol Identifier:	0x01
Instruction:	0x2C
Status:	0x00
Target Logical Address:	0xFE
Transaction Identifier MS:	0x00
Transaction Identifier MS:	0x00
Header CRC:	0xED

-- RMAP incrementing read command - without SpaceWire addresses:

Target Logical Address:	0xFE
Protocol Identifier:	0x01
Instruction:	0x4C
Key:	0x00
Initiator Logical Address:	0x67
Transaction Identifier MS:	0x00
Transaction Identifier LS:	0x01
Extended Address:	0x00
Address MS:	0xA0
Address:	0x00
Address:	0x00
Address LS:	0x00
Data Length MS:	0x00
Data Length:	0x00
Data Length LS:	0x10
Header CRC:	0xC9

-- Expected RMAP successful read reply to previous command - without SpaceWire addresses:

Initiator Logical Address:	0x67
Protocol Identifier:	0x01
Instruction:	0x0C
Status:	0x00

Target Logical Address:	0xFE
Transaction Identifier MS:	0x00
Transaction Identifier MS:	0x01
Reserved:	0x00
Data Length MS:	0x00
Data Length:	0x00
Data Length LS:	0x10
Header CRC:	0x6D
Data:	0x01
Data:	0x23
Data:	0x45
Data:	0x67
Data:	0x89
Data:	0xAB
Data:	0xCD
Data:	0xEF
Data:	0x10
Data:	0x11
Data:	0x12
Data:	0x13
Data:	0x14
Data:	0x15
Data:	0x16
Data:	0x17
Data CRC:	0x56

-- RMAP non-verified incrementing write-with-reply command - with SpaceWire addresses:

Target SpaceWire Address:	0x11 (not part of Header CRC)
Target SpaceWire Address:	0x22 (not part of Header CRC)
Target SpaceWire Address:	0x33 (not part of Header CRC)
Target SpaceWire Address:	0x44 (not part of Header CRC)
Target SpaceWire Address:	0x55 (not part of Header CRC)
Target SpaceWire Address:	0x66 (not part of Header CRC)
Target SpaceWire Address:	0x77 (not part of Header CRC)
Target Logical Address:	0xFE
Protocol Identifier:	0x01
Instruction:	0x6E
Key:	0x00
Reply SpaceWire Address:	0x00

Reply SpaceWire Address:	0x99
Reply SpaceWire Address:	0xAA
Reply SpaceWire Address:	0xBB
Reply SpaceWire Address:	0xCC
Reply SpaceWire Address:	0xDD
Reply SpaceWire Address:	0xEE
Reply SpaceWire Address:	0x00
Initiator Logical Address:	0x67
Transaction Identifier MS:	0x00
Transaction Identifier LS:	0x02
Extended Address:	0x00
Address MS:	0xA0
Address:	0x00
Address:	0x00
Address LS:	0x10
Data Length MS:	0x00
Data Length:	0x00
Data Length LS:	0x10
Header CRC:	0x7F
Data:	0xA0
Data:	0xA1
Data:	0xA2
Data:	0xA3
Data:	0xA4
Data:	0xA5
Data:	0xA6
Data:	0xA7
Data:	0xA8
Data:	0xA9
Data:	0xAA
Data:	0xAB
Data:	0xAC
Data:	0xAD
Data:	0xAE
Data:	0xAF
Data CRC:	0xB4

-- Expected RMAP successful write reply to the previous command - with SpaceWire addresses:

Reply SpaceWire Address:	0x99 (not part of Header CRC)
Reply SpaceWire Address:	0xAA (not part of Header CRC)

```
Reply SpaceWire Address:  0xBB (not part of Header CRC)
Reply SpaceWire Address:  0xCC (not part of Header CRC)
Reply SpaceWire Address:  0xDD (not part of Header CRC)
Reply SpaceWire Address:  0xEE (not part of Header CRC)
Reply SpaceWire Address:  0x00 (not part of Header CRC)
Initiator Logical Address:   0x67
Protocol Identifier:         0x01
Instruction:                 0x2E
Status:                      0x00
Target Logical Address:      0xFE
Transaction Identifier MS:   0x00
Transaction Identifier MS:   0x02
Header CRC:                   0x1D


-------------------------------------------------------------------------------
-- RMAP incrementing read command - with SpaceWire addresses:
Target SpaceWire Address: 0x11 (not part of Header CRC)
Target SpaceWire Address: 0x22 (not part of Header CRC)
Target SpaceWire Address: 0x33 (not part of Header CRC)
Target SpaceWire Address: 0x44 (not part of Header CRC)
Target Logical Address:      0xFE
Protocol Identifier:         0x01
Instruction:                 0x4D
Key:                         0x00
Reply SpaceWire Address:     0x99
Reply SpaceWire Address:     0xAA
Reply SpaceWire Address:     0xBB
Reply SpaceWire Address:     0xCC
Initiator Logical Address:   0x67
Transaction Identifier MS:   0x00
Transaction Identifier LS:   0x03
Extended Address:            0x00
Address MS:                  0xA0
Address:                     0x00
Address:                     0x00
Address LS:                  0x10
Data Length MS:              0x00
Data Length:                 0x00
Data Length LS:              0x10
Header CRC:                  0xF7
```

