

# **IO-Link Communication**

## **Specification**

**Version 1.0  
January 2009**

**Order No: 10.002**

**Document identification: IOL-09-0001**  
**File name: IOL-Comm-Spec\_10002\_V10\_090118.doc**

Prepared, approved and released by the IO-Link Consortium

Any comments, proposals, requests on this document are appreciated. Please use [www.io-link-projects.com](http://www.io-link-projects.com) for your entries and provide name and email address.

Login: *IO-Link-V1*

Password: *Report*

**Important notes:**

NOTE 1 The IO-Link Consortium Rules shall be observed prior to the development and marketing of IO-Link products. The document can be downloaded from the [www.io-link.com](http://www.io-link.com) portal.

NOTE 2 Any IO-Link device shall provide an associated IODD file. Easy access to the file and potential updates shall be possible. It is the responsibility of the IO-Link device manufacturer to test the IODD file with the help of the IODD-Checker tool available per download from [www.io-link.com](http://www.io-link.com).

NOTE 3 Any IO-Link devices shall provide an associated manufacturer declaration on the conformity of the device with this specification, its related IODD, and test documents, available per download from [www.io-link.com](http://www.io-link.com).

**Disclaimer:**

The attention of adopters is directed to the possibility that compliance with or adoption of IO-Link Consortium specifications may require use of an invention covered by patent rights. The IO-Link Consortium shall not be responsible for identifying patents for which a license may be required by any IO-Link Consortium specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. IO-Link Consortium specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

The information contained in this document is subject to change without notice. The material in this document details an IO-Link Consortium specification in accordance with the license and notices set forth on this page. This document does not represent a commitment to implement any portion of this specification in any company's products.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, THE IO-LINK CONSORTIUM MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR PARTICULAR PURPOSE OR USE.

In no event shall the IO-Link Consortium be liable for errors contained herein or for indirect, incidental, special, consequential, reliance or cover damages, including loss of profits, revenue, data or use, incurred by any user or any third party. Compliance with this specification does not absolve manufacturers of IO-Link equipment, from the requirements of safety and regulatory agencies (TÜV, BIA, UL, CSA, etc.).

☪ **IO-Link** ® is registered trade mark. The use is restricted for members of the IO-Link Consortium. More detailed terms for the use can be found in the IO-Link Consortium Rules on [www.io-link.com](http://www.io-link.com).

**Conventions:**

In this specification the following key words (in **bold** text) will be used:

**may:** indicates flexibility of choice with no implied preference.

**should:** indicates flexibility of choice with a strongly preferred implementation.

**shall:** indicates a mandatory requirement. Designers **shall** implement such mandatory requirements to ensure interoperability and to claim conformity with this specification.

Publisher:

**IO-Link Consortium**

Haid-und-Neu-Str. 7

76131 Karlsruhe

Germany

Phone: +49 721 / 96 58 590

Fax: +49 721 / 96 58 589

E-mail: [info@io-link.com](mailto:info@io-link.com)

Web site: [www.io-link.com](http://www.io-link.com)

© No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

---

## Content

1	Management summary - Scope of this document.....	11
2	List of relevant patents .....	11
3	Related documents and references .....	11
3.1	References.....	11
3.2	Related documents.....	12
4	Definitions and abbreviations .....	12
4.1	Terms and definitions .....	12
4.1.1	Address .....	12
4.1.2	Startup .....	12
4.1.3	On-request data.....	12
4.1.4	Operating mode .....	12
4.1.5	Checksum/Checksum byte .....	13
4.1.6	CHKPDU .....	13
4.1.7	CMD .....	13
4.1.8	COM1 .....	13
4.1.9	COM2.....	13
4.1.10	COM3.....	13
4.1.11	Data channel .....	13
4.1.12	Data/Data bytes .....	13
4.1.13	Device .....	13
4.1.14	Direct parameters .....	13
4.1.15	DL command.....	13
4.1.16	DLL .....	13
4.1.17	Event.....	14
4.1.18	Fallback.....	14
4.1.19	Frame.....	14
4.1.20	Frame type .....	14
4.1.21	Framing error.....	14
4.1.22	Hot plug.....	14
4.1.23	Interleave .....	14
4.1.24	Master .....	14
4.1.25	Physic 1.....	14
4.1.26	Physic 2.....	14
4.1.27	Port .....	14
4.1.28	Process data.....	14
4.1.29	Process data cycle.....	14
4.1.30	Retry strategy .....	15
4.1.31	Switching signal.....	15
4.1.32	Service PDU .....	15
4.1.33	Standard IO (SIO).....	15
4.1.34	System Management.....	15
4.1.35	Telegram .....	15
4.1.36	UART format.....	15
4.1.37	Wake-up .....	15
4.1.38	Wake-up request.....	15
4.1.39	Cycle .....	15

