

BS EN 16603-50-52:2014



BSI Standards Publication

Space engineering — SpaceWire — Remote memory access protocol

bsi.

...making excellence a habit.
AVA Documents™
Document Sharing Hub

National foreword

This British Standard is the UK implementation of EN 16603-50-52:2014.

The UK participation in its preparation was entrusted to Technical Committee ACE/68, Space systems and operations.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2014. Published by BSI Standards Limited 2014

ISBN 978 0 580 84189 7

ICS 49.140

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 September 2014.

Amendments issued since publication

Date	Text affected
------	---------------

English version

Space engineering - SpaceWire - Remote memory access protocol

Ingénierie spatiale - SpaceWire - protocole d'accès à
distance à la mémoire

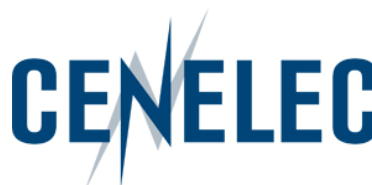
Raumfahrttechnik - SpaceWire - Protokoll zum
ferngesteuerten Speicherzugriff

This European Standard was approved by CEN on 1 March 2014.

CEN and CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration. Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CEN and CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN and CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CEN and CENELEC members are the national standards bodies and national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.



CEN-CENELEC Management Centre:
Avenue Marnix 17, B-1000 Brussels

Table of contents

Foreword	7
1 Scope.....	8
2 Normative references	9
3 Terms, definitions and abbreviated terms.....	10
3.1 Terms defined in other standards	10
3.2 Terms specific to the present standard	10
3.3 Abbreviated terms.....	10
3.4 Conventions.....	11
4 Principles	12
4.1 Remote Memory Access Protocol (RMAP) purpose.....	12
4.2 Guide to clause 5.....	12
4.3 RMAP operations.....	13
4.3.1 Introduction	13
4.3.2 Write commands	13
4.3.3 Read commands.....	14
4.3.4 Read-modify-write	14
5 Requirements.....	15
5.1 RMAP command and reply fields.....	15
5.1.1 Target SpaceWire Address field.....	15
5.1.2 Target Logical Address field	15
5.1.3 Protocol Identifier field.....	15
5.1.4 Instruction field	16
5.1.5 Key field	17
5.1.6 Reply Address field	17
5.1.7 Initiator Logical Address field	19
5.1.8 Transaction Identifier field	19
5.1.9 Extended Address field	19
5.1.10 Address field	19
5.1.11 Data Length field	20

5.1.12	Header CRC field	20
5.1.13	Data field	20
5.1.14	Mask field	20
5.1.15	Data CRC field	20
5.1.16	Reply SpaceWire Address field	20
5.1.17	Status field	20
5.2	Cyclic Redundancy Code	21
5.3	Write Command	23
5.3.1	Write command format	23
5.3.2	Write reply format	26
5.3.3	Write action	28
5.4	Read Command	39
5.4.1	Read command format	39
5.4.2	Read reply format	41
5.4.3	Read action	43
5.5	Read-Modify-Write Command	52
5.5.1	Read-modify-write command format	52
5.5.2	Read-modify-write reply format	56
5.5.3	Read-modify-write action	59
5.6	Error and status codes	70
5.6.1	Error and status codes	70
5.7	Partial Implementations of RMAP	72
5.7.1	Limited functionality nodes	72
5.7.2	Partial implementations	72
5.8	RMAP conformance	73
5.8.1	Overview	73
5.8.2	RMAP partial implementations	73
Annex A (informative)	Example of RMAP CRC implementation	77
A.1	Overview	77
A.2	VHDL implementation of RMAP CRC	77
A.3	C-code implementation of RMAP CRC	79
A.4	RMAP CRC test patterns	81
Annex B (informative)	Example Service Interface specification	88
B.1	Overview	88
B.2	Write Service	89
B.2.1	Initiator	89
B.2.2	WRITE.request	89

