UNISIG

ERTMS/ETCS
STM FFFIS Safe Link Layer
REF : SUBSET-057 ISSUE : 3.0.0 DATE : 2012-02-29

Company	Technical Approval	Management Approval
ALSTOM		
ANSALDO STS		
BOMBARDIER		
INVENSYS RAIL		
SIEMENS		
THALES		

1. MODIFICATION HISTORY

Issue Number Date	Section Number	Modification / Description	Author
0.0.1 07-02-00		Created from Siemens internal document P25020-B7300-A1-8-7629; Rev. 08; Date of issue 26-06-98. Added non-fail-safe communication. New algorithm for Authentication. Added version control. Removed Enable and Disable telegrams.	N. Lindström
1.0.0 21-02-00		Updated after comments from Unisig meeting 17-02-00 in Stuttgart: • New concept for redundancy (send all telegrams on both busses if redundant bus is attached); • New concept for retransmission (send all telegrams twice on each bus); • New command numbers for nonfail-safe telegrams; • Transition table added; • Added an informal appendix describing provided services; • Editorial updates.	N. Lindström
1.0.1 23-02-00		Updated after comments from Alstom and Siemens • Removed authentication for nonfail-safe connections • Corrected command number for nonfail-safe telegrams in paragraph 5.1.1.5. • Added paragraph 5.6.1.1 • Added paragraph 8.1.1.1. • Editorial improvements.	N. Lindström
1.0.2 28-03-00		Updated after comments from Unisig meeting 22-03-00 in Paris and CENELEC meeting 09-03-00: • Added section for definitions and improved several definitions. • Updated the document to the new redundancy concept • Removed the non-safe communication in favour for low safe communication. • Removed the CRC-32 for safe communication. The CRC is to be defined. • Added a CRC-16 for low safe communication. • Added message format for multicast telegrams. Implicit data is sent explicit for multicast telegrams (measure for hazard: incorrect ID). • Added the functionality that a connection can be disconnected by “application” and reconnected thereafter. • Improved disconnect_reason in the Disconnect Telegram. • Idle_cycle_interval is sent in the Connect Request and Connect Confirm telegrams. • Added Dual_bus in the Connect Request and Connect Confirm telegrams, make it possible to monitor if the connection is using one or two busses. • Clarified Isolation (Failure mode) and disconnection of a connection. • Editorial improvements.	N. Lindström
2.0.0 30-03-00	ReF	5.1.3.6, 5.9.1.3, 5.9.1.4, 5.10.1.2, 5.10.1.3, Final Issue to ECSAG	D. Degavre (Ed)

Issue Number Date	Section Number	Modification / Description	Author
2.0.1 2002-01-11		Updated according to the review comments of the UNISIG Workgroup STM listed in “Unisig_COM_WP_STM_SLL”, V0.0.5, 2002-01-11	P. Luehrs (Siemens)
2.0.2 2002-01-24		Updated after comments from the UNISIG Workgroup STM (Meeting 2002-01-22/23 in Stuttgart) see “Unisig_COM_WP_STM_SLL”, V0.0.6, 2002-01-24	P. Luehrs (Siemens)
2.0.3 2002-02-18		Updated according to the review comments of the UNISIG Workgroup STM listed in “Unisig_COM_WP_STM_SLL”, V0.0.7, 2002-02-18	P. Luehrs (Siemens)
2.0.4 2002-03-01		Updated during the Meeting of the UNISIG Workgroup STM in Madrid (2002-02-28 .. 2002-03-01)	P. Luehrs (Siemens)
2.0.5 2002-MAR-08		Updated according to agreed changes resulting from the UNISIG Workgroup STM in Madrid (2002-02-28 .. 2002-03-01)	H. Schweißthal (Siemens)
2.0.6 2002-03-14		Updated according agreed changes in the 14th of March 2002 meeting	J.-Y. Riou (Ansaldo)
2.0.7 2002-03-20	5.1.1.5 5.3.1.8 5.3.1.9 5.3.1.10 5.4.1.5	Homework due to according agreed changes in the 14th of March 2002 meeting	H. Schweißthal (Siemens)
2.0.8 2002-04-11	3.3.1.1 5.1.1.5.1 5.1.1.6 5.3.1.6 5.3.1.6.3 5.3.1.8 6.1.1.5 6.1.1.6 6.2.1.6 7.2	Updated during the Meeting of the UNISIG Workgroup STM in Stuttgart (2002-04-11/12)	P. Luehrs (Siemens)
2.0.9 2002-04-16	5.3.1.9 5.3.1.10 5.4.1.5	Homework according to agreed comments of the Meeting of the UNISIG Workgroup STM in Stuttgart (2002-04-11/12)	P. Luehrs (Siemens)
2.0.10 2002-05-14	3.2.1.4 5.2.2.1 5.2.3.4 5.2.4.1 5.2.4.2 5.2.5.6 5.2.5.9.1 5.2.6.4 5.3.1.7 5.5.1.3.2 5.5.1.4 5.5.1.5 5.5.1.6, 5.5.1.7 6.2.1.6 10.1.1.2	Updated according to agreed comments of the Meeting of the UNISIG Workgroup STM in Brussels (2002-05-14/15)	P. Luehrs (Siemens)
2.0.11 2002-06-12	5.2.1.4 5.2.5.9 5.5.1.7 9	Updated according to agreed comments of the Meeting of the UNISIG Workgroup STM in Braunschweig (2002-06-11/12)	P. Luehrs (Siemens)

Issue Number Date	Section Number	Modification / Description	Author
2.0.12		Updated according to agreed comments of the Meeting of the UNISIG Workgroup STM in Madrid (2002-06-24)	B. Muñoz
2.1.0		Editorial change 2002-06-27	B. Muñoz
2.1.1 2002-11-01	5.3.1.8 5.3.1.9 5.3.1.10 5.3.2.1 5.4.1.5 5.4.1.6 5.4.2.1 8.2.1.3 10 11 (new)	CRC polynomial and application rules together with the calculation of the CRC length included (according to the decision of the UNISIG Workgroup STM in Brussels (2002-10-23)) and editorial changes (e. g. references)	P. Luehrs (Siemens)
2.1.2 2002-12-05	3.3.1.4.4 5.3.1.7 5.3.1.8 6.1.1.3.1 6.2.1.6 10.2	Correction of the examples for the CRC; Example for the calculation of a SIL 2 CRC added; Examples for the checksum validation added.	P. Luehrs (Siemens)
2.1.3 2003-02-11	3.2.1.5, 11.3.3.4.2, 5.2.7.2, 8.2.1.3	Updated during the Meeting of the UNISIG Workgroup STM in Stuttgart (2003-02-11)	P. Luehrs (Siemens)
2.1.4 2003-02-25	3.2.1.5, 4.1.1.5, 8.2.1.3, 11.3.3.4.2	Updated during the Meeting of the UNISIG Workgroup STM in Madrid (2003-02-25)	P. Luehrs (Siemens)
2.2.0 2003-04-11		UNISIG release	P. Luehrs (Siemens)
2.2.1 2005-04-06	5.2.7.2, 6.2.1.7 5.1.1.5, 5.2.1.3, 5.2.2.2, 5.2.7.2, 6.2.1.7 5.1.3.4, 6.2.1.5 5.2.5.9 5.3.1.8 5.3.1.9, 5.3.1.10, 5.3.1.11, 5.4.1.5 5.5.1.3, 5.5.1.6, 5.5.1.7 8	WG-1: Command values for Multicast Telegrams corrected in the telegram structures. WG-4: Reference to CENELEC 50 170 added for the length. WG-7: Minimum value for “Data_length” corrected. WG-10: Inconsistency resolved: “Disconnect_reason_text” shall have a length 0..40. WG-5: Telegram length corrected and CRC updated. WG-9: The content / structure of the telegrams updated to indicate that more information is transmitted (e. g. “version number”). WG-11: Final disconnect after request from upper layer included in the state table for SL 0 connections (see 5.2.5.6). Compatibility Number changed to 3.1.Z, as the compatibility is not affected.	P. Luehrs (Siemens)

Issue Number Date	Section Number	Modification / Description	Author
2.3.0 2010-12-17	5.2.5.6 5.2.5.6.1 5.5.1.3 5.5.1.3.1 5.5.1.5 5.5.1.6 5.5.1.7 5.2.5.7.2 5.2.5.8.3 11.1.1.1.1 11.1.1.1.2 11 11.3.5.1 11.4.2.3.1 11.4.5.1 5.3.1.8 5.2.5.7.3 5.2.5.8.4 5.5.1.5 5.5.1.5 First page Header	WG-2: Made bullet list, adding “incompatible versions” and “final disconnect received” Added note on designers choice of Defect / Idle Added Conn_err” and “Auth_err” and ”with compatible version number”, “Correct” and “Non final” to descriptions of relevant events. Added “in a connection with higher Safety Level. For Safety Level 0 connections, all incorrect receives are considered first incorrect receive” to end of sentence. Added new rows to table: Fdiscon_rcv, Conn_err and Auth_err. Added note on designers choice between Defect and Idle. Added row Send final Disconnect Telegram Added rows FDiscon_snd, FDiscon_Rcv and Conn_err and column Defect Added note on designers choice between Defect and Idle. WG-8: Deleted third bullet: Error upon transmission of the Authentication Telegram or the Authentication Acknowledgement Telegram Deleted second bullet “Error upon transmission of the Connect Request Telegram or the Connect Confirm Telegram. WG-12: changed “32 digits” to “48 bits”, added “, otherwise further analysis is needed” Changed “24 digits” to “32 bits”, added “, otherwise further analysis is needed” Changed all numerical calculations of undetected failure probabilities and MTBF to correspond to new longer codes Changed “digits” to “bits” Changed “digits” to “bits” and “23” to “24” Changed “digits” to “bits” WG-13: Corrected “Data_length” to 06h WG-14: Added new bullet: Authentication Telegram received on Safety level 0 connection or telegram with undefined command code received Added new bullet: Authentication Telegram received on Safety level 0 connection or telegram with undefined command code received Changed table entries for states wConAck/wAuthReq/wAuthAck and event “Timeout_rcv” from **Idle** to **no change** and **f_07** to **f_00** WG-16: "F_xx" changed to "f_xx" WG-17: Updated company names to Ansaldo STS and Thales. Update to new UNISIG document external template 2009	B Sjöbergh (Ansaldo-STS)
2.9.1 2012-01-21	5.2.5.9	WG-18: 4..4 replaced by 5..5 to avoid overwriting previous field	B Sjöbergh (Ansaldo-STS)

Issue Number Date	Section Number	Modification / Description	Author
		WG-19: **CR 1043**	& Thomas Mandry (Alstom)
	1	Delete note on compatibility under history.	
	5.2.1.3, 5.2.2.2	Modify "Version" part of Connect Request & Connect Confirm telegrams	
	5.2.2.1	Remove condition on Compatibility Version	
	5.2.5.6, 5.2.5.8.2	No more final disconnection for incompatible versions	
	5.2.5.7.3, 5.2.5.8.4	“telegram with reserved command code received” is no more a reason for disconnection	
	5.5.1.3, 5.5.1.5, 5.5.1.7	Remove Conn_err event	
	5.2.5.9	Reason for disconn. 01h no more "bad version error"	
	6.2.1.6, 6.2.1.7	Modify "Compatibility" and "CRC" parts	
	5.1.1.5.2, 6.2.1.8	Add requirements about reserved command numbers	
	10.2	Update with this new telegram content	
	8	Delete chapter "Configuration management"	
	whole doc	WG-20: General editorial changes throughout the document: Requirements to be stated with “shall”, not with “is” (as per ERA comment #1 on version 2.3.0) ; Some unclear requirements have been reworded for better clarity.	
	11 (all)	WG-21: Replace appendix B content with calculations for the implemented codes.	
	3.2	Referenced documents: 1 update & 1 addition	
	5.2.5.8.4	Add reference to appendix B	
	5.2.5.7.6, 5.2.5.7.7, 5.2.5.8.5	WG-22: Add conditions "reception of Disconnect telegram" to enter in disconnected state, for consistency with transition tables	
	4.1.1.5, 10.1.2.1.1	WG-23: Deletion of useless notes about Profibus	
	3.5.1.2	WG-24: Clarification of the time-stamp used by Application (as per ERA comment #3 on version 2.3.0)	
	3.3.1.1.1, 3.3.1.2	WG-25: Changes resulting from re-defined Functions at Application level	
	5.1.1.5, 5.1.1.5.1, 5.2.5.7.3 5.2.5.8.4, 5.2.7.2, 6.2.1.7	WG-26: Clarification of the use of command codes	
	5.2.5.8.4	WG-27: No 2nd incorrect receive on SL0	
	5.2.6.2, 5.5.1.7	WG-28: Correction of inconsistency related to sending of idle telegrams (action f_06 is now used in state wConAck for the event Timeout_snd)	
	6.1.1.4	WG-29: Deletion of requirement useless and not possible to implement strictly	
	12.1.1.2	WG-30: Improve the informative interface to SLL	
	3.2	Referenced document integrated in a table (no revision marks)	

Issue Number Date	Section Number	Modification / Description	Author
2.9.2 2012-02-17	Front page 1 3.1 6.2.1.3 12.1.1.2.4	Update according to ERA/SG review remarks for v2.9.1: Removed 'Baseline 3' in top line Correction of Modification History according to SG review remark #2: WG-14 of v2.3.0 & WG-28 of v2.9.1 ERA review remark #2 on v2.9.1: JRU removed WG-31: "lower than" replaced by "lower than or equal to" WG-30b: Improve the informative interface to SLL	B Sjöbergh (Ansaldo-STS)
3.0.0 2012-02-29	No change	Baseline 3 release version	Thomas Mandry (Alstom)

2. TABLE OF CONTENTS

3. GENERAL

3.1 Abbreviations

3.1.1.1	CRC	Cyclical Redundancy Check
3.1.1.2	PROFIBUS	PROcess Field BUS
3.1.1.3	SAP	Service Access Point
3.1.1.4	FDL	Field Data Link
3.1.1.5	DMI	Driver Machine Interface
3.1.1.6	SL	Safety Level

3.2 References

Ref. N°	Document Reference	Title
/1/	CENELEC EN 50159 (2010)	Railway applications - Communication, signalling and processing systems - Safety-related communication in transmission systems
/2/	CENELEC EN 50170-2 (1996)	PROFIBUS
/3/	SUBSET-035	Specific Transmission Module FFFIS
/4/	SUBSET-056	STM FFFIS Safe Time Layer
/5/		Einsatz des Profibus für sicherheitsrelevante Anwendungen (Martina Barten / Jens Braband / Harald Peters) published in „Signal + Draht (90)4/98“
/6/		Quantitative assessment of safety codes (Lucie Karna / Stepan Klapka / Magdalena Harlenderova) published in FORMS/FORMAT 2008 conference proceedings

3.3 Definitions

3.3.1.1 A node is a physical device that communicates through STM busses. A node has one physical bus address but may have several logical addresses (Service Access Point, SAP).