---

4.2	Symbols and abbreviations .....	15
4.3	Conventions for service definitions .....	17
4.3.1	General .....	17
4.3.2	Service parameters .....	17
4.3.3	Service procedures .....	18
5	IO-Link overview .....	18
5.1	IO-Link topology .....	18
5.2	Specification outline .....	19
5.3	General characteristics .....	19
5.4	Compatibility .....	19
5.4.1	Master .....	20
5.4.2	Sensor .....	20
5.4.3	Actuator .....	20
6	Physical layer .....	20
6.1	General .....	20
6.1.1	Overview .....	20
6.1.2	Hierarchical composition .....	21
6.2	Transmitter/Receiver .....	21
6.2.1	Static parameters .....	21
6.2.2	Dynamic parameters .....	27
6.2.3	Power supply .....	31
6.2.4	Medium .....	32
6.3	EMC requirements .....	33
6.3.1	Operating conditions .....	34
6.3.2	Performance criteria .....	34
6.3.3	Required immunity tests .....	34
6.3.4	Required emission tests .....	35
6.3.5	Test configurations for IO-Link masters .....	35
6.3.6	Test configurations for IO-Link devices .....	37
7	Data link layer .....	38
7.1	Data link layer services .....	38
7.1.1	Services depending on station type .....	38
7.1.2	DL-ReadParam .....	39
7.1.3	DL-WriteParam .....	40
7.1.4	DL-SPduTransport .....	40
7.1.5	DL-PDOutputUpdate .....	41
7.1.6	DL-PDOutputTransport .....	42
7.1.7	DL-PDInputUpdate .....	42
7.1.8	DL-PDInputTransport .....	43
7.1.9	DL-PDCycle .....	43
7.1.10	DL-SetMode .....	44
7.1.11	DL-Mode .....	45
7.1.12	DL-Event .....	45
7.1.13	DL-Control .....	46
7.2	Data link layer protocol .....	46
7.2.1	Overview .....	46
7.2.2	Startup handler .....	49
7.2.3	Process data handler .....	60
7.2.4	On-request data handler .....	62

---

7.2.5	Frame handler .....	74
8	Application layer .....	86
8.1	Application layer service .....	86
8.1.1	Services depending on station type .....	86
8.1.2	Services for on-request data access .....	86
8.1.3	Services for process data access .....	89
8.1.4	Event .....	94
8.1.5	Control .....	95
8.2	Application layer protocol .....	95
8.2.1	Overview .....	95
8.2.2	The process data cycle .....	95
8.2.3	On-request data transfer .....	97
8.2.4	ALPDU-specific structure and encoding .....	100
9	System management .....	104
9.1	System management services .....	104
9.1.1	Services depending on station type .....	104
9.1.2	SetMode .....	105
9.1.3	GetMode .....	106
9.1.4	SM-Operate .....	107
9.2	System management protocol .....	108
9.2.1	Startup .....	108
9.3	Cycle time .....	111
9.3.1	Operation with preset cycle time .....	111
9.3.2	Free running .....	113
9.3.3	Frame type .....	113
10	Requirement for certification tests .....	114
	Annex A Error types .....	115
	Annex B Event codes .....	117
	Annex C Data object addressing by Service PDU Index .....	120
	Annex D Example Service PDU transmission .....	123
	Annex E Residual fault probability of the telegram checksum .....	124
	Annex F Supplements and errata .....	125
F.1	IODD device .....	125
F.1.1	Regulations for the application .....	125
F.2	Errata for Version 1.0 dated November 2007 .....	130
F.2.1	Clause 6.2.2.2 Table 4 .....	130
F.2.2	Clause 6.2.4.1 Pin2/5 .....	130
F.2.3	Clause 7.1.10 DL-SetMode .....	130
F.2.4	Clause 7.1.11 DL-Mode .....	130
F.2.5	Clause 7.2.4.1 Figure 41 (formerly 31) .....	130
F.2.6	Clause 7.2.4.3 Figures 42-44 (formerly 32-34) .....	130
F.2.7	Clause 7.2.4.3.1 Table 45 (formerly 43) .....	131
F.2.8	Clause 7.2.4.3.3 .....	131
F.2.9	Clause 7.2.4.3.4 Figure 47 (formerly 37) .....	131
F.2.10	Clause 7.2.4.4.2.5 Table 52 (formerly 50) .....	131
F.2.11	Clause 7.2.5.3 Figure 54 (formerly 44) .....	131
F.2.12	Clause 7.2.5.5.2 .....	131
F.2.13	Clause 7.2.5.5.6 .....	131