-- Expected RMAP successful read reply to the previous command - with SpaceWire addresses:

Reply SpaceWire Address:	0x99 (not part of Header CRC)
Reply SpaceWire Address:	0xAA (not part of Header CRC)
Reply SpaceWire Address:	0xBB (not part of Header CRC)
Reply SpaceWire Address:	0xCC (not part of Header CRC)
Initiator Logical Address:	0x67
Protocol Identifier:	0x01
Instruction:	0x0D
Status:	0x00
Target Logical Address:	0xFE
Transaction Identifier MS:	0x00
Transaction Identifier MS:	0x03
Reserved:	0x00
Data Length MS:	0x00
Data Length:	0x00
Data Length LS:	0x10
Header CRC:	0x52
Data:	0xA0
Data:	0xA1
Data:	0xA2
Data:	0xA3
Data:	0xA4
Data:	0xA5
Data:	0xA6
Data:	0xA7
Data:	0xA8
Data:	0xA9
Data:	0xAA
Data:	0xAB
Data:	0xAC
Data:	0xAD
Data:	0xAE
Data:	0xAF
Data CRC:	0xB4

6.11. Annex Example Service Interface Specification (informative)

Example service interface specifications for RMAP are provided in this annex.

The managed parameters are defined in sub-clause 6.9.

6.11.1. Write Service

6.11.1.1. Initiator

The service primitives associated with this service are:

a) WRITE.request;

b) WRITE.confirmation.

6.11.1.2. WRITE.request

6.11.1.2.1. Function

The RMAP Initiator Write service user passes a WRITE.request primitive to the service provider to request that data is written to memory in a target across the SpaceWire network.

6.11.1.2.2. Semantics

The WRITE.request primitive provides parameters as follows:

WRITE.request (Target SpaceWire Address, Target Logical Address, Write Command Options, Key, Reply Address, Initiator Logical Address, Transaction Identifier, Extended Address, Memory Address, Data Length, Data)

6.11.1.2.3. When Generated

The WRITE.request primitive is passed to the RMAP Initiator Write service provider to request it to write the data into memory in the target.

6.11.1.2.4. Effect On Receipt

Receipt of the WRITE.request primitive causes the RMAP Initiator Write service provider to send an RMAP write command to the target.

6.11.1.3. WRITE.confirmation

6.11.1.3.1. Function

The RMAP Initiator Write service provider passes a WRITE.confirmation primitive to the RMAP Initiator Write Service user to confirm that data has been written to memory in a target across the SpaceWire network or to report that an error occurred.

6.11.1.3.2. Semantics

The WRITE.confirmation primitive provides parameters as follows:

WRITE.confirmation (Transaction Identifier, Status)

6.11.1.3.3. When Generated

The WRITE.confirmation primitive is passed to the RMAP Initiator Write Service user in the initiator when a write reply is received.

6.11.1.3.4. Effect On Receipt

The effect on receipt of the WRITE.confirmation primitive on the RMAP Initiator Write Service user in the initiator is undefined.

6.11.1.3.5. Additional Comments

The transaction identifier parameter is used by the RMAP Initiator Write service user to identify which RMAP transaction is being confirmed.

6.11.1.4. Target

The service primitives associated with this service are:

a) WRITE.authorisation.indication;

b) WRITE.authorisation.response;

c) WRITE.data.indication;

d) WRITE.data.response;

e) WRITE.indication.

6.11.1.5. WRITE.authorisation.indication

6.11.1.5.1. Function

The RMAP Target Write service provider passes a WRITE.authorisation.indication to the RMAP Target Write service user to ask for authorisation to write to memory in the target.