3.3.1.1.1 A node can contain several **logical function**s defined in /3/ (e.g. the same node will be able to gather logical functions Odometer, Juridical Data).

3.3.1.2 Intentionally deleted

3.3.1.3 To ensure correct operation of the communication between two nodes, a **connection** is established:

3.3.1.3.1 A **point-to-point connection** is used for the communication between two nodes.

3.3.1.3.2 A **multicast message** is a message simultaneously sent to all nodes.

3.3.1.4 A point-to-point connection may have different safety levels:

3.3.1.4.1 A point-to-point connection with **Safety Level 0 (SL 0)** is a connection between two nodes whereof one node is non-safe (SIL 0).

3.3.1.4.2 A point-to-point connection with **Safety Level 2 (SL 2)** is a connection which shall be used by nodes with at least SIL 1.

3.3.1.4.3 A point-to-point connection with **Safety Level 4 (SL 4)** is a connection which shall be used by nodes with at least SIL 3.

3.3.1.4.4 A SIL 4 equipment shall be able to handle point-to-point connections with any Safety Level and a SIL 2 equipment shall be able to handle point-to-point connections with SL 2 and SL 0.

3.3.1.5 A **station** is the part of a device relating to the processing of one bus. The term station is defined in /2/.

3.3.1.6 The communication between two nodes is done via one or two busses. The node may be equipped with one or two stations.

3.3.1.7 **Logical connection master** is defined as the node sending the connection request telegram for a point-to-point connection in the Safe Link Layer.

3.3.1.8 Logical connection Master is the master of a logical connection. A logical connection has one and only one master.

3.3.1.9 **Logical connection slave** of point-to-point connection is defined as the node sending the Connect Confirm telegram in the Safe Link Layer.

3.3.1.10 Logical connection Slave is the slave of a logical connection. A point-to-point logical connection has one and only one Logical connection Slave.

3.3.1.11 A Multicast connection has one **Multicast Sender**.

3.3.1.12 A Multicast connection may have zero, one or several **Multicast Receivers**.

3.4 Summary Description

3.4.1.1 PROFIBUS is an open field bus to which any PROFIBUS-compatible equipment can be attached. To satisfy the requirements of a single-channel safety-related transmission system, it is necessary to use an additional safety layer. The protocol for

data transmission with error detection specified in this document enables the relevant requirements of the CENELEC standard /1/ to be met.

3.4.1.1.1 Note: The requirements of the CENELEC standard are not achieved by the Safe Link Layer alone. Only the complete set of specifications

- STM FFFIS (SUBSET-035),
- STM FFFIS Safe Time Layer (SUBSET-056) and
- STM FFFIS Safe Link Layer (SUBSET-057)

together fulfil all the listed requirements.

3.4.1.2 The protocol for data transmission with error detection provides a method of error detection for individual telegrams. The telegram error detection system uses unique sender/receiver identification, a 32-bit sequence number and a CRC error detection suffix.

3.4.1.3 A requirement for the protocol for data transmission with error detection is to set up connections between nodes and monitor these connections cyclically.

3.4.1.4 For increased availability, a redundant configuration of PROFIBUS can be implemented (physical diversity). All telegrams are sent on both busses.

3.4.1.5 For increased availability, the following provisions are provided:

3.4.1.5.1 Multicast telegrams are sent twice from the Safe Link Layer to the lower layer.

3.4.1.5.2 Point-to-point telegrams are sent once from the Safe Link Layer to the lower layer but with the PROFIBUS FDL ACK / NACK protocol with one retransmission.

3.4.1.6 To establish an SL 4 or an SL 2 point-to-point connection, two steps have to be performed:

1. connection set-up with the exchange of random numbers.
2. authentication, to ensure the safety property of the communication partners

3.4.1.7 To establish an SL 0 connection only the connection set-up has to be performed.

3.4.1.8 The Safe Link Layer provides communication with three levels of safety (SL):

1. Safety Level 4 (SL 4)
2. Safety Level 2 (SL 2)
3. Safety Level 0 (SL 0)

3.4.1.8.1 Justification: According to the CENELEC standard /1/ the command numbers, sequence numbers, SAP and CRC have to be different for connections with different safety levels.

3.4.1.9 All nodes that are physically connected to the bus have the safety requirement that they shall not implement any SL protocol corresponding to a higher Safety Integrity Level (SIL) than their own SIL.

3.5 Scope of this Document

3.5.1.1 This document describes the protocol for data transmission with error detection and the PROFIBUS FDL services used to transfer the data to the PROFIBUS FDL driver. The mode of transfer and variant of the PROFIBUS interface used depend on the implementation and are not specified in this document. The PROFIBUS itself and PROFIBUS protocol are specified in /2/. PROFIBUS FDL parameters are out of the scope of this specification, but defined in /3/.

3.5.1.2 Note: the time stamp functionality shown in the diagram on application level uses the STL layer Reference Time service.

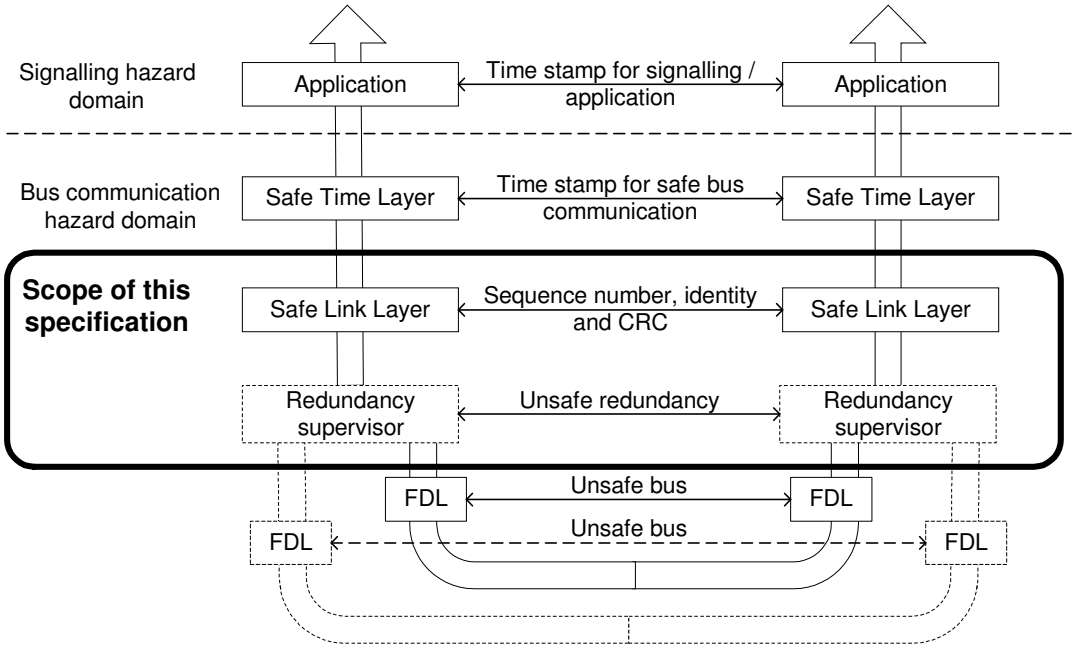


Figure 1 Scope of the protocol specification

3.5.1.3 The redundancy supervisor is a non-safe function that is only required if the node is equipped with dual busses. The redundancy supervisor is specified in section 9.

3.5.1.4 Dual busses is not mandatory.

4. PROFIBUS INTERFACE (FDL)

4.1.1.1 The protocol defined here is designed for PROFIBUS FDL masters for multicast and point-to-point communication.

4.1.1.2 All the point-to-point telegrams for data transmission are transferred using the same PROFIBUS FDL service.

4.1.1.3 Point-to-point telegrams shall be transferred by means of send_data_with_acknowledge (SDA) with one retransmission.

4.1.1.4 Multicast telegrams shall be transferred by means of send_data_with_no_acknowledge (SDN)

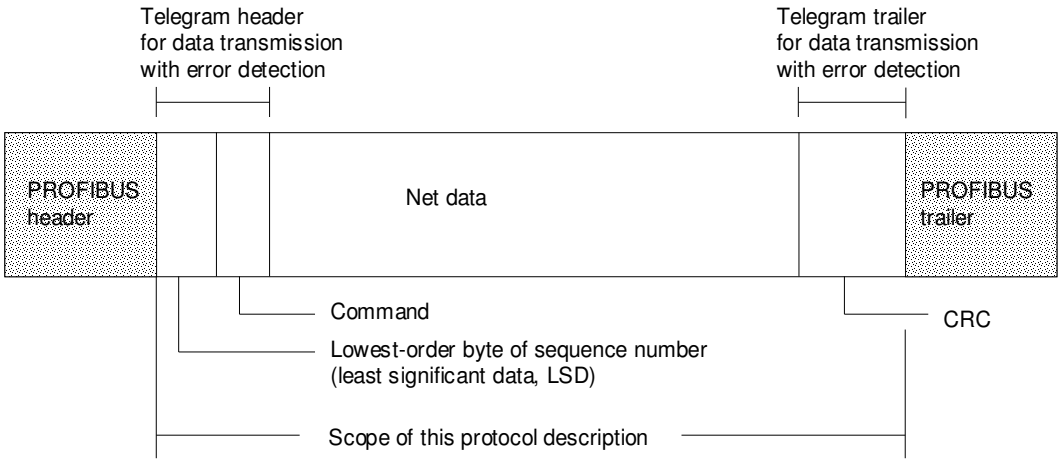
4.1.1.5 Intentionally deleted

5. POINT-TO-POINT CONNECTIONS

5.1 General Structure of a Telegram

5.1.1.1 The telegrams shall have the following fixed telegram frame:

5.1.1.2



The black fields are the PROFIBUS FDL header and trailer.

Figure 2 General structure of a telegram

5.1.1.3 Note: The PROFIBUS FDL header and trailer are part of the PROFIBUS FDL protocol and can be found in the CENELEC standard /2/. A detailed description of the elements of the PROFIBUS FDL protocol is not given here.

5.1.1.4 Detailed protocol structure of the telegram:

5.1.1.5

Byte	Designation	Range of values	Meaning
1	Lowest_order_byte_ sequence_number	0..FFh	Of the 32-bit sequence number only the 8 lowest-order bits are transmitted here. The 24 higher-order bits are used for CRC formation only (implicit data).
2	Command	80h.BFh	Command for **SL 4** data transmission with error detection: 80h: connect request telegram 81h: not used and reserved 82h: connect confirm telegram 83h: authentication telegram 84h: authentication acknowledgement telegram 85h: disconnect telegram 86h: idle telegram 87h: not used and reserved (send disable telegram) 88h: not used and reserved (send enable telegram) 89h: telegram for upper layer 8Ah: not used and reserved (redundancy switchover telegram) 8Bh: not used and reserved (redundancy switchover acknowledgement) 8Ch: not used and reserved 8Dh: multicast telegram for upper layer 8Eh..9Fh: reserved for future extension of the Safe Link Layer A0h..BFh: telegram for upper layer
		00h..3Fh	Command for **SL 2** data transmission with error detection: 00h: connect request telegram 01h: not used and reserved 02h: connect confirm telegram 03h: authentication telegram 04h: authentication acknowledgement telegram 05h: disconnect telegram 06h: idle telegram 07h: not used and reserved (send disable telegram) 08h: not used and reserved (send enable telegram) 09h: telegram for upper layer 0Ah: not used and reserved (redundancy switchover telegram) 0Bh: not used and reserved (redundancy switchover acknowledgement) 0Ch: not used and reserved 0Dh: not used and reserved 0Eh..1Fh: reserved for future extension of the Safe Link Layer 20h..3Fh: telegram for upper layer

Byte	Designation	Range of values	Meaning
		C0h..FFh	Command for **SL 0** data transmission with error detection: C0h: connect request telegram C1h: not used and reserved C2h: connect confirm telegram C3h: not used and reserved (authentication telegram) C4h: not used and reserved (authentication acknowledgement telegram) C5h: disconnect telegram C6h: idle telegram C7h: not used and reserved (send disable telegram) C8h: not used and reserved (send enable telegram) C9h: telegram for upper layer CAh: not used and reserved (redundancy switchover telegram) CBh: not used and reserved (redundancy switchover acknowledgement) CCh: not used and reserved CDh: not used and reserved CEh..DFh: reserved for future extension of the Safe Link Layer E0h..FFh: telegram for upper layer
3.. (n-l)	Net_data		In the data telegram the application data is transferred here. In internal telegrams, e.g. connect request telegrams, the structure of the net data is command-specific (see sections below). n =< 244 bytes (max. length of “DATA_UNIT”, see /2/); for 'l' see below
(n-l+1) .. n	CRC	as per safety level	l: length of CRC checksum:

5.1.1.5.1 Telegrams with command numbers marked as "Telegram for upper layer" shall be handled in the SLL as data telegrams or as multicast telegrams depending on the type of connection.