B.2.3	WRITE.confirmation	89
B.2.4	Target	90
B.2.5	WRITE.authorisation.indication	90
B.2.6	WRITE.authorisation.response	91
B.2.7	WRITE.data.indication	91
B.2.8	WRITE.data.response	92
B.2.9	WRITE.indication	92
B.3	Read Service	93
B.3.1	Initiator	93
B.3.2	READ.request	93
B.3.3	READ.confirmation	93
B.3.4	Target	94
B.3.5	READ.authorisation.indication	94
B.3.6	READ.authorisation.response	95
B.3.7	READ.data.indication	95
B.3.8	READ.data.response	96
B.3.9	READ.indication	96
B.4	Read-Modify-Write Service	97
B.4.1	Initiator	97
B.4.2	RMW.request	97
B.4.3	RMW.confirmation	97
B.4.4	Target	98
B.4.5	RMW.authorisation.indication	98
B.4.6	RMW.authorisation.response	99
B.4.7	RMW.read.data.indication	99
B.4.8	RMW.read.data.response	100
B.4.9	RMW.write.data.indication	100
B.4.10	RMW.write.data.response	101
B.4.11	RMW.indication	102

Annex C (informative) Mapping to CCSDS SOIS Remote memory access service	103
---	------------

Bibliography.....	108
--------------------------	------------

Figures

Figure 5-1: Write Command format	23
Figure 5-2: Write Reply format.....	26
Figure 5-3: Write Command/Reply sequence	28

Figure 5-4: Write Command Header Error	30
Figure 5-5: Write Data Authorisation Rejection	32
Figure 5-6: Write Command Data Error	35
Figure 5-7: Write Reply Error.....	38
Figure 5-8: Read Command format	39
Figure 5-9: Read Reply format	41
Figure 5-10: Read Command/Reply sequence	44
Figure 5-11: Read Command Header Error.....	46
Figure 5-12: Read Authorisation Rejection	49
Figure 5-13: Read Reply Header Error	51
Figure 5-14: Read Reply Data Error	52
Figure 5-15: Read-Modify-Write Command format	53
Figure 5-16: Example Operation of Read-Modify-Write Command	56
Figure 5-17: Read-Modify-Write Reply format	57
Figure 5-18: Read-Modify-Write Command/Reply sequence	59
Figure 5-19: Read-Modify-Write Command Header Error.....	61
Figure 5-20: Read-Modify-Write Command Data Error	64
Figure 5-21: Read-Modify-Write Authorisation Rejection	66
Figure 5-22: Read-Modify-Write Reply Error.....	68
Figure 5-23: RMW Reply Data Error.....	69
 Figure B-1 : SOIS communication architecture	 88
Figure C-1 : RMAP model	103
Figure C-2 : SOIS model	104

Tables

Table 5-1: RMAP Command Codes	16
Table 5-2: Reply Address field size	18
Table 5-3: Example Reply Address field to Reply SpaceWire Address mappings.....	18
Table 5-4: Error and Status codes.....	71
Table 5-5: SpaceWire RMAP write command.....	74
Table 5-6: Example of Write Command Product Characteristics	74
Table 5-7: SpaceWire RMAP Read Command.....	75
Table 5-8: Example Read Command Product Characteristics	75
Table 5-9: SpaceWire RMAP Read-Modify-Write Command	76
Table 5-10: Example Read-Modify-Write Command Product Characteristics	76

Table C-1 : Comparison of RMAP and Remote Memory Access primitives	105
Table C-2 : WRITE.request parameters	106
Table C-3 : WRITE.confirmation parameters	106
Table C-4 : READ.request parameters	106
Table C-5 : READ.confirmation parameters.....	106
Table C-6 : RMW.request parameters	107
Table C-7 : RMW.confirmation parameters.....	107

Foreword

This document (EN 16603-50-52:2014) has been prepared by Technical Committee CEN/CLC/TC 5 "Space", the secretariat of which is held by DIN.

This standard (EN 16603-50-52:2014) originates from ECSS-E-ST-50-52C.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by March 2015, and conflicting national standards shall be withdrawn at the latest by March 2015.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN [and/or CENELEC] shall not be held responsible for identifying any or all such patent rights.

This document has been prepared under a mandate given to CEN by the European Commission and the European Free Trade Association.

This document has been developed to cover specifically space systems and has therefore precedence over any EN covering the same scope but with a wider domain of applicability (e.g. : aerospace).

According to the CEN-CENELEC Internal Regulations, the national standards organizations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

1

Scope

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

This Standard specifies the Remote Memory Access protocol (RMAP), which is one of these protocols that works over SpaceWire.

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 specified in this Standard.