F.2.14	Clause 7.2.5.5.9.....	131
F.2.15	Clause 8.2.2 Figure 70 (formerly 60) .....	132
F.2.16	Clause 8.2.3.2 Figure 74 (formerly 63).....	132
F.2.17	Clause 8.2.4.1.3 Table 76 (formerly 74).....	132
F.2.18	Clause 8.2.4.2 Figure 80 (formerly 69).....	132
F.2.19	Clause 9.1.2 Target Mode, Real Mode.....	132
F.2.20	Clause 10 .....	132
F.2.21	Annex E Figure E.1 .....	132
Annex G	Layout for a manufacturer's declaration of conformity .....	133

## Figures

Figure 1	— IO-Link topology .....	18
Figure 2	— IO-Link layer structure.....	19
Figure 3	— PHY1.....	20
Figure 4	— PHY2.....	21
Figure 5	— Hierarchical composition .....	21
Figure 6	— Driver reference schematics .....	22
Figure 7	— Receiver reference schematics.....	22
Figure 8	— Master and device reference schematics for PHY2 .....	23
Figure 9	— Master and device reference schematics for PHY1 .....	23
Figure 10	— Voltage level definitions .....	24
Figure 11	— Switching thresholds .....	25
Figure 12	— UART character .....	27
Figure 13	— Eye diagram for "H" and "L" detection.....	28
Figure 14	— Eye diagram for safe detection of a UART frame.....	29
Figure 15	— Wake-up request.....	30
Figure 16	— Power-on timing .....	31
Figure 17	— Pin layout front view.....	32
Figure 18	— Reference schematic for effective line capacitance and loop resistance .....	33
Figure 19	— Test setup for electrostatic discharge .....	36
Figure 20	— Test setup for RF electromagnetic field.....	36
Figure 21	— Test setup for fast transients .....	36
Figure 22	— Test setup for RF common mode.....	37
Figure 23	— Test setup for electrostatic discharge (device) .....	37
Figure 24	— Test setup for RF electromagnetic field (device) .....	37
Figure 25	— Test setup for fast transients (device).....	38
Figure 26	— Test setup for RF common mode (device).....	38
Figure 27	— Structure and interfaces of the data link layer (master).....	47
Figure 28	— Structure and interfaces of the data link layer (device).....	47
Figure 29	— Master DL state diagram .....	50
Figure 30	— Successful establishment of communication .....	51
Figure 31	— Failed attempt to establish communication.....	52
Figure 32	— Communication establishment retries .....	52