5.1.1.5.2 Telegrams with command numbers reserved for future extension shall be ignored by the receiver.

5.1.1.6 **The format of the data added by the Safe Link Layer shall be the little endian format low-order byte first**; i.e. 16- and 32-bit values for example are transferred bytewise, and the bytes are transferred as follows:

16-bit values: lowbyte, highbyte

32-bit values: lowword.lowbyte, lowword.highbyte, highword.lowbyte, highword.highbyte

5.1.2 Sequence Number

5.1.2.1 The sequence number shall be a 32-bit value which is incremented with each telegram. The sequence number shall be connection-specific, i.e. separate sequence numbers are maintained for each connection. Distinct, independent sequence numbers shall be used for the send and receive directions.

5.1.2.2 Sequence numbers shall be used for authentication in order to meet relevant requirements of /1/.

5.1.3 CRC Checksum computation

5.1.3.1 The CRC checksum length in bits, the polynomial and the initial value are different for the three SLs. The correct algorithm for the Safety Level of the connection shall be used.

5.1.3.2 The checksum shall be computed on the following data areas:

- implicit data
- telegram header for data transmission with error detection
- net data

5.1.3.3 The implicit data used to compute the checksum is not contained in the data-transmission-with-error-detection telegram. PROFIBUS FDL supplies the following implicit data:

- data length
- receive address
- send address
- partner's service access point
- local service access point

5.1.3.4 Structure of the implicit data:

Byte	Designation	Range of values	Meaning
1	Data_length	2..244	The length of the user data bytes (“DATA_UNIT” from the PROFIBUS FDL viewpoint) in the telegram, i.e. length of telegram header for data transmission with error detection + length of net data + length of telegram trailer for data transmission with error detection
2	Receiver_address	0..126	The receiver's address can be handled as implicit data because it is available on the PROFIBUS FDL.
3	Sender_address	0..126	The sender's address can be handled as implicit data because it is available on the PROFIBUS FDL.
4	DSAP	0..62	The partner's service access point can be handled as implicit data because it is available on the PROFIBUS FDL.
5	SSAP	0..62	The local service access point can be handled as implicit data because it is available on the PROFIBUS FDL.
6..8	Higher_order_24bits_of_sequence_number	000000h.. FFFFFFh	Except for the lowest-order byte, the sequence number does not need to be transmitted. The receiver adds the known higher-order bits of the sequence number to the lowest-order byte transferred.

5.1.3.5 The specified sequence of the data must be observed when forming the CRC checksum.

5.1.4 Authentication

5.1.4.1 An authentication shall be performed to start an SL 4 or an SL 2 connection. The authentication is used to check that the connection has the required Safety Level (SL). This implies that computers/units of at least the required Safety Integrity Level (SIL) are being used at either end of the connection for data transmission. Random numbers modified by an algorithm known only to computers with the respective Safety Integrity Level are exchanged in separate telegrams (see 5.2.3 and 5.2.4).

5.1.4.2 Algorithm for authentication for SL 4 communication

5.1.4.2.1 The random number received during connection set-up is incremented by 1.

5.1.4.3 Algorithm for authentication for SL 2 communication

5.1.4.3.1 The random number received during connection set-up is incremented by 2.

5.1.4.4 Authentication is not performed for SL 0 connections.

5.2 Telegram Structure

5.2.1 Connect Request Telegram

5.2.1.1 Connection set-up shall be initiated by a Connect Request telegram.

5.2.1.2 A Connect Request Telegram shall be sent on request from upper layer (the Safe Time Layer).

5.2.1.3 Structure of the Connect Request Telegram, net data:

Byte	Designation	Range of values	Meaning
1	Lowest_order_byte_ sequence_number	0..255	Of the 32-bit sequence number only the 8 lowest-order bits are transferred (lowest byte). The 24 higher-order bits are used for CRC formation only. The sequence number of this telegram is the same as the random number field below.
2	Command	80h or 00h or C0h	Command for data transmission with error detection: Connect Request telegram
3..6	Random_number	00000000h.. FFFFFFFFh	A random number representing the sequence number for the telegram is created randomly and transmitted.
7..8	Idle_cycle_timeout	0..65500	Max. time between the reception of two telegrams that the receiver of this message has to check according to chapter 5.2.6 in ms: 0: no supervision of the idle_cycle_timeout 100..65500: idle_cycle_timeout in steps of 100 ms.
9	Configuration data prefix X	0..255	Value = 3
10	Configuration data prefix Y	0..255	Value = 0
11	Configuration data prefix Z	0..255	Value = 0
12	Dual bus	0..255	0 : The node is not equipped with dual busses 1 : The node is equipped with dual busses 2..255 : not used (reserved)
13.. n-l	net_data		Data from upper layers n <= 244 bytes (max. length of “DATA_UNIT”, see /2/) l: length of CRC checksum
n-l+1.. n	CRC	As per Safety Level	CRC as per Safety Level

5.2.1.3.1 The Safe Time Layer Configuration Data shall be transmitted as “net_data”.

5.2.1.4 After the Connect Request telegram has been sent, the Connect Confirm telegram is expected within the acknowledgement timeout period. If the acknowledgement fails to arrive in time, or if the acknowledgement is faulty, the connection shall be identified as having been erroneous and a disconnection shall be performed.

5.2.2 Connect Confirm Telegram

5.2.2.1 The connection set-up is acknowledged by the partner by means of a Connect Confirm telegram if the connection set-up has been accepted. Conditions for acceptance shall be:

- a connection does not already exist
- the connect request telegram is formally correct (no CRC error)

5.2.2.2 Structure of the Connect Confirm telegram, net data:

Byte	Designation	Range of values	Meaning
1	Lowest_order_byte_ sequence_number	0..255	Of the 32-bit sequence number only the 8 lowest-order bits are transmitted. The 24 higher-order bits are used for CRC formation only. The sequence number of this telegram is the same as the random number below.
2	Command	82h or 02h or C2h	Command for data transmission with error detection: Connect Confirm telegram
3..6	Random_number	00000000h.. FFFFFFFFh	A random number representing the sequence number of the telegram is created randomly and transferred.
7..8	Idle_cycle_timeout	0..65500	Max. time between the reception of two telegrams that the receiver of this message has to check according to chapter 5.2.6 in ms: 0: no supervision of the idle_cycle_timeout 100..65500: idle_cycle_timeout in steps of 100 ms.
9	Configuration data prefix X	0..255	Value = 3
10	Configuration data prefix Y	0..255	Value = 0
11	Configuration data prefix Z	0..255	Value = 0
12	Dual bus	0..255	0 : The node is not equipped with dual busses 1 : The node is equipped with dual busses 2..255 : not used (reserved)
13.. n-l	Net_data		Data from upper layer n <= 244 bytes (max. length of “DATA_UNIT”, see /2/) l: length of CRC checksum
n-l+1.. n	CRC		CRC as per Safety Level

5.2.2.2.1 The Safe Time Layer Configuration Data shall be transmitted as “net_data”.

5.2.2.3 After the Connect Confirm telegram has been sent, the Authentication telegram is expected for SL 4 and SL 2 connections within a response timeout period (= acknowledgement timeout period). If the Authentication telegram does not arrive in time, or the authentication is faulty, a Disconnect telegram is issued (final disconnection).

5.2.3 Authentication Telegram

5.2.3.1 Introduction: Besides the connection establishment proper, successful connection set-up of SL 4 and SL 2 connections also requires authentication. The authentication is used to check that the connection has the required Safety Level (SL). This implies that the two nodes wishing to establish the connection are at least of the required Safety Integrity Level (SIL). To detect this, a random number is used as the initial sequence number in the Connect Request telegram as specified above. Secondly, the random number in the authentication received with the Connect Confirm telegram is processed using the algorithm specified in section 5.1.4. The processed random number (= authentication number) is exchanged with the Authentication telegram/Authentication Acknowledgement and checked at the receiving end.

5.2.3.2 Authentication is used to improve the distinction between communication of different safety levels.

5.2.3.3 The Authentication Telegram shall not be transmitted in case of an SL 0 connection.

5.2.3.4 Structure of the Authentication telegram, net data:

Byte	Designation	Range of values	Meaning
1	Lowest_order_byte_ sequence_number	0..255	Of the 32-bit sequence number only the 8 lowest-order bits are transferred. The 24 higher-order bits are used for CRC formation only. This sequence number is incremented by one with respect to the Connect Request telegram.
2	Command	83h or 03h	Command for data transmission with error detection: Authentication telegram
3..6	Authentication_num ber	00000000h.. FFFFFFFFh	The authentication number is the received random number after processing by the defined algorithm (see chapter 5.1.4).
7 .. 7+l	CRC		CRC as per Safety Level l: length of CRC checksum

5.2.3.5 After the Authentication telegram has been sent, the Authentication Acknowledgement is expected within an acknowledgement timeout period. If the Authentication Acknowledgement fails to arrive in time, or the Authentication Acknowledgement is faulty, a Disconnect telegram shall be issued (final disconnection).

5.2.4 Authentication Acknowledgement Telegram

5.2.4.1 Introduction: The Authentication Acknowledgement telegram is an Acknowledgement Telegram necessary to establish an SL 4 or an SL 2 connection. In the Authentication Acknowledgement the partner passes the received random number after modification by the defined algorithm.

5.2.4.2 On successful completion of the authentication procedure, data communication can begin.

5.2.4.3 The Authentication Acknowledgement Telegram shall not be transmitted in case of an SL 0 connection.

5.2.4.4 Structure of the Authentication Acknowledgement, net data:

Byte	Designation	Range of values	Meaning
1	Lowest_order_byte_ sequence_number	0..255	Of the 32-bit sequence number only the 8 lowest-order bits are transmitted. The 24 higher-order bits are used for CRC formation only. This sequence number is incremented by one with respect to the Connect Confirm telegram.
2	Command	84h or 04h	Command for data transmission with error detection: Authentication Acknowledgement
3..6	Authentication_numb er	00000000h.. FFFFFFFFh	The authentication number is the random number received after processing by the defined algorithm.
7.. 7+l	CRC		CRC as per Safety Level l: length of CRC checksum

5.2.5 Disconnect Telegram

5.2.5.1 A connection shall be actively closed

- after a disconnect request from an upper layer or
- when an error is detected in the Safe Link Layer.

5.2.5.2 A Disconnect telegram shall be transmitted from the node which actively closes the connection to inform the other node.

5.2.5.3 In the Disconnect telegram, information shall be passed indicating whether the disconnection is final or if the connection may be established again (non-final disconnection).

5.2.5.4 Following a non-final disconnection, the connection may be established again.

5.2.5.5 Following a final disconnection, the connection shall not be established again.

5.2.5.5.1 Note: After a final disconnection it is necessary to restart the respective node(s). The table 5.5.1.5 indicates that both alternatives Defect / Idle are legal implementations for "Final Disconnect Received", designer may choose either of them.

5.2.5.6 For an SL 0 connection all reasons for a disconnection detected by the Safe Link Layer shall lead to a non-final disconnection except:

- final disconnect request from the upper layer
- final disconnect received.

5.2.5.6.1 Note: The table 5.5.1.7 indicates that both alternatives Defect / Idle are legal implementations for the last bullet, designer may choose either of them.

5.2.5.7 Reasons for a non-final disconnection:

5.2.5.7.1 A request for non final disconnection is received from the upper layer ("disconnect request from application").

5.2.5.7.2 A time-out occurred during establishing the connection ("error during connection set-up"):

- Connect Confirm Telegram not received within the specified time (acknowledgement timeout).

5.2.5.7.3 A first telegram with a general error is received ("1st incorrect receive"):

- Incorrect sequence number
- CRC error
- Authentication Telegram received on Safety level 0 connection

5.2.5.7.4 No Idle or Data Telegram received within idle_cycle_timeout ("idle cycle time-out").

5.2.5.7.5 An unexpected telegram is received on an existing connection ("connection set-up for existing connection")

- A Connect Request Telegram is received on an existing connection.
- A Connect Confirm Telegram is received on an existing connection.
- An Authentication Telegram is received on an existing connection.
- An Authentication Acknowledgement Telegram is received on an existing connection.