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

2

Normative references

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

EN reference	Reference in text	Title
EN 16601-00-01	ECSS-S-ST-00-01	ECSS system - Glossary of terms
EN 16603-50-12	ECSS-E-ST-50-12	Space engineering - SpaceWire - Links, nodes, routers and networks
EN 16603-50-51	ECSS-E-ST-50-51	Space engineering - SpaceWire protocol identification

Terms, definitions and abbreviated terms

3.1 Terms defined in other standards

For the purpose of this Standard, the terms and definitions from ECSS-S-ST-00-01 and ECSS-E-ST-50-51 apply.

3.2 Terms specific to the present standard

None.

3.3 Abbreviated terms

The following abbreviations are defined and used within this standard:

Abbreviation	Meaning
CRC	cyclic redundancy code
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
SOIS	spacecraft onboard interface services
SpW	SpaceWire
SSNSAP	source subnetwork service access point

VHDL	vhsic hardware description language
VHSIC	very high speed integrated circuit

3.4 Conventions

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.

4

Principles

4.1 Remote Memory Access Protocol (RMAP) purpose

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 specified in this Standard.

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.

4.2 Guide to clause 5

Specification of the fields used in RMAP commands and replies is given in clause 5. The CRC used by RMAP is specified in clause 5.2. The write command is defined in clause 5.3, the read command in clause 5.4 and the read-modify-write command in clause 5.5. The error codes that are used in RMAP replies are listed in clause 5.6. The way in which partial implementations of RMAP can be implemented is described in clause 5.7. Clause 5.8 specifies the conformance statements i.e. 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 Annex A.

4.3 RMAP operations

4.3.1 Introduction

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.

4.3.2 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 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.

4.3.3 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.

4.3.4 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.

5

Requirements

5.1 RMAP command and reply fields

5.1.1 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.

5.1.2 Target Logical Address field

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

NOTE 1 The Target Logical Address field is normally set to a logical address recognised by the target.

NOTE 2 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 3 A target can have more than one logical address.

5.1.3 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.

5.1.4 Instruction field

5.1.4.1 General

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

5.1.4.2 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.

5.1.4.3 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 5-1.

Table 5-1: 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

Bit 5	Bit 4	Bit 3	Bit 2	Command Field
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

5.1.4.4 Reply Address length field

- a. 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.

5.1.5 Key field

- a. 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.

5.1.6 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 5-2.

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).

Table 5-2: 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

- 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.
- g. Some examples of the mapping between the contents of the Reply Address field and the Reply SpaceWire Address are listed in Table 5-3.

Table 5-3: 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

- h. 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.

- i. An RMAP implementation may use an implicit return address or implicit partial return address.

NOTE For example, a SpaceWire router with an RMAP configuration port can automatically route the reply to an RMAP command out of the same port that the RMAP command arrived on without the need for this being explicitly specified in the Reply Address field.

5.1.7 Initiator Logical Address field

- a. The Initiator Logical Address field shall be an 8-bit field that contains either:
 - 1. The logical address of the initiator of a command packet, if the initiator has a logical address, or
 - 2. 254 (0xFE) otherwise.

NOTE 1 The value 254 (0xFE) is the default logical address (see ECSS-E-ST-50-51, Clause 5.2.1).

NOTE 2 An initiator can have more than one logical address.

5.1.8 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.

5.1.9 Extended Address field

- a. 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.

5.1.10 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.

5.1.11 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.

5.1.12 Header CRC field

- a. 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.

5.1.13 Data field

- a. 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 clause 5.8).

5.1.14 Mask field

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

5.1.15 Data CRC field

- a. 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.

5.1.16 Reply SpaceWire Address field

- a. 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.

5.1.17 Status field

- a. The Status field shall be an 8-bit field in a reply containing a status/error code as defined in clause 5.6.

5.2 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:
 - (a) the n -bit input is defined to be the set of bits $B_{i,j}$ grouped into $n/8$ bytes where $i = \{0, 1, \dots, n/8 - 1\}$ is the byte index and $j = \{7, 6, \dots, 0\}$ is the bit index;
 - (b) 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 $i = n/8 - 1$;
 - (c) $m(x)$ is a polynomial $m_{n-1}x^{n-1} + m_{n-2}x^{n-2} + \dots + m_0x^0$ having binary coefficients m_i ;
 - (d) $m(x)$ 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;
 - (e) 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:

$$\begin{aligned} m_{n-1} &= B_{0,0}, m_{n-2} = B_{0,1}, m_{n-3} = B_{0,2}, \dots, m_{n-7} = B_{0,6}, m_{n-8} = B_{0,7}, \\ m_{n-9} &= B_{1,0}, m_{n-10} = B_{1,1}, m_{n-11} = B_{1,2}, \dots, m_{n-15} = B_{1,6}, m_{n-16} = B_{1,7}, \dots \\ m_7 &= B_{n/8-1,0}, m_6 = B_{n/8-1,1}, m_5 = B_{n/8-1,2}, \dots, m_1 = B_{n/8-1,6}, m_0 = B_{n/8-1,7} \end{aligned}$$