Figure 33 — Device DL state diagram .....	54
Figure 34 — Min Cycle Time .....	57
Figure 35 — Frame Capability .....	57
Figure 36 — IO-Link Revision ID .....	58
Figure 37 — Process Data In .....	59
Figure 38 — Output data transport sequence.....	61
Figure 39 — Input Data Transport Sequence .....	61
Figure 40 — Interleave mode sequence .....	62
Figure 41 — On-request handler on the master .....	63
Figure 42 — Error-free Service PDU transport sequence .....	64
Figure 43 — Service PDU transport sequence with error on device .....	64
Figure 44 — Service PDU transport sequence on master.....	64
Figure 45 — Service handler on the master .....	65
Figure 46 — Service handler on the device .....	67
Figure 47 — Calculation of CHKPDU.....	69
Figure 48 — Event transmission without event details.....	70
Figure 49 — Event transmission with event details .....	70
Figure 50 — Meaning of Event_1 to Event_6 .....	72
Figure 51 — Event qualifier.....	72
Figure 52 — Frame handler startup .....	75
Figure 53 — Cyclic frame handler .....	75
Figure 54 — Frame structure .....	76
Figure 55 — Transmission data format .....	76
Figure 56 — DL Command byte .....	77
Figure 57 — CHECK/TYPE byte.....	77
Figure 58 — CHECK/STAT byte.....	78
Figure 59 — Frame type 0.....	80
Figure 60 — Frame type 1 (process data).....	80
Figure 61 — Frame type 1 (on-request data) .....	81
Figure 62 — Frame type 2.1.....	81
Figure 63 — Frame type 2.2.....	82
Figure 64 — Frame type 2.3.....	82
Figure 65 — Frame type 2.4.....	82
Figure 66 — Frame type 2.5.....	83
Figure 67 — Frame timing.....	84
Figure 68 — Cycle timing .....	85
Figure 69 — Structure and interfaces of the AL .....	95
Figure 70 — Output data transmission.....	96
Figure 71 — Input data transmission .....	97
Figure 72 — Master AL state machine.....	98
Figure 73 — Device AL state machine .....	99
Figure 74 — Event transfer .....	100
Figure 75 — Service PDU .....	100

Figure 76 — SERVICE byte .....	102
Figure 77 — Examples of index formats in Service PDUs.....	103
Figure 78 — Examples of Request Service PDUs .....	103
Figure 79 — Examples of Response Service PDUs.....	104
Figure 80 — Examples of Read and Write Service PDUs .....	104
Figure 81 — System management states.....	108
Figure 82 — System management start-up .....	109
Figure 83 — System management start-up for port 4 .....	110
Figure 84 — System management start-up master and device .....	111
Figure 85 — Cycle time management.....	112
Figure 86 — Example of port cycle operations.....	113
Figure C.1 — General mapping of data objects.....	120
Figure D.1 — Example for Service PDU transmission .....	123
Figure E.1 — Residual fault probability.....	124
Figure F.1 — Example data structure .....	126
Figure F.2 — Class B port definitions .....	130
Figure F.3 — Changes in figure 42.....	131
Figure G.1 — Layout proposal for a manufacturer's declaration .....	133

## Tables

Table 1 — Static parameters of receiver.....	24
Table 2 — Static master parameters .....	25
Table 3 — Static device parameters.....	26
Table 4 — Transmission parameters .....	29
Table 5 — Wake-up request definitions .....	31
Table 6 — Power-on definitions.....	31
Table 7 — Pin assignment .....	32
Table 8 — Cable parameters.....	33
Table 9 — Cable conductor assignment .....	33
Table 10 — Cycle times .....	34
Table 11 — EMC test levels .....	35
Table 12 — Assignment of services to master and device.....	38
Table 13 — DL-ReadParam .....	39
Table 14 — DL-WriteParam .....	40
Table 15 — DL-SPduTransport .....	40
Table 16 — DL-PDOutputUpdate.....	41
Table 17 — DL-PDOutputTransport.....	42
Table 18 — DL-PDInputUpdate .....	42
Table 19 — DL-PDInputTransport .....	43
Table 20 — DL-PDCycle .....	43
Table 21 — DL-SetMode.....	44
Table 22 — DL-Mode .....	45
Table 23 — DL-Event.....	45