5.2.5.7.6 Reception of a Non-Final Disconnect telegram

5.2.5.7.7 Reception of a Final Disconnect telegram (depending on implementation)

5.2.5.8 Reasons for a final disconnection:

5.2.5.8.1 A request for final disconnection is received from the upper layer (“disconnect request from application”).

5.2.5.8.2 Intentionally deleted

5.2.5.8.3 An error was detected during Authentication (“authentication error”):

- Authentication Telegram or Authentication Acknowledgement Telegram not received within the specified time (acknowledgement timeout)
- Authentication number incorrect
- Unexpected telegram received during authentication

5.2.5.8.4 A second telegram with a general error is received within a defined period (“2nd incorrect receive” – see chapter 11 for calculated value):

- Incorrect sequence number
- CRC error

5.2.5.8.5 Reception of a Final Disconnect telegram (depending on implementation)

5.2.5.9 Structure of the Disconnect telegram, net data:

Byte	Designation	Range of values	Meaning
1	Lowest_order_byte_ sequence number	0..255	Of the 32-bit sequence number only the 8 lowest-order bits are transmitted. The 24 higher-order bits are used for CRC formation only.
2	Command	85h or 05h or C5h	Command for data transmission with error detection: Disconnect telegram
3	New_setup_desired	1 , 0	This indicates whether the disconnection is final (0) or a new connection may be established (1).
4	Disconnect_reason	0..FFh	Reason for disconnection: 00h: disconnect request from application 01h: reserved for future use 02h: error during connection set-up 03h: authentication error 04h: connection set-up for existing connection 05h: idle cycle time-out 06h: 1st incorrect receive 07h: 2nd incorrect receive 08h..1Eh: not used (reserved for future extension) 1Fh: Other (not in this list) reason for disconnect issued by Safe Link Layer 20h..40h: not used and reserved 41h..FFh: not used (for future extension)
5.. 5+m	Disconnect_reason_ text	Text	Vendor specific text, amend for diagnostics. m: 0..40; the length of the text in byte.
5+m+1 .. 5+m+l	CRC		CRC as per Safety Level l: length of CRC checksum

5.2.5.9.1 Note: The ‘not used and reserved’ values (20h .. 40h) are those allocated to the Safe Time Layer (see /4/).

5.2.6 Idle Telegram

5.2.6.1 Introduction: For data transmission with error detection, not only the connection set-up needs to be protected (authentication for SL 2 and SL 4) but also the connection itself. For this purpose, idle telegrams are exchanged when no data telegrams (application data) have been sent for a specific time.

5.2.6.2 Intentionally deleted

5.2.6.2.1 The monitoring of the reception of telegrams starts when the last telegram of the connection establishment procedure has been received (For SL 2 and SL 4 it is the Authentication Acknowledgement telegram for the connection master, and the Authentication telegram for connection slave. For SL 0 it is the Connect Request telegram for connection slave, and Connect Confirm telegram for connection master.).

5.2.6.2.2 The sending of idle telegrams starts after the last telegram of the connection establishment procedure has been sent (For SL 2 and SL 4 it is the Authentication Acknowledgement telegram for the connection slave, and the Authentication telegram for connection master. For SL 0 it is the Connect Request telegram for connection master, and Connect Confirm telegram for connection slave.).

5.2.6.2.3

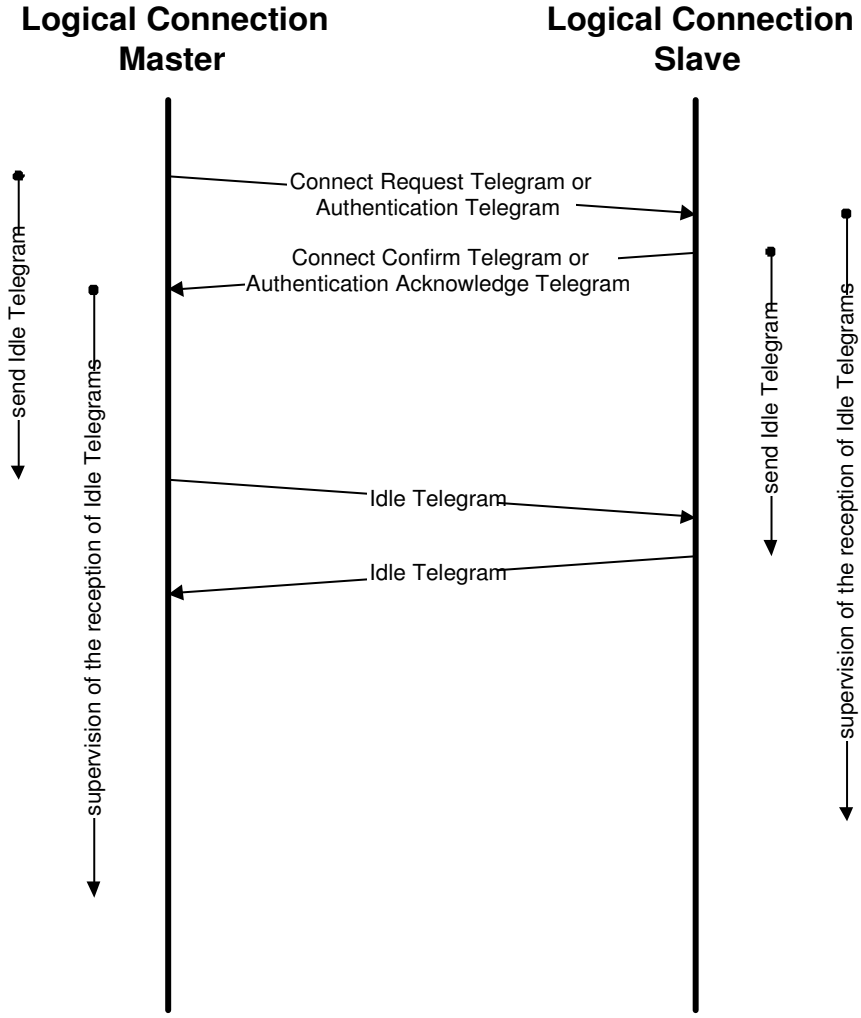


Figure 3 Supervision of Idle Telegrams

5.2.6.3 The reception of the idle/data telegram shall be monitored by the Connection Slave using the idle_cycle_timeout received in the Connect Request Telegram from the Connection Master. This period shall be reset each time an idle telegram or data telegram is received.

5.2.6.4 The reception of the idle/data telegram shall be monitored by the Connection Master using the idle_cycle_timeout received in the Connect Confirm Telegram from the Connection Slave. This period shall be reset each time an idle telegram or data telegram is received.

5.2.6.5 The sending interval of the idle telegram shall be timed to fulfil the idle_cycle_timeout at the receiving node. This period shall be reset each time an idle telegram or data telegram is sent.

5.2.6.6 Structure of the idle telegram, net data:

Byte	Designation	Range of values	Meaning
1	Lowest_order_byte_ sequence_number	0..255	Of the 32-bit sequence number only the 8 lowest-order bits are transmitted. The 24 higher-order bits are used for CRC formation only.
2	Command	86h or 06h or C6h	Command for data transmission with error detection: idle telegram
3 .. 3+l	CRC		CRC as per Safety Level l: length of CRC checksum

5.2.7 Data Telegram

5.2.7.1 The receiver does not acknowledge data telegrams on this protocol level.

5.2.7.1.1 Note: the Application-layer protocol may implement acknowledge.

5.2.7.1.2 Note: For point-to-point telegrams acknowledge is performed by the PROFIBUS FDL.

5.2.7.2 Structure of the data telegram, net data:

Byte	Designation	Range of values	Meaning
1	Lowest_order_byte_ sequence_number	00h..FFh	Of the 32-bit sequence number only the 8 lowest-order bits are transferred. The 24 higher-order bits are used for CRC formation only.
2	Command	89h or 09h or C9h, 20h .. 3Fh, A0h .. BFh and E0h .. FFh	Command for data transmission with error detection: Data telegram
3.. (n-l)	net_data		Application data n <= 244 bytes (max. length of “DATA_UNIT”, see /2/) l: length of CRC checksum
n-l+1.. n	CRC		CRC as per Safety Level

5.3 Telegram Sequences (SL 4 and SL 2 Point-to-Point Connections)

5.3.1 Establish a Connection

5.3.1.1 Before data can be transferred between two nodes, a connection must be established. The connection set-up shall be performed in two steps:

- connection set-up
- authentication

5.3.1.2 In the connection set-up, the sequence numbers to be used at the start of the data transfer shall be exchanged. In accordance with the CENELEC standard /1/, the initial sequence numbers shall be random numbers.

5.3.1.3 The authentication is used for SL 4 and SL 2 connections to check that the connection has the required Safety Level (SL). For this purpose, the received random numbers in each partner shall be modified using a defined algorithm (see section 5.1.4) and exchanged via the Authentication telegrams.

5.3.1.4 Following successful authentication the connection shall be considered as established and data exchange can begin. At the same time cyclical monitoring of the connection shall be initiated by means of additional idle telegram.

5.3.1.5 Implementation should at least make probable that the random numbers used for authentication on different connections are different.

5.3.1.6 In the diagrams of connection establishment below the following is assumed as an example:

5.3.1.6.1 Node 01 begins with sequence number 01234567h.

5.3.1.6.2 Node 02 begins with sequence number 89ABCDEFh.

5.3.1.6.3 Both nodes select 500 ms (= 1F4h) as the idle_cycle_timeout.

5.3.1.7 Example of the computation of the CRC of an SL 4 point-to-point connection using the implicit (marked ‘i’) and transmitted data (marked ‘t’):

	Authentication Telegram	Authentication Acknowledge Telegram
Data_length	0Ch (i)	0Ch (i)
Receive_address	01h (i)	08h (i)
Send_address	08h (i)	01h (i)
Partner_SAP	03h (i)	03h (i)
Local_SAP	03h (i)	03h (i)
Higher_order_24bits_of_sequence_number (low word, high byte)	45h (i)	CDh (i)
Higher_order_24bits_of_sequence_number (high word, low byte)	23h (i)	ABh (i)
Higher_order_24bits_of_sequence_number (high word, high byte)	01h (i)	89h (i)
Lowest_order_byte_sequence number	68h (t)	F0h (t)
Command	83h (t)	84h (t)
Authentication number (low word, low byte)	F0h (t)	68h (t)
Authentication number (low word, high byte)	CDh (t)	45h (t)
Authentication number (high word, low byte)	ABh (t)	23h (t)
Authentication number (high word, high byte)	89h (t)	01h (t)
CRC	20h (t)	A6h (t)
CRC	2Ah (t)	2Ah (t)
CRC	5Dh (t)	71h (t)
CRC	E9h (t)	97h (t)
CRC	BAh (t)	C5h (t)
CRC	6Dh (t)	85h (t)

5.3.1.8 Example of the computation of the CRC of an SL 2 point-to-point connection using the implicit (marked ‘i’) and transmitted data (marked ‘t’):

	Idle Telegram
Data_length	06h (i)
Receive_address	01h (i)
Send_address	08h (i)
Partner_SAP	09h (i)
Local_SAP	09h (i)
Higher_order_24bits_of_sequence_number (low word, high byte)	49h (i)
Higher_order_24bits_of_sequence_number (high word, low byte)	34h (i)
Higher_order_24bits_of_sequence_number (high word, high byte)	60h (i)
Lowest_order_byte_sequence number	31h (t)
Command	06h (t)
CRC	81h (t)
CRC	6Eh (t)
CRC	5Eh (t)
CRC	07h (t)

5.3.1.9 Example for a telegram sequence to establish an SL 4 connection:

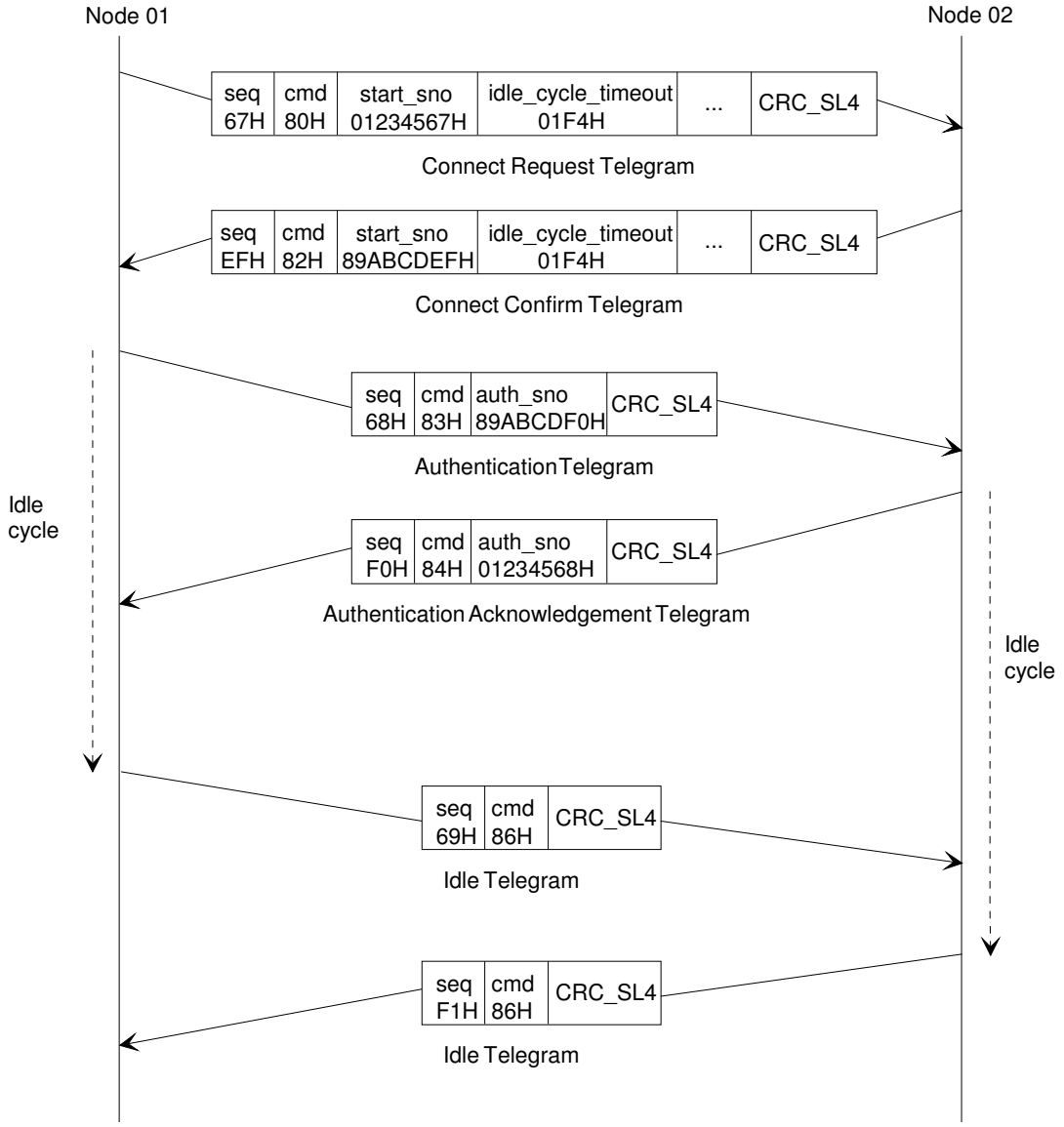


Figure 4 Telegram sequence to establish an SL 4 connection (connection set-up and authentication)