2. The procedure generates the remainder polynomial $r(x)$ given by the equation:

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

where $r(x) = r_7x^7 + r_6x^6 + \dots + r_0x^0$ 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 1 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 2 When a Galois version of a Linear Feedback Shift Register is used for CRC generation, its initial value is zero.

NOTE 3 Example VHDL and C-code implementations of this CRC algorithm are included in clause Annex A.

- d. If the CRC generation procedure is applied to the bytes covered by the CRC *excluding* the CRC byte then the generated CRC shall 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 shall be zero if no errors have been detected and non-zero if an error has been detected.

NOTE 1 When the codeword $c^*(x)$ is input to the CRC generator then the remainder is the syndrome:
$$s(x) = [c^*(x) \cdot x^8] \text{ modulo } g(x).$$

NOTE 2 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 as specified in ECSS-E-ST-50-12.

NOTE 1 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.

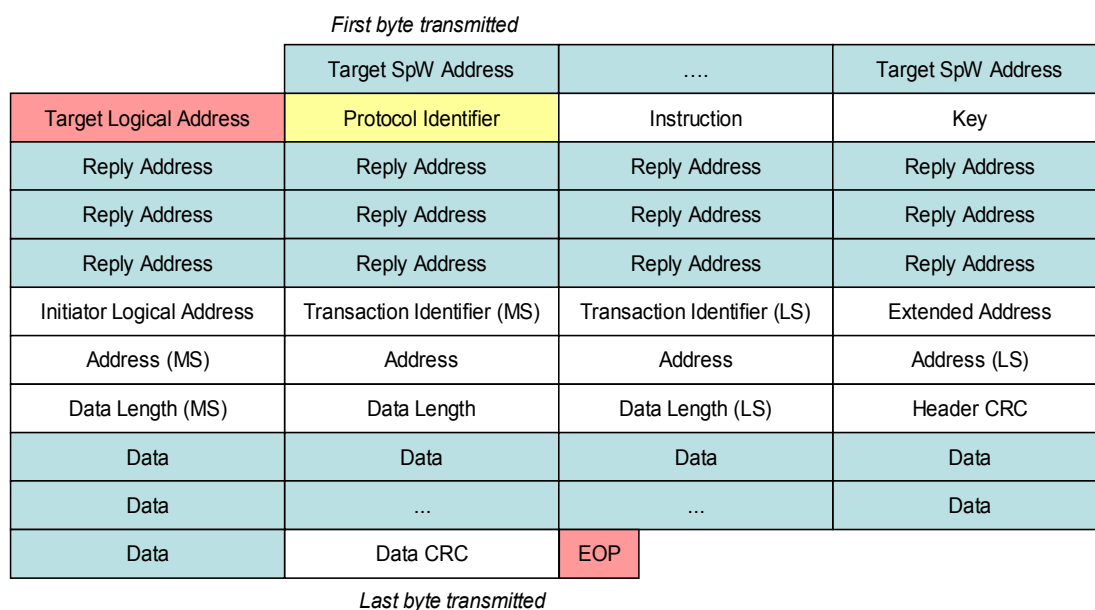
NOTE 2 See clause Annex A for some examples of how the CRC is implemented along with some test patterns.

5.3 Write Command

5.3.1 Write command format

5.3.1.1 Fields

- a. The write command shall contain the fields shown in Figure 5-1.



Bits in Instruction Field

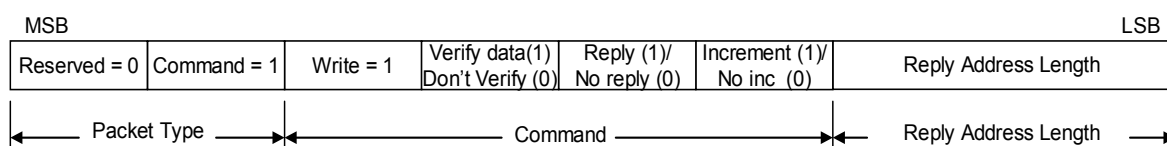


Figure 5-1: Write Command format

5.3.1.2 Target SpaceWire Address field

- a. The Target SpaceWire Address field shall be as defined in clause 5.1.1.