6.11.1.5.2. Semantics

The WRITE.authorisation.indication primitive provides parameters as follows:

WRITE.authorisation.indication (Target Logical Address, Instruction, Key, Initiator Logical Address, Transaction Identifier, Extended Address, Memory Address, Data Length)

6.11.1.5.3. When Generated

The WRITE.authorisation.indication primitive is passed from the RMAP Target Write service provider to the RMAP Target Write Service user to request permission to write to memory in the target.

6.11.1.5.4. Effect On Receipt

The effect of receipt of the WRITE.authorisation.indication primitive on the RMAP Target Write Service user is for it to issue a WRITE.authorisation.response primitive either authorising or not authorising the memory write operation.

6.11.1.6. WRITE.authorisation.response

6.11.1.6.1. Function

The RMAP Target Write service user passes a WRITE.authorisation.response to the RMAP Target Write service provider to give permission or deny permission to write to memory in the target.

6.11.1.6.2. Semantics

The WRITE.authorisation.response primitive provides parameters as follows:

WRITE.authorisation.response (Authorise)

6.11.1.6.3. When Generated

The WRITE.authorisation.response primitive is passed from the RMAP Target Write service user to the RMAP Target Write service provider at the target in response to a WRITE.authorisation.indication primitive.

6.11.1.6.4. Effect On Receipt

The effect of receipt of the WRITE.authorisation.response primitive on the RMAP Target Write service provider is for it to write data to memory if authorisation is given.

6.11.1.7. WRITE.data.indication

6.11.1.7.1. Function

The RMAP Target Write service provider passes a WRITE.data.indication to the RMAP Write service user to write data to memory in the target.

6.11.1.7.2. Semantics

The WRITE.data.indication primitive provides parameters as follows:

WRITE.data.indication (Extended Address, Address, Data Length, Incrementing/Non-incrementing, Data)

6.11.1.7.3. When Generated

The WRITE.data.indication primitive is passed from the RMAP Target Write service provider to the RMAP Target Write Service user when permission to write data has been given by the WRITE.authorisation.response primitive.

6.11.1.7.4. Effect On Receipt

The effect of receipt of the WRITE.data.indication primitive on the RMAP Target Write service user is for data to be written into memory in the target.

6.11.1.8. WRITE.data.response

6.11.1.8.1. Function

The RMAP Target Write service user passes a WRITE.data.response to the RMAP Write service provided when data has been written to memory in the target.

6.11.1.8.2. Semantics

The WRITE.data.response primitive provides parameters as follows:

WRITE.data.response (Status)

6.11.1.8.3. When Generated

The WRITE.data.response primitive is passed from the RMAP Target Write service user to the RMAP Target Write service provider when data has been successfully written to target memory or a failure has occurred while writing data to target memory by the WRITE.data.indication primitive.

6.11.1.8.4. Effect On Receipt

The effect of receipt of the WRITE.data.response primitive on the RMAP Target Write service provider is for a reply to be sent to the initiator (or other node) if requested and for a WRITE.indication to be passed to the RMAP Target Write service user.

6.11.1.9. WRITE.indication

6.11.1.9.1. Function

The RMAP Target Write service provider passes a WRITE.indication to the RMAP Write service user to indicate that data has been successfully written to memory in the target.

6.11.1.9.2. Semantics

The WRITE.indication primitive does not have any parameters.

6.11.1.9.3. When Generated

The WRITE.indication primitive is produced when a WRITE.data.response is received from the RMAP Target Write service user with its status parameter indicating that no fault has occurred during the write operation.

6.11.1.9.4. Effect On Receipt

The effect of receipt of the WRITE.indication primitive on the RMAP Target Write service user is undefined.

6.11.2. Read Service

6.11.2.1. Initiator

The service primitives associated with this service are:

a) READ.request;

b) READ.confirmation.

6.11.2.2. READ.request

6.11.2.2.1. Function

The RMAP Initiator Read service user passes a READ.request primitive to the RMAP Initiator Read service provider to request that data is read from memory in a target across the SpaceWire network.

6.11.2.2.2. Semantics

The READ.request primitive provides parameters as follows:

READ.request (Target SpaceWire Address, Target Logical Address, Read Command Options, Key, Reply Address, Initiator Logical Address, Transaction Identifier, Extended Address, Memory Address, Data Length)

6.11.2.2.3. When Generated

The READ.request primitive is passed to the RMAP Initiator Read service provider to request it to read data from memory in the target.

6.11.2.2.4. Effect On Receipt

Receipt of the READ.request primitive causes the RMAP Initiator Read service provider to send an RMAP read command to the target.