5.3.1.10 Example for a telegram sequence for a fault at establishing a connection (SL 4 connection):

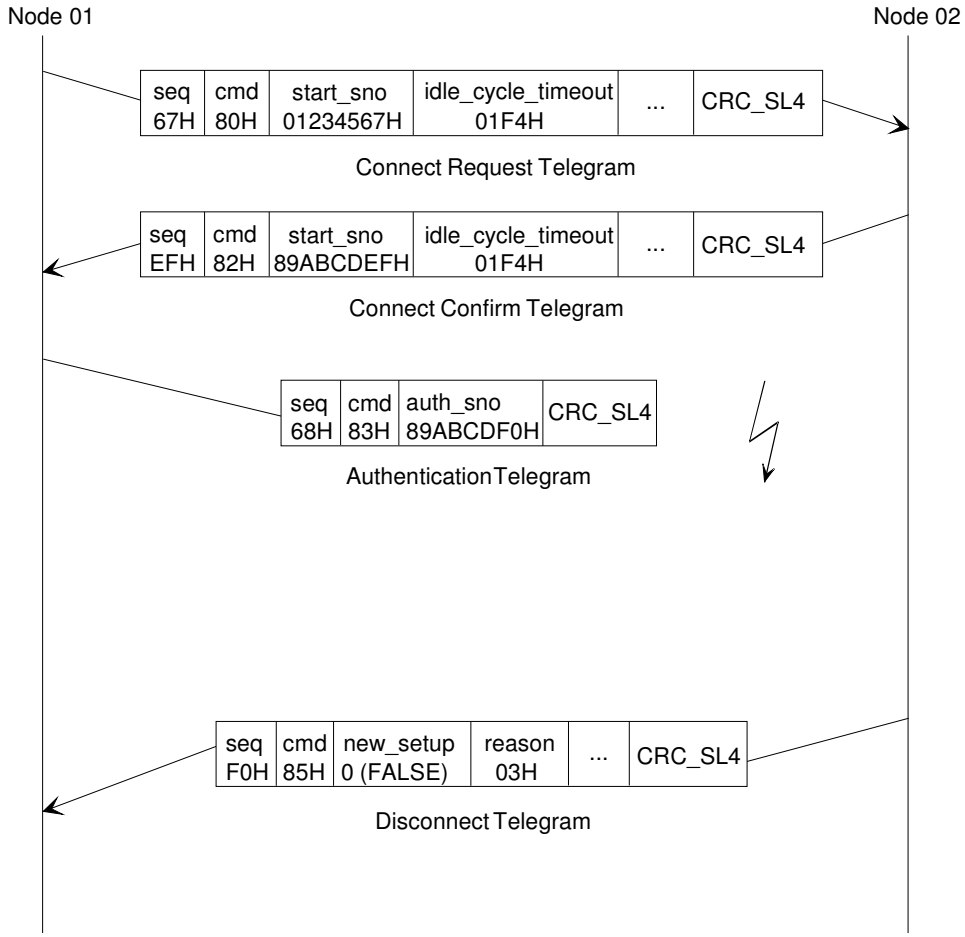


Figure 5 Telegram sequence of a faulty authentication due to an authentication telegram error

5.3.1.11 Example for a telegram sequence to establish an SL 2 connection:

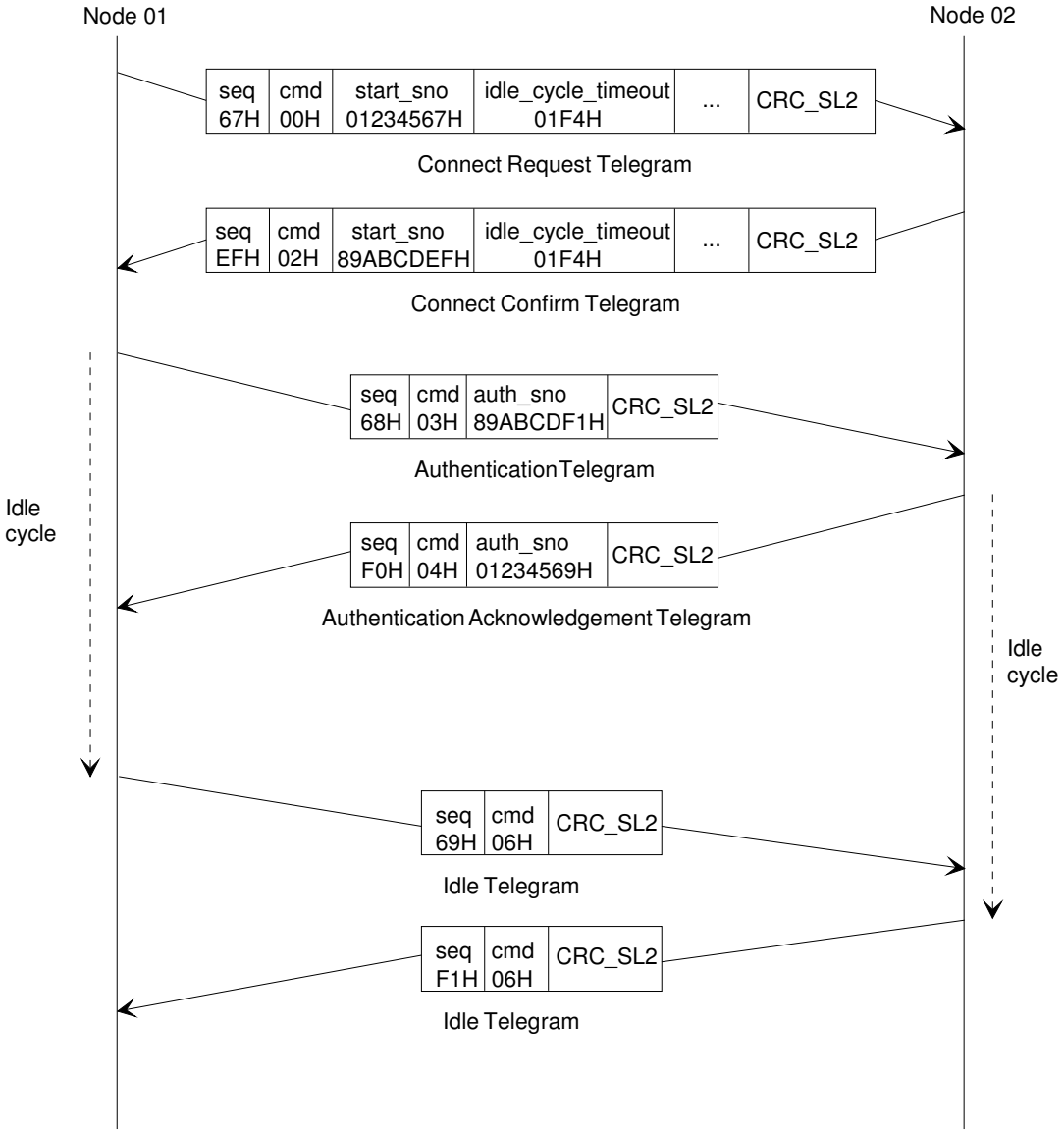


Figure 6 Telegram sequence to establish an SL 2 connection (connection set-up and authentication)

5.3.2 Data Exchange

5.3.2.1 Example for a telegram sequence of an SL 4 connection after the connection is established:

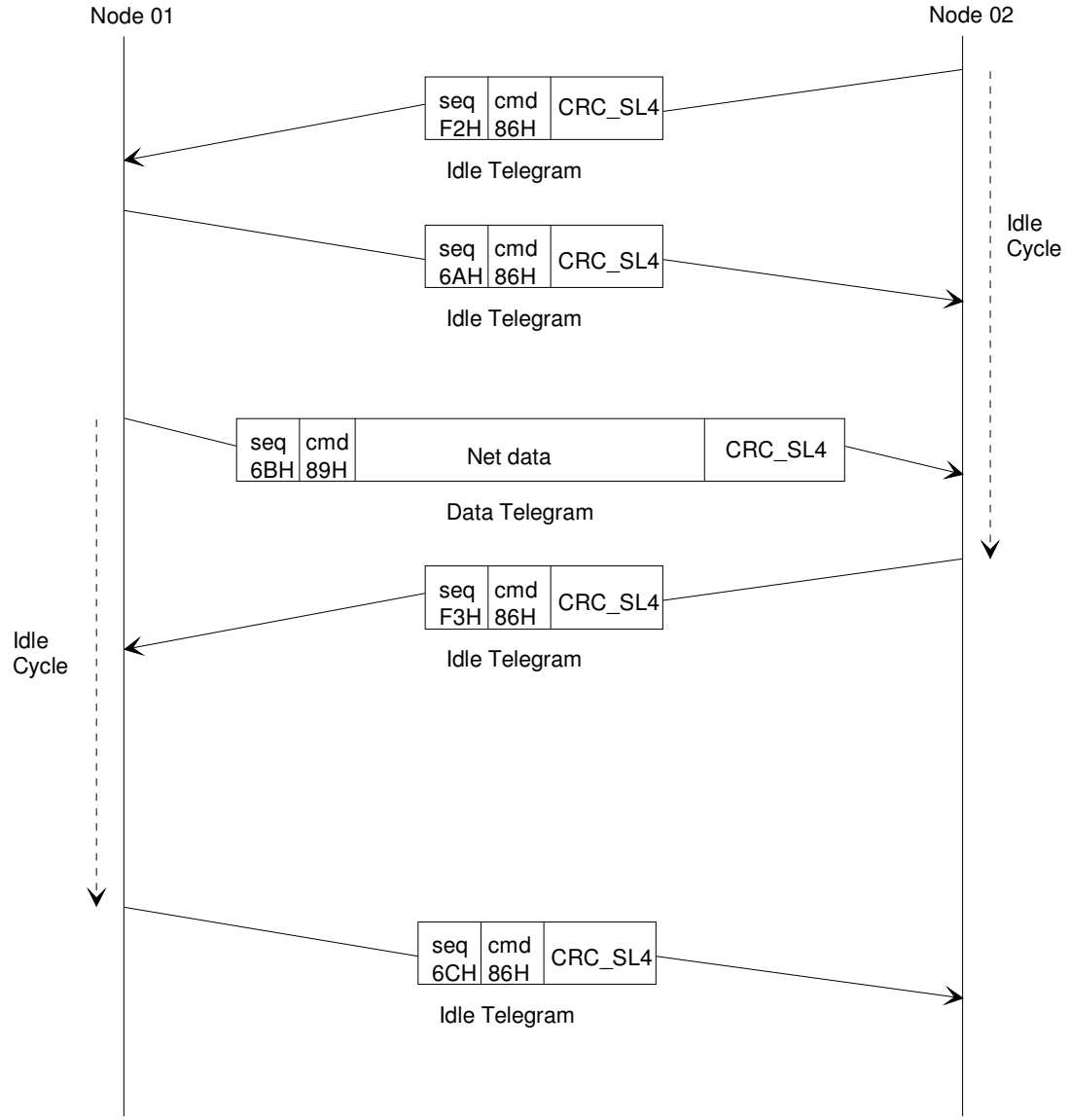


Figure 7 Telegram sequence for data exchange (SL 4 connection)

5.4 Telegram Sequences (SL 0 Point-to-Point connections)

5.4.1 Establish a Connection

5.4.1.1 Before data can be transferred between two nodes, a connection shall be established. This is done by performing the connection set-up.

5.4.1.2 During connection set-up, the sequence numbers to be used at the start of the data transfer shall be exchanged. The initial sequence numbers shall be random numbers.

5.4.1.3 Implementation should at least make probable that the random numbers for the sequence numbers on different connections are different.

5.4.1.4 Following successful connection set-up the connection shall be considered to be established and data exchange can begin. At the same time cyclical monitoring of the connection shall be initiated by means of additional idle telegram.