Table 24 — DL-Control .....	46
Table 25 — AL service primitives sent to the master's DL .....	48
Table 26 — DL service primitives sent to the master's AL .....	48
Table 27 — SM service primitives sent to the master's DL .....	48
Table 28 — DL service primitives sent to the master's SM .....	49
Table 29 — Local variables on the master's DL .....	49
Table 30 — Master DL states .....	50
Table 31 — WakeUp procedure and level definitions .....	52
Table 32 — Startup sequence .....	53
Table 33 — StartUp parameters in InitMaster .....	53
Table 34 — StartUp parameters in InitDevice .....	54
Table 35 — Device DL states .....	54
Table 36 — Addresses on the parameter data channel .....	55
Table 37 — Values of Master Command .....	56
Table 38 — Values of Min Cycle Time .....	57
Table 39 — Values of SPDU .....	58
Table 40 — Values of Type1 .....	58
Table 41 — Values of PHY1 .....	58
Table 42 — Values of SIO .....	59
Table 43 — Values of BYTE .....	59
Table 44 — Values of Offset Time .....	60
Table 45 — Master DL states .....	65
Table 46 — Device DL states .....	67
Table 47 — Definition for FlowCTRL .....	68
Table 48 — Event .....	71
Table 49 — Values of INSTANCE .....	72
Table 50 — Values of TYPE .....	73
Table 51 — Values of MODE .....	73
Table 52 — Event buffer addresses .....	73
Table 53 — Control codes .....	74
Table 54 — Values of Data Channel .....	77
Table 55 — Values of R/W .....	77
Table 56 — Values of Frame type .....	78
Table 57 — Values of Event .....	78
Table 58 — Typical cycle times .....	84
Table 59 — Assignment of services to master and device .....	86
Table 60 — Read .....	86
Table 61 — Write .....	88
Table 62 — Abort .....	89
Table 63 — Reject .....	89
Table 64 — GetInput .....	90
Table 65 — NewInput .....	91
Table 66 — SetInput .....	91

Table 67 — PDCycle.....	92
Table 68 — GetOutput.....	92
Table 69 — NewOutput.....	93
Table 70 — SetOutput.....	93
Table 71 — Event.....	94
Table 72 — Control.....	95
Table 73 — Master AL states.....	98
Table 74 — Device AL states.....	99
Table 75 — Definition for Service.....	101
Table 76 — Definition for Length.....	101
Table 77 — Use of index formats.....	102
Table 78 — Service PDU syntax.....	103
Table 79 — Assignment of services to master and device.....	104
Table 80 — SetMode.....	105
Table 81 — Target modes.....	105
Table 82 — GetMode.....	107
Table 83 — SM-Operate.....	107
Table 84 — Use of frame types.....	113
Table A.1 — Error Codes.....	115
Table A.2 — Additional Codes for Error Code S_APP_DEV.....	116
Table B.1 — Event Codes.....	117
Table B.2 — Exemplary event data.....	118
Table C.1 — Index definition for Service PDU.....	120
Table C.2 — Values of System Command.....	121
Table F.1 — Definition for Length.....	132

## Revision Log

Identification	Version	Originator	Date	Change Note / History / Reason
TC3-07-0002a	1.00	Team Technology	16-Nov-07	Final release of IO-Link communication
IOL-09-0001	1.00	Team Technology	18-Jan-09	Formal adjustments, updated EMC requirements, errata and supplements and manufacturer's declaration form

## 1 Management summary - Scope of this document

IO-Link is a point-to-point digital communication technology developed by the IO-Link Consortium under the umbrella of PROFIBUS and PROFINET International. IO-Link mainly focuses on simple sensors and actuators in factory automation, which are using nowadays small and cost-effective microcontrollers. With the help of the IO-Link communication technology, the obstacles of traditional signal connection technologies such as switching 24V, analog 0...10V, etc. can be overcome. Classic sensors and actuators usually are connected to a fieldbus system via input/output modules in so-called remote I/O devices. The new IO-Link master function will enable these modules to map IO-Link devices on a fieldbus system or build up direct gateways. Thus parameter data and diagnosis data can be transferred from the controller level down to the sensor/actuator level by means of the IO-Link communication. This is a contribution to consistent parameter storage and maintenance support. IO-Link is compatible to traditional signal switching technology according to IEC 61131-2.

This specification describes the IO-Link communication interface. It contains the definitions of the physical layer, the data link layer and the application layer for IO-Link masters and IO-Link devices according to the ISO/OSI reference model.

This extended version contains the original specification dated November 2007 with only formal corrections, updated EMC requirements, and includes several new annexes, one for errata and minor supplements. It also includes a template for the manufacturer's conformity declaration.