6.11.2.3. READ.confirmation

6.11.2.3.1. Function

The RMAP Initiator Read service provider passes a READ.confirmation primitive to the RMAP Initiator Read service user to confirm that data has been read from memory in a target across the SpaceWire network and to provide that data to the RMAP Initiator Read service user, or to report that an error occurred.

6.11.2.3.2. Semantics

The READ.confirmation primitive provides parameters as follows:

READ.confirmation (Transaction Identifier, Status, Data Length, Data)

6.11.2.3.3. When Generated

The READ.confirmation primitive is passed to the RMAP Initiator Read service user when a read reply is received.

6.11.2.3.4. Effect On Receipt

The effect on receipt of the READ.confirmation primitive by the RMAP Initiator Read service user is undefined.

6.11.2.3.5. Additional Comments

The transaction identifier parameter is used by the RMAP Initiator Read service user to identify which RMAP transaction is being confirmed.

6.11.2.4. Target

The service primitives associated with this service are:

a) READ.authorisation.indication;

b) READ.authorisation.response;

c) READ.data.indication;

d) READ.data.response;

e) READ.indication;

6.11.2.5. READ.authorisation.indication

6.11.2.5.1. Function

The RMAP Target Read service provider passes a READ.authorisation.indication to the RMAP Target Read service user to ask for authorisation to read data from memory in the target.

6.11.2.5.2. Semantics

The READ.authorisation.indication primitive provides parameters as follows:

READ.authorisation indication (Target Logical Address, Instruction, Key, Initiator Logical Address, Transaction Identifier, Extended Address, Memory Address, Data Length)

6.11.2.5.3. When Generated

The READ.authorisation.indication primitive is passed from the RMAP Target Read service provider to the RMAP Target Read service user at the target to request permission to read data from memory in the target.

6.11.2.5.4. Effect On Receipt

The effect of receipt of the READ.authorisation.indication primitive on the RMAP Target Read service user is for it to issue a READ.authorisation.response primitive either accepting or rejecting the read operation.

6.11.2.6. READ.authorisation.response

6.11.2.6.1. Function

The RMAP Target Read service user passes a READ.authorisation.response to the RMAP Target Read service provider to give permission or deny permission to read from memory in the target.

6.11.2.6.2. Semantics

The READ.authorisation.response primitive provides parameters as follows:

READ.authorisation.response (Authorise)

6.11.2.6.3. When Generated

The READ.authorisation.response primitive is passed from the RMAP Target Read service user to the RMAP Target Read service provider at the target in response to a READ.authorisation.indication primitive.

6.11.2.6.4. Effect On Receipt

The effect of receipt of the READ.authorisation.response primitive on the RMAP Target Read service provider is for it to read data from memory by issuing a READ.data.indication primitive.

6.11.2.7. READ.data.indication

6.11.2.7.1. Function

The RMAP Target Read service provider passes a READ.data.indication to the RMAP Target Read service user to read data from memory in the target.

6.11.2.7.2. Semantics

The READ.data.indication primitive provides parameters as follows:

READ.data.indication (Extended Address, Address, Data Length, Incrementing/Non-incrementing)

6.11.2.7.3. When Generated

The READ.data.indication primitive is passed from the RMAP Target Read service provider to the RMAP Target Read service user at the target when permission to read data has been given by the READ.authorisation.response primitive.

6.11.2.7.4. Effect On Receipt

The effect of receipt of the READ.data.indication primitive on the RMAP Target Read service user is for data to be read from memory in the target.

6.11.2.8. READ.data.response

6.11.2.8.1. Function

The RMAP Target Read service provider passes a READ.data.response to the RMAP Target Read service user to provide data read from memory in the target.

6.11.2.8.2. Semantics

The READ.data.response primitive provides parameters as follows:

READ.data.response (Status, Data Length, Data)

6.11.2.8.3. When Generated

The READ.data.response primitive is passed from the RMAP Target Read service user to the RMAP Target Read service provider after a READ.data.indication has been received.

6.11.2.8.4. Effect On Receipt

The effect of receipt of the READ.data.response primitive on the RMAP Target Read service provider is for the data read from memory in the target or an error to be returned to the initiator (or other node).

6.11.2.9. READ.indication

6.11.2.9.1. Function

The RMAP Target Read service provider passes a READ.indication to the RMAP Target Read service user to indicate that data has been successfully read from memory in the target.

6.11.2.9.2. Semantics

The READ.indication primitive provides parameters as follows:

READ.indication

6.11.2.9.3. When Generated

The READ.indication primitive is passed from the RMAP Target Read service provider to the RMAP Target Read service user at the target when data has been successfully read from memory in the target.