5.3.1.3 Target Logical Address field

- a. The Target Logical Address field shall be as defined in clause 5.1.2.

5.3.1.4 Protocol Identifier field

- a. The Protocol Identifier field shall be as defined in clause 5.1.3.

5.3.1.5 Instruction field

5.3.1.5.1 Instruction field format

- a. The Instruction field format shall be as defined in clause 5.1.4.

5.3.1.5.2 Packet type field

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

5.3.1.5.3 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, and
 - 2. Clear (0) if data is written to a single memory address.

5.3.1.5.4 Reply Address length field

- a. 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.

NOTE For example, if three Reply SpaceWire Address bytes are used then the Reply Address Length field is set to one (0b01).

5.3.1.6 Key field

- a. The Key field shall be as defined in clause 5.1.5.

5.3.1.7 Reply Address field

- a. The Reply Address field shall be as defined in clause 5.1.6.

5.3.1.8 Initiator Logical Address field

- a. The Initiator Logical Address field shall be as defined in clause 5.1.7.

5.3.1.9 Transaction Identifier field

- a. The Transaction Identifier field format shall be as defined in clause 5.1.8.

5.3.1.10 Extended Address field

- a. The Extended Address field shall be as defined in clause 5.1.9.
- b. The Extended Address field shall hold the most-significant 8-bits of the starting memory address to be written to.

5.3.1.11 Address field

- a. The Address field format shall be as defined in clause 5.1.10.
- 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.

5.3.1.12 Data Length field

- a. The Data Length field format shall be as defined in clause 5.1.11.

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.

5.3.1.13 Header CRC field

- a. The Header CRC field shall contain an 8-bit CRC as defined in clauses 5.1.12 and 5.2.

5.3.1.14 Data field

- a. The Data field shall contain zero or more bytes of data that are written into the memory of the target as defined in clause 5.1.13.

5.3.1.15 Data CRC field

- a. The Data CRC shall contain an 8-bit CRC as defined in clauses 5.1.15 and 5.2.

5.3.1.16 EOP character

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

5.3.2 Write reply format

5.3.2.1 Format

- a. The format of the reply to a write command shall contain the fields shown in Figure 5-2.

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.

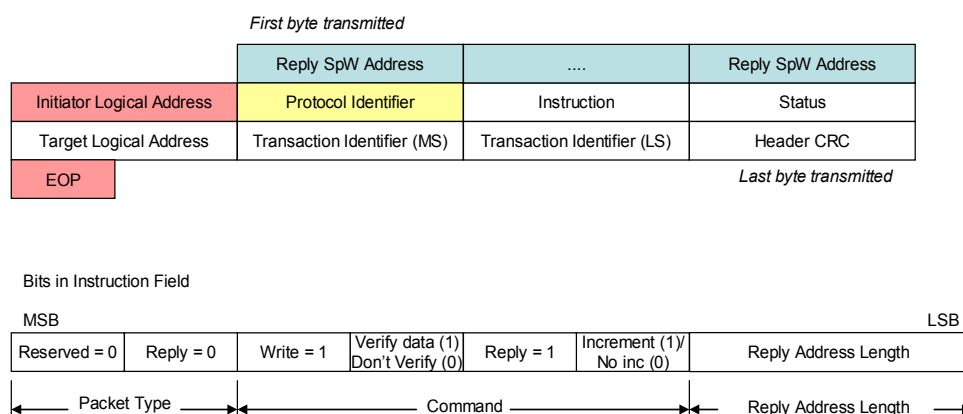


Figure 5-2: Write Reply format

5.3.2.2 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 clause 5.1.6.

5.3.2.3 Initiator Logical Address field

- a. The Initiator Logical Address field shall be as defined in clause 5.1.7.

5.3.2.4 Protocol Identifier field

- a. The Protocol Identifier field shall be as defined in clause 5.1.3.

5.3.2.5 Instruction field

- a. The Instruction field format shall be as defined in clause 5.1.4.
- 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, clause 5.3.1.5.3.
- d. The Reply Address Length field shall be set to the same value as in the Reply Address Length field of the write command, clause 5.3.1.5.4.

5.3.2.6 Status field

- a. The Status field format shall be as defined in clause 5.1.17.
- 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 clause 5.6.