## 2 List of relevant patents

There is no final result of a patent search, neither by the members of the IO-Link Consortium nor by externals. The following patents seem to be relevant and shall be taken into account:

DE 100 30 845 A1  
DE 100 54 288 A1  
DE 102005 014783  
DE 199 04 543 C2  
DE 102 33436 A1

## 3 Related documents and references

### 3.1 References

ISO/IEC 10731, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

ISO/IEC 7498-1, *Information technology – Open Systems Interconnection – Basic Reference Model – Part 1: The Basic Model*

IEC 60947-5-2, *Low-voltage switchgear and controlgear - Part 5-2: Control circuit devices and switching elements - Proximity switches*

IEC 61131-2, *Programmable controllers - Part 2: Equipment requirements and tests*

IEC 61158-2, *Digital data communications for measurement and control - Fieldbus for use in industrial control systems - Part 2: Physical layer specification and service definition*

IEC 61158-3, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 3: Data link layer service definition*

IEC 61158-4, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 4: Data link layer protocol specification*

IEC 61158-5, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 5: Application layer service definition*

IEC 61158-6, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 6: Application layer protocol specification*

IEC 61784-1, *Digital data communications for measurement and control – Part 1: Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems*

### **3.2 Related documents**

IEC 61000-4-2, *Electromagnetic compatibility (EMC)- Part 4-2: Testing and measurement techniques - Electrostatic discharge immunity test*

IEC 61000-4-3, *Electromagnetic compatibility (EMC) - Part 4-3 : Testing and measurement techniques - Radiated, radio-frequency, electromagnetic field immunity test*

IEC 61000-4-4, *Electromagnetic compatibility (EMC) - Part 4-4: Testing and measurement techniques - Electrical fast transient/burst immunity test*

IEC 61000-4-5, *Electromagnetic compatibility (EMC) - Part 4-5: Testing and measurement techniques - Surge immunity test*

IEC 61000-4-6, *Electromagnetic compatibility (EMC) - Part 4-6: Testing and measurement techniques - Immunity to conducted disturbances, induced by radio-frequency fields*

EN 55011, *Industrial, scientific and medical (ISM) radio-frequency Equipment - Electromagnetic disturbance characteristics - Limits and methods of measurement*

EN 55022, *Information technology equipment - Radio disturbance characteristics - Limits and methods of measurement*

## **4 Definitions and abbreviations**

### **4.1 Terms and definitions**

#### **4.1.1 Address**

Part of the DL command. Used to reference data within a data quality, i.e., within a data channel. See Chapter 7.2.5.3.1 for more information.

#### **4.1.2 Startup**

This is the phase during which communication is established between master and device. It includes the wake-up procedure and the reading of the communication parameters.

#### **4.1.3 On-request data**

Data, which are transmitted acyclically at the request of the application. These include parameter data and event data.

#### **4.1.4 Operating mode**

A distinction is drawn between “Standard IO” and “IO Link” modes.

#### 4.1.5 Checksum/Checksum byte

In addition to data integrity protection on the physical layer with the parity bit, data transfer is also secured on the data link layer with a checksum.

The device checksum byte also contains the event bit.

#### 4.1.6 CHKPDU

Data integrity protection within a Service PDU. It is generated by exclusively ORing the byte of a request or response.

#### 4.1.7 CMD

See "DL Command".

#### 4.1.8 COM1

Operating mode of the physical layer with a transmission rate of 4,800 bauds.

#### 4.1.9 COM2

Operating mode of the physical layer with a transmission rate of 38,400 bauds.

#### 4.1.10 COM3

Operating mode of the physical layer with a transmission rate of 230,400 bauds.

#### 4.1.11 Data channel

Data channel is part of the DL command. In the IO-Link specification, a variety of data qualities (process data, parameter data, on-request-data, and event messages) are made available via various channels. These channels exist independently and harmoniously.

#### 4.1.12 Data/Data bytes

The DLL user data in a telegram. The meaning of the data bytes is defined by the command.

#### 4.1.13 Device

Sensor or actuator supporting IO-Link. An IO-Link-compatible actuator only supports IO-Link mode. Where the IO-Link interface is concerned, the device is the passive communication peer.