6.11.2.9.4. Effect On Receipt

The effect of receipt of the READ.indication primitive on the RMAP Target Read service user is undefined.

6.11.3. Read-Modify-Write Service

6.11.3.1. Initiator

The service primitives associated with this service are:

a) RMW.request;

b) RMW.confirmation.

6.11.3.2. RMW.request

6.11.3.2.1. Function

At the initiator, the RMAP read-modify-write service user passes a RMW.request primitive to the service provider to request that data is read-modify-write memory in a target across the SpaceWire network.

6.11.3.2.2. Semantics

The RMW.request primitive provides parameters as follows:

RMW.request (Target SpaceWire Address, Target Logical Address, RMW Command Options, Key, Reply Address, Initiator Logical Address, Transaction Identifier, Extended Address, Memory Address, Data Length, Data, Mask)

6.11.3.2.3. When Generated

The RMW.request primitive is passed to the service provider to request it to read-modify-write memory in the target.

6.11.3.2.4. Effect On Receipt

Receipt of the RMW.request primitive causes the service provider to send an RMAP read-modify-write command.

6.11.3.3. RMW.confirmation

6.11.3.3.1. Function

At the initiator, the service provider passes a RMW.confirmation primitive to the RMAP read-modify-write service user to confirm that data has been read-modify-written to memory in the target across the SpaceWire network.

6.11.3.3.2. Semantics

The RMW.confirmation primitive provides parameters as follows:

RMW.confirmation (Transaction Identifier, Status, Data)

6.11.3.3.3. When Generated

The RMW.confirmation primitive is passed to the RMAP read-modify-write service user in the initiator when a RMW reply is received to confirm that data has been read-modify-written to the memory in the target and to provide the data read to the initiator.

6.11.3.3.4. Effect On Receipt

Receipt of the RMW.confirmation primitive by the RMAP read-modify-write service user in the initiator is undefined.

6.11.3.3.5. Additional Comments

The transaction identifier parameter is used in the initiator to identify which RMAP transaction is being confirmed.

6.11.3.4. Target

a) RMW.authorisation.indication;

b) RMW.authorisation.response;

c) RMW.read.data.indication;

d) RMW.read data.response;

e) RMW.write.data.indication;

f) RMW.write.data.response;

g) RMW.indication;

.

6.11.3.5. RMW.authorisation.indication

6.11.3.5.1. Function

The RMAP Target RMW service provider passes a RMW.authorisation.indication to the RMAP Target RMW service user to ask for authorisation to read-modify-write memory in the target.

6.11.3.5.2. Semantics

The RMW.authorisation.indication primitive provides parameters as follows:

RMW.authorisation.indication (Target Logical Address, Instruction, Key, Initiator Logical Address, Transaction Identifier, Extended Address, Memory Address, Data Length)

6.11.3.5.3. When Generated

The RMW.authorisation.indication primitive is passed from the RMAP Target RMW service provider to the RMAP Target RMW service user at the target to request permission to read-modify-write memory in the target.

6.11.3.5.4. Effect On Receipt

The effect of receipt of the RMW.authorisation.indication primitive by the RMAP Target RMW service user is for it to issue a RMW.authorisation.response primitive either accepting or rejecting the RMW operation.

6.11.3.6. RMW.authorisation.response

6.11.3.6.1. Function

The RMAP Target RMW service user passes a RMW.authorisation.response to the RMAP Target RMW service provider to give permission or deny permission to read-modify-write memory in the target.

6.11.3.6.2. Semantics

The RMW.authorisation.response primitive provides parameters as follows:

RMW.authorisation.response (Authorise)

6.11.3.6.3. When Generated

The RMW.authorisation.response primitive is passed from the RMAP Target RMW service user to the RMAP Target RMW service provider at the target in response to a READ.authorisation primitive.

6.11.3.6.4. Effect On Receipt

The effect of receipt of the RMW.authorisation.response primitive on the RMAP Target RMW service provider is for it to read memory in the target by issuing a RMW.data.indication primitive.

6.11.3.7. RMW.read.data.indication

6.11.3.7.1. Function

The RMAP Target RMW service provider passes a RMW.read.data.indication to the RMAP Target RMW service user to read data from memory in the target.

6.11.3.7.2. Semantics

The RMW.read.data.indication primitive provides parameters as follows:

RMW.read.data.indication (Extended Address, Address, Data Length, Incrementing/Non-incrementing)

6.11.3.7.3. When Generated

The RMW.read.data.indication primitive is passed from the RMAP Target RMW service provider to the RMAP Target RMW service user at the target when permission to read data has been given by the RMW.authorisation.response primitive.