5.3.2.7 Target Logical Address field

- a. The Target Logical Address field shall be set to either of:
 - 1. the value of the Target Logical Address field of the write command, see clause 5.3.1.3, or
 - 2. a logical address of the target.

NOTE Normally these are the same.

5.3.2.8 Transaction Identifier field

- a. The Transaction Identifier field shall be set to the same value as the Transaction Identifier in the write command, see clause 5.3.1.9.

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

5.3.2.9 Header CRC field

- a. The Header CRC field shall contain an 8-bit CRC as defined in clauses 5.1.12 and 5.2.

5.3.2.10 EOP character

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

5.3.3 Write action

5.3.3.1 Overview

The normal sequence of actions for a write command is illustrated in Figure 5-3.

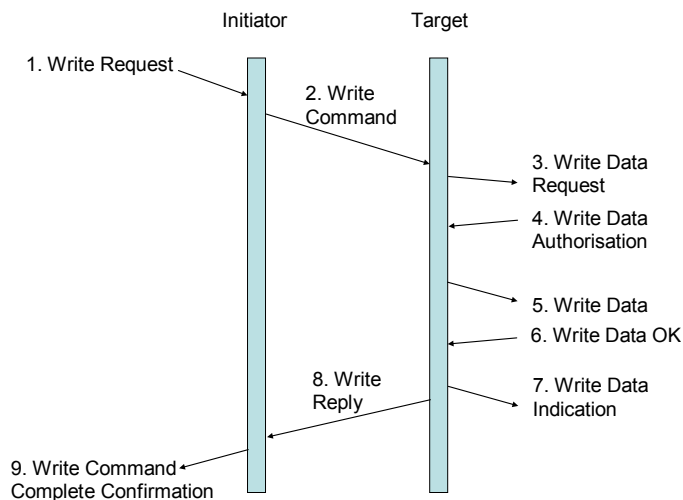


Figure 5-3: Write Command/Reply sequence

5.3.3.2 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

5.3.3.3 Write command

- a. 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.

5.3.3.4 Write data request

5.3.3.4.1 Protocol identifier

- 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.

5.3.3.4.2 Incomplete header

- a. If an EEP or EOP is received before the complete header including header CRC has been received the target shall:
 - 1. Discard the entire packet,
 - 2. Not send a reply packet.
- b. If an EEP or EOP is received before the complete header including header CRC has been received the target should update the error information to reflect the “EEP” or “Early EOP” error if the target supports error information gathering.

5.3.3.4.3 Error End of Packet

- a. If an EEP is received immediately after the complete header including header CRC has been received the target shall:
 - 1. Discard the entire packet,
 - 2. Not send a reply packet.
- b. If an EEP is received immediately after the complete header including header CRC has been received the target should update the error information to reflect the “EEP” error if the target supports error information gathering.

5.3.3.4.4 Header CRC check

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

5.3.3.4.5 Header CRC error

- a. When checking the Header CRC indicates an error in the header the target shall:
 - 1. Discard the entire packet,
 - 2. Not send a reply packet.
- b. When checking the Header CRC indicates an error in the header the target should update the error information to reflect the “Header CRC” error if the target supports error information gathering.

NOTE The sequence of events that occurs when there is a CRC error in the header of the write command is illustrated in Figure 5-4.

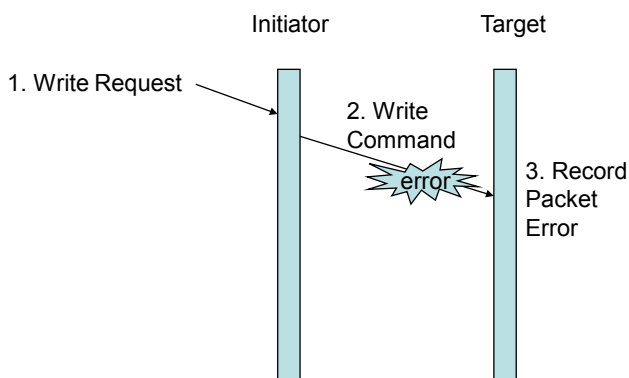


Figure 5-4: Write Command Header Error

5.3.3.4.6 Unused packet type

- a. When checking the Header CRC indicates no error present in the header and if the Instruction field contains an unused packet type (0b10 or 0b11), the target shall:
 1. Discard the command packet,
 2. Not send a reply.
- b. When checking the Header CRC indicates no error present in the header and if the Instruction field contains an unused packet type (0b10 or 0b11), the target should update the error information to reflect the “unused RMAP packet type or command code” error if the target supports error information gathering.
- c. When checking the Header CRC indicates no error present in the header and if the Instruction field contains an unused packet type (0b10 or 0b11), the target may send a reply containing an “unused RMAP packet type or command code” error as specified in clause 5.6 to the node specified in the Reply Address and Initiator Logical Address fields, if a reply has been requested (Reply bit set).