5.4.1.5 Example for a telegram sequence to establish an SL 0 connection:

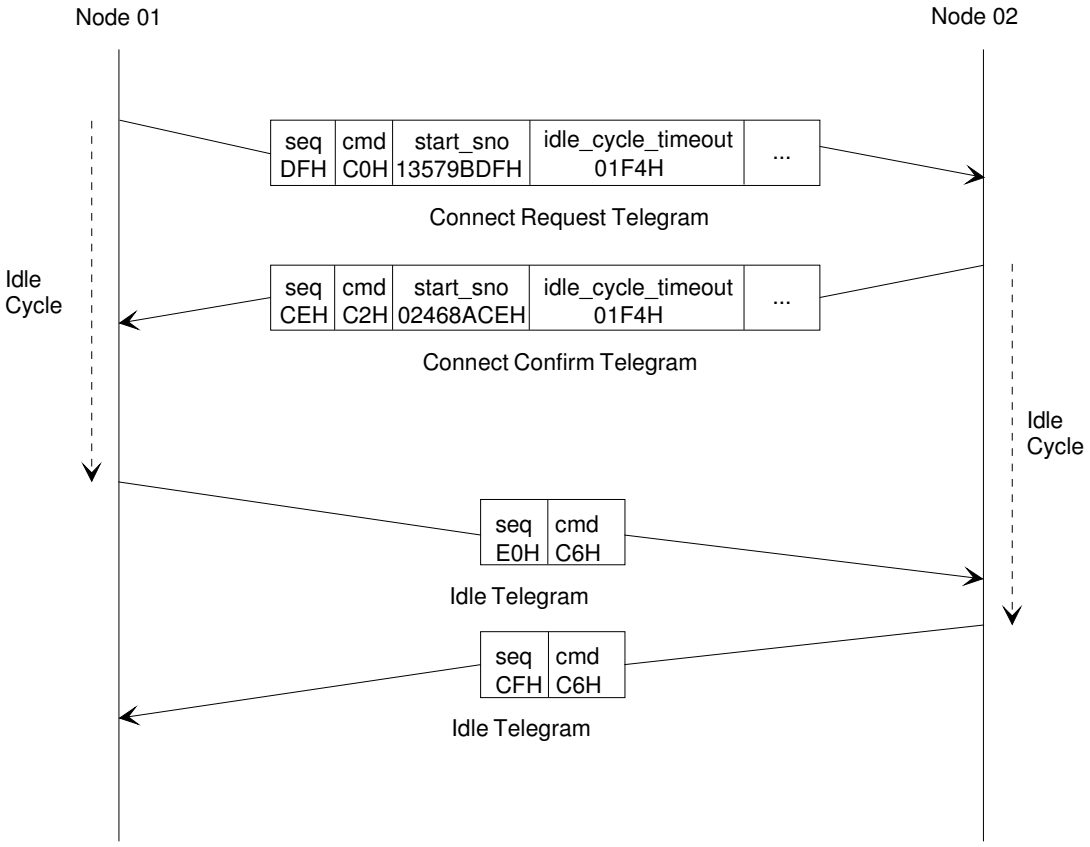


Figure 8 Telegram sequence to establish an SL 0 connection (connection set-up)

5.4.2 Data Exchange

5.4.2.1 Example for a telegram sequence of an SL 0 connection after the connection is established:

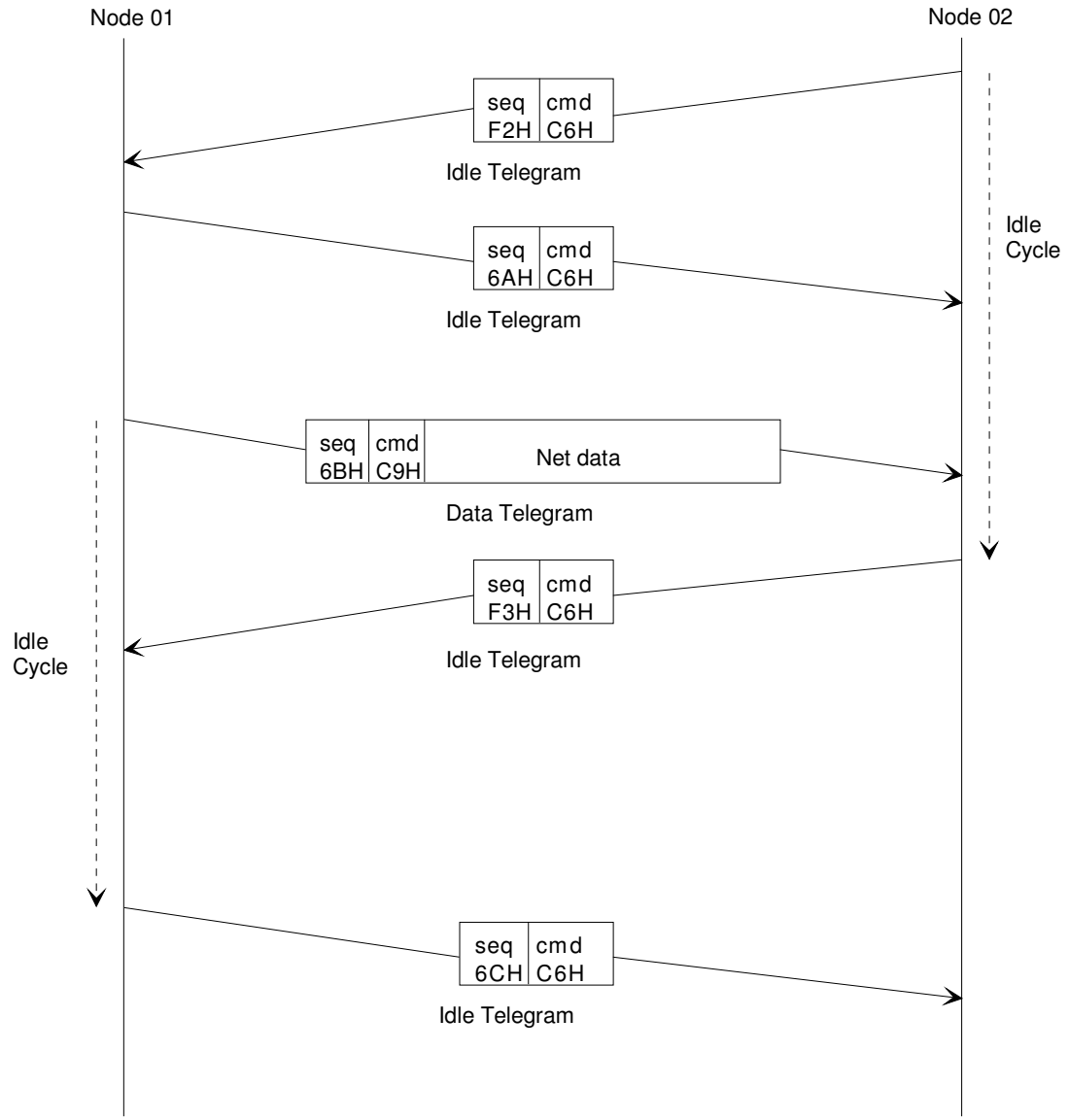


Figure 9 Telegram sequence for data exchange (SL 0 connection)

5.5 Transition Tables

5.5.1.1 The Safe Link Layer shall fulfil the transition tables defined in 5.5.1.5 for an SL 4 and an SL 2 connection and 5.5.1.7 for an SL 0 connection.

5.5.1.2 The table below defines the states:

State Name	Description
Idle	Wait for connection set-up (Connect Request Telegram (Slave) or Connect Request from application (Master))
wConAck	Wait for Connect Confirm Telegram (Only for Master)
wAuthReq	Wait for Authentication Request (Authentication Telegram) (Only for Slave)
wAuthAck	Wait for Authentication Acknowledge (Authentication Acknowledgement Telegram) (Only Master)
Data	"Application" data exchange phase
Defect	Connection is permanently closed

5.5.1.3 The table below defines the events:

Event	Description
Connect_snd	Connect request from "application"
Data_snd	Data from "application"
Timeout_snd	Expiration of Idle cycle interval timer
Timeout_rcv	Expiration of Idle cycle timeout timer
Timeout_brk	Expiration of acknowledgement timeout timer (Value see chapter 7.2)
Data_rcv	Data Telegram received
Idle_rcv	Idle Telegram received
Incorrect_rcv	Incorrect telegram received (single and multiple fault, see 5.5.1.3.1)
ConReq_rcv	Connect Request Telegram received
ConAck_rcv	Connect Confirm Telegram received
AuthReq_rcv	Correct Authentication Telegram received
AuthAck_rcv	Correct Authentication Acknowledge Telegram received
Discon_snd	Disconnect request from upper layer
FDiscon_snd	Final Disconnect request from upper layer
Discon_rcv	Non final Disconnect request received
FDiscon_rcv	Final Disconnect request received
Auth_err	Incorrect Authentication telegram received or timeout of Authentication telegram

5.5.1.3.1 Incorrect telegram received; as laid out in the specification there is a single and a double fault possible. A double fault occurs if a corrupted telegram occurs two times within a defined period (see chapter 5.2.5.8.4) in a connection with higher Safety Level. For Safety Level 0 connections, all incorrect receives are considered first incorrect receive.

5.5.1.4 The table below defines the actions (SL 4 and SL 2 connections):

ID	Operation
f_00	No operation
f_01	Send Connect Request Telegram & trigger acknowledgement timeout timer
f_02	Send Connect Confirm Telegram & trigger acknowledgement timeout timer
f_03	Send Authentication Telegram & trigger acknowledgement timeout timer & trigger idle cycle interval timer
f_04	Send Authentication Acknowledge Telegram & trigger idle cycle timeout timer (reception supervision)& trigger idle cycle interval timer (sending interval) & stop acknowledge timer
f_05	Send Data Telegram & trigger idle cycle interval timer (sending interval)
f_06	Send Idle Telegram & trigger idle cycle interval timer (sending interval)
f_07	Send Disconnect Telegram
f_08	Send final Disconnect Telegram
f_09	Pass data to ”application” & trigger idle cycle timeout timer (reception supervision)
f_10	Trigger idle cycle timeout timer (reception supervision)
f_11	Save current time & Send Disconnect Telegram
f_12	Trigger idle cycle timeout timer (reception supervision) & stop acknowledge timer
f_13	Save current time

5.5.1.5 The table below defines the transitions between the states (SL 4 and SL 2 connections):

Action / Next State		State: Idle	wConAck	wAuthReq	wAuthAck	Data	Defect
Event	Connect_snd	f_01	f_00	f_00	f_00	f_00	f_00
		wConAck	no change	no change	no change	no change	no change
	Data_snd	f_00	f_00	f_00	f_00	f_05	f_00
		no change	no change	no change	no change	no change	no change
	Timeout_snd	f_00	f_00	f_00	f_06	f_06	f_00
		no change	no change	no change	no change	no change	no change
	Timeout_rcv	f_00	f_00	f_00	f_00	f_07	f_00
		no change	no change	no change	no change	Idle	no change
	Timeout_brk	f_00	f_07	f_08	f_08	f_07	f_00
		no change	Idle	Defect	Defect	Idle	no change
	Data_rcv	f_00	f_00	f_08	f_08	f_09	f_00
		no change	no change	Defect	Defect	no change	no change
	Idle_rcv	f_00	f_00	f_08	f_08	f_10	f_00
		no change	no change	Defect	Defect	no change	no change
	1st Incorrect_rcv	f_13	f_11	f_11	f_11	f_11	f_00
		no change	Idle	Idle	Idle	Idle	no change
	2nd Incorrect_rcv	f_00	f_08	f_08	f_08	f_08	f_00
		Defect	Defect	Defect	Defect	Defect	no change
	ConReq_rcv	f_02	f_00	f_08	f_08	f_07	f_00
		wAuthReq	no change	Defect	Defect	Idle	no change
	ConAck_rcv	f_00	f_03	f_08	f_08	f_07	f_00
		no change	wAuthAck	Defect	Defect	Idle	no change
	AuthReq_rcv	f_00	f_00	f_04	f_08	f_07	f_00
		no change	no change	Data	Defect	Idle	no change
	AuthAck_rcv	f_00	f_00	f_08	f_12	f_07	f_00
		no change	no change	Defect	Data	Idle	no change
	Discon_snd	f_00	f_07	f_07	f_07	f_07	f_00
		no change	Idle	Idle	Idle	Idle	no change
	FDiscon_snd	f_00	f_08	f_08	f_08	f_08	f_00
		no change	Defect	Defect	Defect	Defect	no change
	Discon_rcv	f_00	f_00	f_00	f_00	f_00	f_00
		no change	Idle	Idle	Idle	Idle	no change
	FDiscon_rcv	f_00 no change	f_00 Defect/Idle*	f_00 Defect/Idle*	f_00 Defect/Idle*	f_00 Defect/Idle*	f_00 no change
	Auth_err	f_00 no change	f_00 no change	f_08 Defect	f_08 Defect	f_07 Idle	f_00 no change

Note: * in the table indicates that both alternatives Defect / Idle are legal implementations, designer may choose either of them. For FDiscon_rcv, it is reasonable to select defect if the STM receives the final disconnect, and idle if the ETCS onboard function receives the final disconnect. The ETCS kernel may run without the STM, but not vice versa.