6.11.3.7.4. Effect On Receipt

The effect of receipt of the RMW.read.data.indication primitive on the RMAP Target RMW service user is for data to be read from memory in the target and returned to the service provider in a RMW.read.data.response.

6.11.3.8. RMW.read.data.response

6.11.3.8.1. Function

The RMAP Target RMW service provider passes a RMW.read.data.response to the RMAP Target RMW service user to provide data read from memory in the target.

6.11.3.8.2. Semantics

The READ.data.response primitive provides parameters as follows:

READ.data.response (Status, Data Length, Data)

6.11.3.8.3. When Generated

The RMW.read.data.response primitive is passed from the RMAP Target RMW service user to the RMAP Target RMW service provider after a RMW.read.data.indication has been received.

6.11.3.8.4. Effect On Receipt

The effect of receipt of the RMW.read.data.response primitive on the RMAP Target RMW service provider is for the data and mask from the RMW command to be passed to the RMAP RMW service user for combining with the data read from memory and writing back into memory.

6.11.3.9. RMW.write.data.indication

6.11.3.9.1. Function

The RMAP Target RMW service provider passes a RMW.write.data.indication to the RMAP RMW service user to modify and write data to memory in the target.

6.11.3.9.2. Semantics

The RMW.write.data.indication primitive provides parameters as follows:

RMW.write.data.indication (Extended Address, Address, Data Length, Incrementing/Non-incrementing, Data, Mask)

6.11.3.9.3. When Generated

The RMWwrite.data.indication primitive is passed from the RMAP Target RMW service provider to the RMAP Target RMW Service user when data has been read from memory and returned by the RMW.read.data.response primitive.

6.11.3.9.4. Effect On Receipt

The effect of receipt of the RMW.write.data.indication primitive on the RMAP Target Write service user is for the data and mask to be combined in some way with the data previously in memory and the new value written into the same memory location in the target.

6.11.3.10. RMW.write.data.response

6.11.3.10.1. Function

The RMAP Target RMW service user passes a RMW.write.data.response to the RMAP RMW service provided when data and mask have been combined with the data previously in memory and the new result written to memory in the target.

6.11.3.10.2. Semantics

The RMW.write.data.response primitive provides parameters as follows:

RMW.write.data.response (Status)

6.11.3.10.3. When Generated

The RMW.write.data.response primitive is passed from the RMAP Target RMW service user to the RMAP Target RMW service provider when the data and mask have been successfully combined with the data previously in memory and the new result written to target memory or a failure has occurred while combining or writing data to target memory by the RMW.write.data.indication primitive.

6.11.3.10.4. Effect On Receipt

The effect of receipt of the RMW.write.data.response primitive on the RMAP Target RMW service provider is for a reply to be sent to the initiator (or other node) if requested and for a RMW.indication to be passed to the RMAP Target RMW service user.

6.11.3.11. RMW.indication

6.11.3.11.1. Function

At the target, the service provider passes a RMW.indication to the RMAP RMW service user after read-modify-writing memory in the target.

6.11.3.11.2. Semantics

The RMW.indication primitive does not have any parameters.

6.11.3.11.3. When Generated

The RMW.indication primitive is produced when a RMW.write.data.response is received from the RMAP Target RMW service user with its status parameter indicating that no fault has occurred during the read-modify-write operation.

6.11.3.11.4. Effect On Receipt

The effect of receipt of the RMW.indication primitive on the RMAP Target RMW service user is undefined.

7.This heading is here because of problems with clause not being a heading

7. CCSDS Packet Encapsulation Protocol

7.1. Overview

7.1.1. Purpose

The CCSDS Packet Encapsulation Protocol has been designed to encapsulate a CCSDS Packet into a SpaceWire packet, transfer it from an initiator to a target across a SpaceWire network, extract it from the SpaceWire packet and pass it to a target user application. This protocol does not provide any means for ensuring delivery of the packet. An optional mechanism to route the packets to different communication channels at the target user is provided.

The CCSDS Packet Protocol is defined in the following documents:

- CCSDS 133.0-B-1 Blue Book, CCSDS Packet Protocol,
- CCSDS 713.0-B-1 Blue Book, Space Communications Protocol Specification - Network Protocol (SCPS-NP),
- CCSDS 133.1-B-1 Blue Book, Encapsulation Service,
- Internet Protocol STD 5, September 1981 [RFC 791, RFC950, RFC 919, RFC 922, RFC 792, RFC 1112],
- CCSDS 135-B-1 Blue Book, Space Link Identifiers.