5.3.3.4.7 Invalid command code

- a. When checking the Header CRC indicates no error present in the header and if the Instruction field contains an invalid command code as specified in Table 5-1, the target shall:
 1. Discard the command packet,
 2. Return an “unused RMAP packet type or command code” error as specified in clause 5.6 to the node specified in the Reply Address and Initiator Logical Address fields, if a reply has been requested (Reply bit set).
- b. When checking the Header CRC indicates no error present in the header and if the Instruction field contains an invalid command code, as specified in Table 5-1 the target should update the error information to reflect the “unused RMAP packet type or command code” error if the target supports error information gathering.

5.3.3.4.8 Write data request action

- a. When checking the Header CRC indicates no error present in the header and 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:
 1. Target Logical Address
 2. Instruction
 3. Key
 4. Initiator Logical Address
 5. Transaction Identifier
 6. Extended Address
 7. Memory address
 8. Data Length

5.3.3.5 Write data authorisation

5.3.3.5.1 Write operation authorisation

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

5.3.3.5.2 Invalid key

- a. If the value of the Key is not the value expected by the target user application, the target shall:
 1. Discard the command packet,
 2. Return an “invalid key” error as specified in clause 5.6 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).
- b. If the value of the Key is not the value expected by the target user application, the target should update the error information to reflect the “invalid key” error if the target supports error information gathering.

5.3.3.5.3 Invalid logical address

- a. If the Target Logical Address is not a logical address recognised by the target user application, the target shall:
 1. Discard the command packet,
 2. Return an “invalid Target Logical Address” error as specified in clause 5.6 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).
- b. If the Target Logical Address is not a logical address recognised by the target user application, the target should update the error information to reflect the “invalid Target Logical Address” error if the target supports error information gathering.

5.3.3.5.4 Command rejection

- a. If the command is not accepted by the target user application for any other reason, the target shall:
 1. Discard the command packet,
 2. Return a “RMAP command not implemented or not authorised” error as specified in clause 5.6 to the node specified in the Reply Address and Initiator Logical Address fields if a reply has been requested, Reply bit set (1).
- b. If the command is not accepted by the target user application for any other reason, the target should update the error information to reflect the “RMAP command not implemented or not authorised” error if the target supports error information gathering.

NOTE 1 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 2 The sequence of events that occurs when a write command is not authorised is illustrated in Figure 5-5.

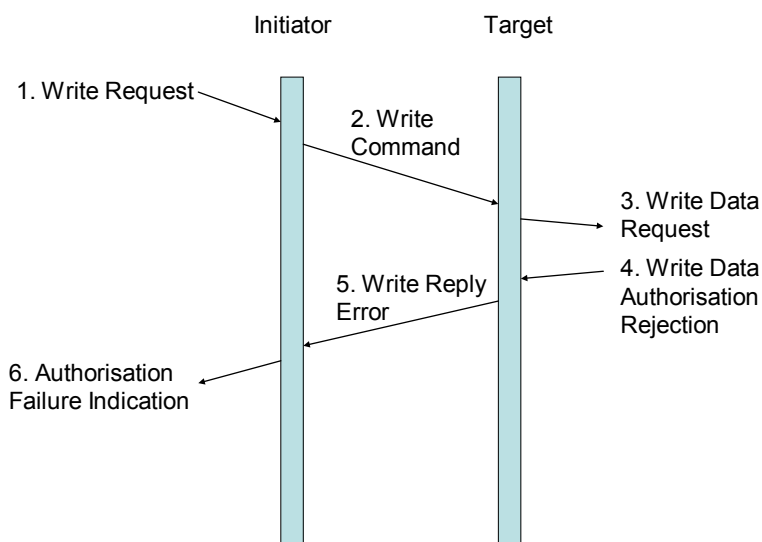


Figure 5-5: Write Data Authorisation Rejection

5.3.3.6 Write data

5.3.3.6.1 Write data action

- 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 5-3).