5.5.1.6 The table below defines the actions (SL 0 connections):

ID	Operation
f_00	No operation
f_01	Send Connect Request Telegram & trigger acknowledgement timeout timer & trigger idle cycle interval timer (sending interval)
f_02	Send Connect Confirm Telegram & trigger idle cycle interval timer (sending interval) & trigger idle cycle timeout timer (reception supervision)
f_05	Send Data Telegram & trigger idle cycle interval timer (sending interval)
f_06	Send Idle Telegram & trigger idle cycle interval timer (sending interval)
f_07	Send Disconnect Telegram
f_08	Send final Disconnect Telegram
f_09	Pass data to ”application” & trigger idle cycle timeout timer (reception supervision)
f_10	Trigger idle cycle timeout timer (reception supervision)
f_14	Stop acknowledge timer & trigger idle cycle timeout timer (reception supervision)

5.5.1.7 The table below defines the transitions between the states (SL 0 connections):

	Action / Next State	State			
		Idle	wConAck	Data	Defect
Event	Connect_snd	f_01	f_00	f_00	f_00
		wConAck	no change	no change	no change
	Data_snd	f_00	f_00	f_05	f_00
		no change	no change	no change	no change
	Timeout_snd	f_00	f_06	f_06	f_00
		no change	no change	no change	no change
	Timeout_rcv	f_00	f_00	f_07	f_00
		no change	no change	Idle	no change
	Timeout_brk	f_00	f_07	f_00	f_00
		no change	Idle	no change	no change
	Data_rcv	f_00	f_00	f_09	f_00
		no change	no change	no change	no change
	Idle_rcv	f_00	f_00	f_10	f_00
		no change	no change	no change	no change
	1st Incorrect_rcv	f_00	f_07	f_07	f_00
		no change	Idle	Idle	no change
	ConReq_rcv	f_02	f_00	f_07	f_00
		Data	no change	Idle	no change
	ConAck_rcv	f_00	f_14	f_07	f_00
		no change	Data	Idle	no change
	Discon_snd	f_00	f_07	f_07	f_00
		no change	Idle	Idle	no change
	FDiscon_snd	f_00	f_08	f_08	f_00
		no change	Defect	Defect	no change
	Discon_rcv	f_00	f_00	f_00	f_00
		no change	Idle	Idle	no change
	FDiscon_rcv	f_00	f_00	f_00	f_00
		no change	Defect/Idle*	Defect/Idle*	no change

Note: * in the table indicates that both alternatives Defect / Idle are legal implementations, designer may choose either of them. For FDiscon_rcv, it is reasonable to select defect if the STM receives the final disconnect, and idle if the

ETCS onboard function receives the final disconnect. The ETCS kernel may run without the STM, but not vice versa.

6. MULTICAST

6.1 General Aspects

6.1.1.1 Example of the data flow for a Multicast message:

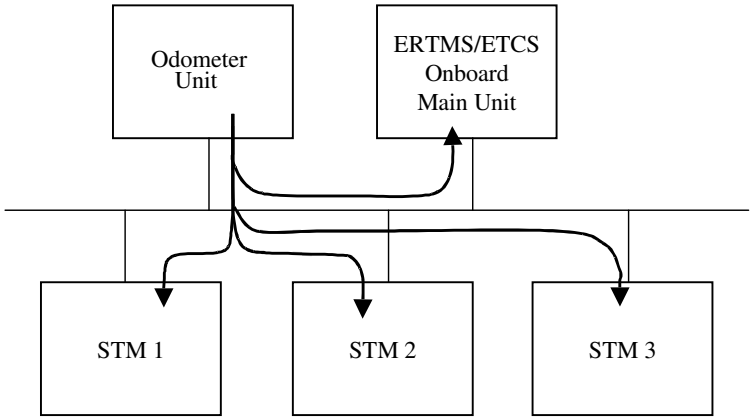


Figure 10 Example of the data flow for Multicast messages

6.1.1.1.1 Note: In this example the Odometer Node is separated from ERTMS/ETCS Onboard Main Node. This is only to clarify the two roles, but they may coincide in the same bus node.

6.1.1.2 Note: Safe multicast messages do not use Logical connection Masters nor Logical connection Slaves.

6.1.1.3 Only SIL 4 equipment is allowed to implement the procedures to send Multicast Telegrams.

6.1.1.3.1 Note: The receiver of Multicast Telegrams may be of any SIL (SIL 0 to SIL 4).

6.1.1.4 Intentionally deleted

6.1.1.5 Multicast is separated from other logical connections by a SAP of its own.

6.1.1.6 Since multicast is a 1-to-N transmission there is no possible data flow from receiver to sender on any level. As a consequence the receiver does not acknowledge any Multicast Data Telegram.

6.1.1.7 Multicast messages are allowed to be lost occasionally. The following message will provide an update, and the lost message is no longer needed. One consecutive multicast being lost or corrupted should be acceptable as a minimum. This shall be handled by the upper layer.

6.1.1.8 If the bus is realised redundantly all multicast messages shall be sent on both redundant busses.

6.1.1.9 As the sender is expected to transmit all Multicast Telegrams twice on each bus, the function to suppress copies is mandatory also for nodes with a single bus configuration.

6.2 Telegram Structure

6.2.1.1 Safe Multicast message transmission implements the following:

- Double transmission on each bus (time diversity)
- Cyclic transmission
- Sequence number
- CRC according to SL
- Implicit data

6.2.1.2 The sequence number is incremented by one with every transmitted telegram.

6.2.1.3 Sequence numbers with a value lower than or equal to a previously received one will not be forwarded to the upper layer.

6.2.1.4 Wrap around of the sequence number is not allowed. Wrap around of the sequence number shall be detected at the sender side. In this case the sender shall stop to send the Multicast messages.

6.2.1.4.1 Note: With a transmission cycle of 50 ms the time to a wrap around will be 6.8 years.

6.2.1.5 Format of Implicit Data

Byte	Designation	Range of values	Meaning
1	Data_length	14..244	The length of the user data bytes (“DATA_UNIT” from the PROFIBUS FDL viewpoint) in the telegram, i.e. length of telegram header for data transmission with error detection + length of net data + length of telegram trailer for data transmission with error detection
2	Receiver_address	127	The receiver's address can be processed as implicit data because it is available on PROFIBUS FDL Level.
3	Sender_address	0..126	The sender's address can be processed as implicit data because it is available on PROFIBUS FDL Level.
4	DSAP	0..62	Destination SAP (DSAP) is equal to SSAP, see /2/.
5	SSAP	0..62	Source SAP (SSAP).

6.2.1.6 Example of the computation of the CRC using the implicit (marked ‘i’) and transmitted data (marked ‘t’):

	Multicast Data Telegram
Data_length	0Eh (i)
Receiver_address	7Fh (i)
Sender_address	53h (i)
DSAP	21h (i)
SSAP	21h (i)
Configuration data prefix X	03h (t)
Configuration data prefix Y	00h (t)
Configuration data prefix Z	00h (t)
Command	8Dh (t)
Sequence_Number (low word, low byte)	EFh (t)
Sequence_Number (low word, high byte)	CDh (t)
Sequence_Number (high word, low byte)	ABh (t)
Sequence_Number (high word, high byte)	89h (t)
CRC	97h (t)
CRC	E4h (t)
CRC	C3h (t)
CRC	C1h (t)
CRC	6Ah (t)
CRC	30h (t)

6.2.1.7 Structure of the Multicast Telegram, net data:

Byte	Designation	Range of values	Meaning
1	Configuration data prefix X	0..255	Value = 3
2	Configuration data prefix Y	0..255	Value = 0
3	Configuration data prefix Z	0..255	Value = 0
4	Command	8Dh, A0h-BFh	These commands have been reserved in the Point-to-Point connections.
5..8	Sequence Number	00000000h .. FFFFFFFFh	Full sequence number must be transmitted since no multicast start-up procedure is possible. The start value shall be 00000000h.
9… (n-l)	Net data	0..255	Application data n <= 244 bytes (max. length of “DATA_UNIT”, see /2/) l: length of CRC checksum
n-l+1.. n	CRC		CRC for Multicast messages

6.2.1.8 Telegrams with command numbers reserved for future extension shall be ignored by the receiver.

7. LIST OF CONSTANTS

7.1.1.1 This chapter gives a list of the constants used within the Safe Link Layer together with their definition.

7.2 Acknowledgement timeout period

7.2.1.1 See chapter 5.2.1.4 and 5.2.5.7.2: Best estimated value 5 seconds.

8. INTENTIONALLY DELETED

9. REDUNDANCY SUPERVISOR

9.1.1.1 The redundancy supervisor is a non-safe function.

9.1.1.2 The redundancy supervisor is required if the node is equipped with dual busses.

9.1.1.3 The redundancy supervisor is not required if the node is equipped with one bus.

9.2 Function at sending a telegram

9.2.1.1 The redundancy supervisor shall send a telegram received from the Safe Link Layer on both busses.

9.2.1.2 Note: Exactly the same telegram is sent on both busses.

9.3 Function at receiving a telegram

9.3.1.1 The redundancy supervisor shall collect all telegrams received from both busses. All copies except one of a telegram shall be discarded. Only one copy of the telegram shall be sent to the Safe Link Layer.

9.3.1.2 Note: It is enough to look at the SAP and sequence_number of a telegram and compare them with previous telegrams to identify if a received telegram is a copy or not of a previous received one.

10. APPENDIX A: CRC GENERATOR POLYNOMIAL AND APPLICATION RULES

10.1 General

10.1.1.1 The communication may occur between nodes of different Safety Levels.

10.1.1.2 For point-to-point connections the communication shall achieve a safe communication at the level of the node supporting the lowest Safety Level (see 3.3.1.4).

10.1.1.2.1 Example: Communication between an SIL 4 and an SIL 2 equipment shall use the SL 2 communication.

10.1.1.3 For multicast connections the communication shall achieve a safe communication between the sender (SIL 4, see 6.1.1.3) and receivers with different safety levels.

10.1.1.4 Two different polynomials are defined for SL 2 and SL 4 respectively.

10.1.1.5 For SL 4 point-to-point communication the CRC_SL4 polynomial shall be used.

10.1.1.6 For SL 2 point-to-point communication the CRC_SL2 polynomial shall be used.

10.1.1.7 For SL 0 point-to-point communication no CRC shall be used.

10.1.1.8 For multicast communication the CRC_SL4 shall be used by the sender.

10.1.1.9 For multicast communication the receivers shall use the CRC listed in 10.1.1.5 and 10.1.1.6 according to their safety level.

10.1.1.10 Before calculating a CRC the initial value of all bits of the register shall be set to 0.

10.1.1.11 The checksum of a message shall not be used to determine the Safety Level of the message.

10.1.1.12 The CRC_SL2 polynomial is a factor in the CRC_SL4 polynomial. Therefore a message generated with CRC_SL4 polynomial can be checked with the CRC_SL2 polynomial. The achieved error detection capability corresponds to that of CRC_SL2.

10.1.1.13 This factorisation between the polynomials of different Safety Levels allows any SL 4 Multicast message to be verified by nodes supporting SL 2 and SL 4. As no CRC verification is performed by SL 0 nodes, the Multicast message can also be received by SL 0 equipments.

10.1.1.13.1 Note: There is a non-zero probability that the end of a message match the checksum of a higher Safety Level. This hazard has a low probability, but is mitigated by additional protection mechanism of the Safe Link Layer.

10.1.2 SL 2 Communication

10.1.2.1 The generator polynomial is:

CRC_SL2

$= x^{32} + x^{30} + x^{27} + x^{25} + x^{22} + x^{20} + x^{13} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{6} + x^{5} + x^{4} + 1$

10.1.2.1.1 Intentionally deleted

10.1.3 SL 4 Communication

10.1.3.1 For the Safety Level 4 the defined polynomial comes from the multiplication of the polynomial CRC_SL2 and an additional polynomial.

10.1.3.2 The generator polynomial is:

CRC_SL4

$= x^{48} + x^{47} + x^{46} + x^{44} + x^{41} + x^{39} + x^{35} + x^{34} + x^{32} + x^{31} + x^{29} + x^{28} + x^{25} + x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{17} + x^{14} + x^{11} + x^{9} + x^{7} + x^{5} + x^{3} + x^{2} + 1$

$= (x^{16} + x^{15} + x^{13} + x^{12} + x^{4} + x^{3} + x^{2} + 1) * \text{CRC\_SL2}$

$= (x^{16} + x^{15} + x^{13} + x^{12} + x^{4} + x^{3} + x^{2} + 1) * (x^{32} + x^{30} + x^{27} + x^{25} + x^{22} + x^{20} + x^{13} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{6} + x^{5} + x^{4} + 1)$

10.2 Checksum validation (non-normative)

10.2.1.1 The checksums are to be validated as for normal cyclic checksums, also when validating a message generated with a higher Safety Level polynomial. At least two principles are applicable.

10.2.1.2 One method is to divide the whole message including checksum by the generator polynomial and verify that the remainder is zero.

10.2.1.2.1 Example 1: Sender: SIL 4 / Receiver: SIL 4 (based on the example in chapter 6.2.1.6)

Telegram: 03h 00h 00h 8Dh EFh CDh ABh 89h 97h E4h C3h C1h 6Ah 30h

Input *): *0Eh 7Fh 53h 21h 21h* 03h 00h 00h 8Dh EFh CDh ABh 89h 97h E4h C3h C1h 6Ah 30h
Polynomial: D2h 8Dh B3h FAh 4Ah ADh
Result: 00h 00h 00h 00h 00h 00h

*) The input for the checksum validation is the received telegram plus the *implicit data*.

10.2.1.2.2 Example 2: Sender: SIL 4 / Receiver: SIL 2 (based on the example in chapter 6.2.1.6)

Telegram: 03h 00h 00h 8Dh EFh CDh ABh 89h 97h E4h C3h C1h 6Ah 30h

Input *): *0Eh 7Fh 53h 21h 21h* 03h 00h 00h 8Dh EFh CDh ABh 89h 97h E4h C3h C1h 6Ah 30h
Polynomial: 4Ah 50h 3Dh F1h
Result: 00h 00h 00h 00h

*) The input for the checksum validation is the received telegram plus the *implicit data*.

10.2.1.3 The other method compares the message checksum with the computed checksum, and they shall be equal for a valid message. Remove the last bits of the message corresponding to the length of the polynomial used for verification. Divide the message (except the removed last bits) with the verification polynomial and compare with the removed bits. They shall be equal.

10.2.1.3.1 Note: In the above paragraph, the message is considered to include its checksum, so the removed part is (part of) checksum but not application data.

10.2.1.3.2 Example 1: Sender: SIL 4 / Receiver: SIL 4 (based on the example in chapter 6.2.1.6)

Telegram: 03h 00h 00h 8Dh EFh CDh ABh 89h 97h E4h C3h C1h 6Ah 30h

Input *): *0Eh 7Fh 53h 21h 21h* 03h 00h 00h 8Dh EFh CDh ABh 89h
Polynomial: D2h 8Dh B3h FAh 4Ah ADh
Result: 97h E4h C3h C1h 6Ah 30h

*) The input for the checksum validation is the received telegram plus the *implicit data*, but the last bits are removed (in this case, because of the 48 bit polynomial: 48).