7.1.2. Guide to clause 7

Specification of the fields used in the CCSDS Packet Encapsulation Protocol packets is given in sub-clause 7.2. The format of these packets is then given in sub-clause 7.3. The operation of the CCSDS Packet Encapsulation Protocol is described in sub-clause 7.4.

7.2. CCSDS Packet Encapsulation Protocol fields

7.2.1. Target SpaceWire Address field

a. The Target SpaceWire Address field shall comprise zero or more data characters forming the SpaceWire address which is used to route the CCSDS Packet Encapsulation Protocol packet to the target.

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

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

c. The Target SpaceWire Address field shall not be used when a single logical address is being used for routing the CCSDS Packet Encapsulation Protocol packet to the target.

NOTE In this case the CCSDS Packet Encapsulation Protocol packet is routed to the target by the Target Logical Address contained in the Target Logical Address field.

7.2.2. Target Logical Address field

Target Logical Address field shall be an 8-bit field that contains a logical address of the target.

NOTE The Target Logical Address field is normally set to a logical address recognised by the target.

NOTE If the target does not have a specific logical address then the Target Logical Address field can be set to the default value 254 (0xFE).

NOTE A target can have more than one logical address.

7.2.3. 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 2 (0x02) which is the Protocol Identifier for the CCSDS Packet Encapsulation Protocol.

7.2.4. User Application 1 field

a. The User Application field 1 shall be an 8-bit field that is set to 0x00.

b. The User Application 1 field may be set according to one of the following options:

1. Option A Virtual Channels:

(a) The User Application 1 field may be used as a transaction identifier (i.e. set to any value) that is used to identify a particular packet.

7.2.5. User Application 2 field

a. The User Application 2 field shall be an 8-bit field that is set to 0x00.

b. The User Application 2 field may be set according to one of the following options:

1. Option A Virtual Channels:

(a) If the target supports virtual channels, the User Application 1field shall be an 8-bit field that is set to a virtual channel number that is supported by the target.

7.2.6. CCSDS Packet field

a. The CCSDS Packet field shall be a variable length field that contains the CCSDS Packet.

b. The first byte of the CCSDS Packet field shall be the first byte of the CCSDS Packet.

c. The byte order of the CCSDS Packet field shall be the same as the CCSDS Packet.

7.3. CCSDS Packet Encapsulation Protocol format

7.3.1.1. Fields

The CCSDS Packet Encapsulation Protocol packet shall contain the fields shown in Figure 7-1.

First byte transmitted

	Target SpW Address	....	Target SpW Address
Target Logical Address	Protocol Identifier	User Application 1	User Application 2
CCSDS Packet (First Byte)	CCSDS Packet	CCSDS Packet	CCSDS Packet
CCSDS Packet	...	...	CCSDS Packet
CCSDS Packet	CCSDS Packet (Last Byte)	EOP	

Last byte transmitted

Figure 7-1 Encapsulated CCSDS Packet Format

7.3.1.2. Target SpaceWire Address field

The Target SpaceWire Address field shall be as defined in sub-clause 7.2.1.

7.3.1.3. Target Logical Address field

The Target Logical Address field shall be as defined in sub-clause 7.2.2.

7.3.1.4. Protocol Identifier field

The Protocol Identifier field shall be as defined in sub-clause 7.2.3.

7.3.1.5. User Application 1 field

The User Application 1 field format shall be as defined in sub-clause 7.2.4.

7.3.1.6. User Application 2 field

The User Application 2 field format shall be as defined in sub-clause 7.2.5.

7.3.1.7. CCSDS Packet field

The CCSDS Packet field format shall be as defined in sub-clause 7.2.6.

7.3.1.8. EOP character

The end of the CCSDS Packet Encapsulation Protocol packet shall be indicated by an EOP character.

7.4. CCSDS Packet Encapsulation Protocol Action

7.4.1. Overview

The normal sequence of actions for a CCSDS Packet Encapsulation Protocol packet transfer is illustrated in Figure 7-2.