10.2.1.3.3 Example 2: Sender: SIL 4 / Receiver: SIL 2 (based on the example in chapter 6.2.1.6)

Telegram: 03h 00h 00h 8Dh EFh CDh ABh 89h 97h E4h C3h C1h 6Ah 30h

Input *): *0Eh 7Fh 53h 21h 21h* 03h 00h 00h 8Dh EFh CDh ABh 89h 97h E4h
Polynomial: 4Ah 50h 3Dh F1h
Result: C3h C1h 6Ah 30h

*) The input for the checksum validation is the received telegram plus the *implicit data*, but the last bits are removed (in this case, because of the 32 bit polynomial: 32).

11. APPENDIX B: CALCULATIONS FOR THE SAFETY CODES

11.1.1.1 This appendix gives the calculation of preconditions for required safety level of the CRC for the STM bus according to the CENELEC norm for Safety-related communication in transmission systems (/1/, Annex C.4):

11.1.1.1.1 A safety code CRC_SL4 with 48 bits is sufficient for a SIL 4 connection, if the conditions listed in chapter 11.3.5 are met.

11.1.1.1.2 A safety code CRC_SL2 with 32 bits is sufficient for a SIL 2 connection, if the conditions listed in chapter 11.4.5 are met.

11.2 Safety target for the transmission (CENELEC EN 50129)

11.2.1.1

Safety Integrity Level SIL	Tolerable Hazard Rate THR per hour and per function	Tolerable Hazard Rate THR per hour and per part of function *
4	$10^{-9} \leq$ THR $< 10^{-8}$	**$10^{-11} \leq$ THR $< 10^{-10}$**
2	$10^{-7} \leq$ THR $< 10^{-6}$	**$10^{-9} \leq$ THR $< 10^{-8}$**

* The transmission is only a part of a function. The safety target for this function is estimated to be 1% of the safety target for the function.

11.3 Calculation of preconditions for the CRC_SL4

11.3.1.1 **$R_{H1} + R_{H2} + R_{H3} \leq R_H$**

11.3.1.2 R_H: Tolerable failure rate of the complete transmission channel

11.3.1.2.1 => $R_H = 10^{-11}$ (from 11.2.1.1)

11.3.2 Hardware faults

11.3.2.1 **$R_{HW} \times p_{US} \times k_1 = R_{H1}$**

11.3.2.2 R_{H1}: Hazardous failure rate of the complete transmission channel due to hardware failures

11.3.2.3 p_{US}: Probability of undetected failure due to the performance of the safety code

11.3.2.3.1 From the analysis of CRC_SL4 published in /6/ based on Binary Symmetric Channel model, the worst case probability of undetected failure is 3.564×10^{-15} (for packet length 376bits). So we can assume:

11.3.2.3.2 $p_{US} = 3.564 \times 10^{-15}$

11.3.2.4 k_1: Factor for hardware faults including safety margin

11.3.2.4.1 Assumption(s):
- 2 consecutive corrupted messages (n = 2)
- safety factor 5 (m = 5)

11.3.2.4.2 => $k_1 \geq n \times m = 2 \times 5 = 10$

11.3.2.5 This leads to the computation of dependency of Hazardous failure rate R_{H1} from hardware faults of the non-trusted transmission system with failure rate R_{HW}:

$$R_{H1} = (R_{HW} \times p_{US} \times k_1) = (R_{HW} \times 3.563 \times 10^{-14})h^{-1}$$

11.3.3 EMI

11.3.3.1 $\mathbf{p_{US} \times p_{UT} \times f_W = R_{H2}}$

11.3.3.2 R_{H2}: Hazardous failure rate of the complete transmission channel due to EMI

11.3.3.3 p_{US}: Probability of undetected failure due to the performance of the safety code

11.3.3.3.1 As mentioned in 11.3.2.3.1 according to /6/

11.3.3.3.2 => $p_{US} = 3.564 \times 10^{-15}$

11.3.3.4 p_{UT}: Probability of undetected failure due to the performance of the transmission code is according to /5/

11.3.3.4.1 => $p_{UT} \leq 5 \times 10^{-6}$

11.3.3.4.2 Note:

It is not allowed to use the formula $p_{UT} = 2^{-b}$ as given in EN 50159 /1/ because the Profibus uses a combined code: horizontal parity and arithmetic sum (see EN 50170-2)!

An analysis of the probability of undetected failures states that assuming common bit error rates of less than ($p = 10^{-6}$) as well as bit error rate due to noise ($p = 0,5$) it is less than $5x10^{-6}$ (see /5/):

Figure 11 Probability of undetected failures

11.3.3.5 This leads to the computation of Hazardous failure rate R_{H2} as a function of frequency of wrong (corrupted) messages f_W:

$R_{H2} = (p_{US} \times p_{UT} \times f_w) = (3.564 \times 10^{-15} \times 5x10^{-6} \times f_w)\ h^{-1} = (1.782 \times 10^{-20} \times f_w)\ h^{-1}$

11.3.4 Transmission code faults

11.3.4.1 **$k_2 \times p_{us} \times 1/T = R_{H3}$**

11.3.4.2 R_{H3}: Hazardous failure rate of the complete transmission channel due to transmission code faults

11.3.4.3 As mentioned in 11.3.2.3.1 according to /6/:

11.3.4.3.1 => $p_{US} = 3.564x10^{-15}$

11.3.4.4 k_2: Factor which describes the percentage of hardware faults that result in undetected disabling of transmission decoding

11.3.4.4.1 Taking recommended value for k_2 from EN50159 /1/ => $k_2 = 10^{-4}$

11.3.4.5 This leads to the computation of Hazardous failure rate of transmission code checker as a function of Time span of a minimal acceptable time of receiving two consecutive corrupted messages:

$\mathbf{R_{H3}} = (k_2 \times p_{US} \times 1/T) = (10^{-4} \times 3.564 \times 10^{-15} \times 1/T) = (3.564 \times 10^{-19} \times 1/T)\ h^{-1}$

11.3.4.6 Value 1/T may be substituted by f_w (frequency of corrupted messages) since we expect in computation reception of 2 corrupted messages in a time span T and since the difference between the frequency of corrupted messages and the frequency of detected corrupted messages can be neglected, so

11.3.4.7 $\mathbf{R_{H3}} = (3.564 \times 10^{-19} \times f_w)\ h^{-1}$

11.3.5 Result

11.3.5.1 Using computed values from 11.3.2.5, 11.3.3.5, 11.3.4.7 together with inequality $\mathbf{R_{H1} + R_{H2} + R_{H3} \leq R_H}$, we can compute mathematical dependency between frequency of corrupted messages (time between two corrupted messages are received) and failure rate of the non-trusted transmission system:

11.3.5.2 $(R_{HW} \times 3.564 \times 10^{-14}) + (1.782 \times 10^{-20} \times f_w) + (3.564 \times 10^{-19} \times f_w) \leq 10^{-11}$
$\mathbf{f_w \leq (10^{-11} - 3.564 \times 10^{-14} \times R_{HW}) / (3.7422 \times 10^{-19})\ h^{-1}}$

11.3.5.3 The safe fall back state shall be entered when inequality 11.3.5.2 is not fullfilled.

11.3.5.4 Sample values for minimum period for 2nd incorrect receive according to 11.3.5.2

R_{HW}	Minimum period for 2nd incorrect receive [ms]
1.10^{-3}	0,13471968
1.10^{-5}	0,134719205
1.10^{-8}	0,1347192

11.4 Calculation of preconditions for the CRC_SL2

11.4.1.1 $\mathbf{R_{H1} + R_{H2} + R_{H3} \leq R_H}$

11.4.1.2 R_H: Tolerable failure rate of the complete transmission channel

11.4.1.2.1 => $R_H = 10^{-9}$ (from 11.2.1.1)

11.4.2 Hardware faults

11.4.2.1 $\mathbf{R_{HW} \times p_{US} \times k_1 = R_{H1}}$

11.4.2.2 R_{H1}: Hazardous failure rate of the complete transmission channel due to hardware failures

11.4.2.3 p_{US}: Probability of undetected failure due to the performance of the safety code

11.4.2.3.1 From the analysis of CRC_SL2 published in /6/ based on Binary Symmetric Channel model, the worst case probability of undetected failure is 1.32×10^{-8} (for packet length 72bits). So we can assume:

11.4.2.3.2 $p_{us} = 1.32 \times 10^{-8}$

11.4.2.4 This leads to the computation of dependency of Hazardous failure rate R_{H1} from hardware faults of the non-trusted transmission system (in this case the Profibus) R_{HW}:

$$R_{H1} = (R_{HW} \times p_{US} \times k_1) = (R_{HW} \times 1.32 \times 10^{-7})\ \mathbf{h^{-1}}$$

11.4.3 EMI

11.4.3.1 $\mathbf{p_{US} \times p_{UT} \times f_W = R_{H2}}$

11.4.3.2 R_{H2}: Hazardous failure rate of the complete transmission channel due to EMI

11.4.3.3 As mentioned in 11.4.2.3.1 according to /6/

11.4.3.3.1 => $p_{us} = 1.32 \times 10^{-8}$

11.4.3.4 This leads to the computation of Hazardous failure rate R_{H2} as a function of frequency of wrong (corrupted) messages f_W:

$$R_{H2} = (p_{US} \times p_{UT} \times f_w) = (1.32 \times 10^{-8} \times 5 \times 10^{-6} \times f_w)h^{-1} = (6.6 \times 10^{-14} \times f_w)h^{-1}$$

11.4.4 Transmission code faults

11.4.4.1 $\mathbf{k_2 \times p_{us} \times 1/T = R_{H3}}$

11.4.4.2 R_{H3}: Hazardous failure rate of the complete transmission channel due to transmission code faults

11.4.4.3 As mentioned in 11.4.2.3.1 according to /6/:

11.4.4.3.1 => $p_{us} = 1.32 \times 10^{-8}$

11.4.4.4 This leads to the computation of Hazardous failure rate of transmission code checker as a function of Time span of a minimal acceptable time of receiving two consecutive corrupted messages:

$$\mathbf{R_{H3} =} (k_2 \times p_{US} \times 1/T) = (10^{-4} \times 1.32 \times 10^{-8} \times 1/T) = (1.32 \times 10^{-12} \times 1/T)\ h^{-1}$$

11.4.4.5 Value 1/T may be substituted by f_w (frequency of corrupted messages) since we expect in computation reception of 2 corrupted messages in a time span T and since the difference between the frequency of corrupted messages and the frequency of detected corrupted messages can be neglected, so

11.4.4.6 $\mathbf{R_{H3} =}$ $(1.32 \times 10^{-12} \times f_w)\ h^{-1}$

11.4.5 Result

11.4.5.1 Using computed values from 11.4.2.4, 11.4.3.4, 11.4.4.6 together with inequality $\mathbf{R_{H1} + R_{H2} + R_{H3} \leq R_H}$, we can compute mathematical dependency between frequency of corrupted messages (time between two corrupted messages are received) and failure rate of the non-trusted transmission system:

11.4.5.2 $(R_{HW} \times 1.32 \times 10^{-7}) + (6.6 \times 10^{-14} \times f_w) + (1.32 \times 10^{-12} \times f_w) \leq 10^{-9}$
$\mathbf{f_w \leq (10^{-9} - 1.32 \times 10^{-7} \times R_{HW}) / (1.386 \times 10^{-12})\ h^{-1}}$

11.4.5.3 The safe fall back state shall be entered, when inequality 11.4.5.2 is not fullfilled.

11.4.5.4 Sample values for minimum period for 2nd incorrect receive according to 11.4.5.2

$\mathbf{R_{HW}}$	**Minimum period for 2nd incorrect receive [s]**
1.10^{-3}	5,748387097
1.10^{-5}	4,996194977
1.10^{-8}	4,989606586

12. APPENDIX C: SERVICES PROVIDED BY THE SAFE LINK LAYER

12.1.1.1 This appendix is informal and shall be read as an example.

12.1.1.2 The Safe Link Layer may provide the following services for upper layers or application:

Service	Description	Parameters	Return
Connect	Initialises a connection	Multicast Sender, Multicast receiver, Point-to-point master or Point-to-point slave; Safety Level of the connection (SL 0, SL 2 or SL 4); Receive_address; Send_address; partner_SAP; Local_SAP; Idle_Cycle_interval; Idle_Cycle_timeout	Connection ID or Fail; The other node is using dual busses or Not
Send Data	Send net data	Connection ID; Command; Net data	Send time stamp or Fail
Get Data	Returns received net data	Connection ID	Net Data, No data or Fail; Reception time stamp
Disconnect	Disconnect a connection	Connection ID; Reason for disconnection; Type of disconnect; Reason for disconnection text	OK or Fail
Connection active	Ask if a connection is active	Connection ID	Connected, Connecting, Connection closed, Final Disconnected or Fail; Reason for Disconnection; Reason for disconnection text; The other node is using dual busses or Not

12.1.1.2.1 Point-to-point master indicates that the Connect Request Telegram is sent from this partner.

12.1.1.2.2 Point-to-point slave indicates that this partner receives the Connect Request Telegram.

12.1.1.2.3 The Idle Telegram is sent with the Idle_cycle_interval time.

12.1.1.2.4 The Safe Link Layer transmits the Idle_cycle_timeout value (higher value than the Idle_cycle_interval) in the Connect Request Telegram.