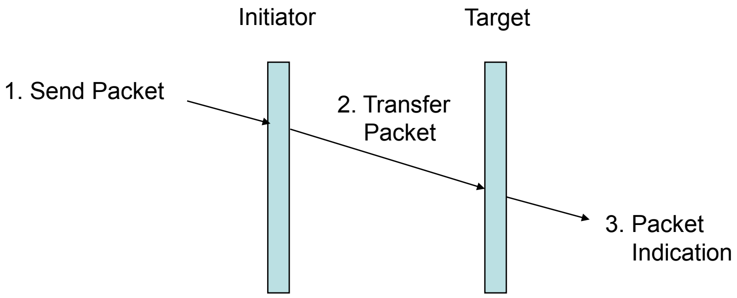


Figure 7-2 CCSDS Packet Encapsulation Protocol Packet Transfer

7.4.2. Send request

a. The CCSDS Packet Encapsulation Protocol packet transfer shall begin when an initiator user application requests to send a CCSDS Packet Encapsulation Protocol packet (Send Request).

b. The initiator user application shall pass the following information to the initiator:

1. Target SpaceWire Address
2. Target Logical Address
3. CCSDS Packet

7.4.3. Transfer packet

a. In response to the send request the initiator shall encapsulate the CCSSDS Packet into a SpaceWire packet as described in sub-clause 7.3 and send it across the SpaceWire network to the target (Transfer Packet).

NOTE The Target SpaceWire Address and Target Logical Address are used to route the command packet to the target.

7.4.4. Packet indication

a. When a SpaceWire packet is received at the target and the Protocol Identifier field is 0x02 the packet shall be regarded as a CCSDS Packet Encapsulation Protocol packet.

b. If the CCSDS Packet Encapsulation Protocol packet arrives at the target containing more than four bytes and is terminated by an EOP, the CCSDS Packet shall be extracted from the SpaceWire packet and passed to the target user application.

c. If the CCSDS Packet Encapsulation Protocol packet arrives at the target containing less than five bytes and is terminated by an EOP, the target user application should be informed that an Encapsulated CCSDS Packet has arrived which is too short, i.e. it does not contain a CCSDS Packet.

d. If the CCSDS Packet Encapsulation Protocol packet arrives at the target terminated by an EEP, the available CCSDS Packet information (if any) shall be extracted from the CCSDS Packet Encapsulation Protocol packet and passed to the target user application and the application informed that the CCSDS Packet Encapsulation Protocol packet was terminated by an EEP.

e. The User Application fields shall be ignored by the target unless one of the following options is implemented:

1. Option A Virtual Channels:

(a) If the target supports virtual channels it may use the User Application 1 field to identify the specific packet.

(b) If the target supports virtual channels, the User Application 2 field shall be used to route the Encapsulated CCSDS Packet to the target user application.

7.5. Annex Example Service Interface Specification for CCSDS Packet Encapsulation Protocol

Example service interface specifications for the CCSDS Packet Encapsulation Protocol are provided in this section.

7.5.1. CCSDS Packet Transfer Service

The service primitives associated with this service are:

a) CCSDS_PACKET.send;

b) CCSDS_PACKET.indication.

7.5.2. CCSDS_PACKET.send

7.5.2.1. Function

At the initiator, the CCSDS Packet Transfer service user passes a CCSDS_PACKET.send primitive to the service provider to request that a CCSDS Packet be transferred to the user at the target across the SpaceWire network.

7.5.2.2. Semantics

The CCSDS_PACKET.send primitive provides parameters as follows:

CCSDS_PACKET.send (CCSDS Packet, Target SpaceWire Address, Target Logical Address)

7.5.2.3. When Generated

The CCSDS_PACKET.send primitive is passed to the service provider to request it to send the CCSDS Packet.

7.5.2.4. Effect On Receipt

Receipt of the CCSDS_PACKET.send primitive causes the service provider to transfer the CCSDS Packet.

7.5.2.5. Additional Comments

The CCSDS_PACKET.request primitive is used to transfer CCSDS Packets across the SpaceWire network along the router defined by the Target SpaceWire Address and Target Logical Address parameters.

7.5.3. CCSDS_PACKET.indication

7.5.3.1. Function

At the target, the service provider passes a CCSDS_PACKET.indication to the CCSDS Packet Service user to deliver a Packet.

7.5.3.2. Semantics

The CCSDS_PACKET.indication primitive provides parameters as follows:

CCSDS_PACKET.indication (CCSDS Packet, Length, Error)

7.5.3.3. When Generated

The CCSDS_PACKET.indication primitive is passed from the service provider to the CCSDS Packet Service user at the target to deliver a CCSDS Packet.

7.5.3.4. Effect On Receipt

The effect of receipt of the CCSDS_PACKET.indication primitive by the CCSDS Packet Service user is undefined.

7.5.3.5. Additional Comments

The CCSDS_PACKET.indication primitive shall be used to deliver CCSDS Packets to the CCSDS Packet Service user at the target.