#### 4.1.14 Direct parameters

On-request data quality. Direct parameters are transferred without acknowledgment on a dedicated data channel.

#### 4.1.15 DL command

The command is part of the command byte. See Chapter 7.2.5.3 for more information.

There are 8 different commands. The commands are used to gain read or write access to 4 defined data channels.

The command byte is the first byte in a frame. It comprises the command (read/write and data channel) and the address.

#### 4.1.16 DLL

Data link layer

The data link layer regulates the transfer of process data and on-request data via the IO-Link interface.

#### **4.1.17 Event**

On-request data quality. The transfer of events is initiated by setting the event flag in the device's CHK/STAT byte and takes place on a dedicated data channel. Events may be initiated in the application or in the communication protocol.

#### **4.1.18 Fallback**

Fallback is the transition from communication mode to Switching or Standard IO (SIO) mode.

#### **4.1.19 Frame**

A frame comprises a master telegram and the subsequent device telegram.

#### **4.1.20 Frame type**

A frame type represents a combination of master telegram and device telegram for the transport of a certain number of process data and on-request data in both directions.

#### **4.1.21 Framing error**

Error on the UART layer causing the UART format of the stop bit to become corrupted.

#### **4.1.22 Hot plug**

Disconnection/connection of a device from/to a port during active operation.

#### **4.1.23 Interleave**

Interleave is an operating mode for cyclic operation. In this mode, frames for process data are always transmitted alternately with frames for on-request data. This makes the mode particularly suitable for devices with more than 2 bytes of process data.

#### **4.1.24 Master**

A master is part of a functional unit (e.g., a hub). It is connected at one end to a higher-level bus system (e.g., Profibus) and at the other to 1 to n devices via the IO-Link interface. Where the IO-Link interface is concerned, the master is the active communication peer.

#### **4.1.25 Physic 1**

2-wire transfer, power and data together in the time division multiplex.

#### **4.1.26 Physic 2**

3-wire transfer, power and data separate.

#### **4.1.27 Port**

Physical manifestation of the communication-medium interface. A device has just one port, a master may have several ports. The ports on a master are counted consecutively starting at 0 or 1.

#### **4.1.28 Process data**

Data transferred with high priority and in a fixed schedule. Transfer starts automatically following startup (no need for a request from the application). Process data include measured variables and controlled variables.

#### **4.1.29 Process data cycle**

The process data cycle refers to the transfer of all process data from an individual device. This transfer operation takes place in a fixed schedule. Depending on the number of process data items from a device, a process data cycle may comprise one or a number of cycles.

#### 4.1.30 Retry strategy

The retry strategy is a procedure to be followed by the master if it receives an incorrect checksum, or no checksum at all, from a device (e.g., no device connected).

#### 4.1.31 Switching signal

The binary signal from an IO-Link device, see SIO.

#### 4.1.32 Service PDU

Protocol data unit for the transfer of on-request data with acknowledgment. The Service PDU is packaged in a number of data frames for transfer. It is sent as a dedicated data quality on a dedicated data channel.

#### 4.1.33 Standard IO (SIO)

On the master: On the port, the level at C/Q is used as a discrete switching signal.

On the device: Operating mode in which a device sets or receives a discrete level at C/Q.

#### 4.1.34 System Management

System management controls master and device startup. It also coordinates the individual PortDLs on a master with more than one port.

#### 4.1.35 Telegram

In respect of IO-Link communication, a telegram is the coherent, directed transfer of UART characters via the IO-Link interface (either "from master to device" or "from device to master").

#### 4.1.36 UART format

Each byte is transferred on the data line in the following character format:

1 start bit	8 data bits	1 parity bit	1 stop bit
	Bit0	Bit 7	

#### 4.1.37 Wake-up

Change of state of a device from Wait or SIO mode to communication mode. Includes the Wake-up request and the first successful frame following a Wake-up request.

#### 4.1.38 Wake-up request

Master drives C/Q line to inverted level for a defined time interval.

#### 4.1.39 Cycle

In a cycle, a frame is transmitted between the master and a connected device. Transmission is followed by idle time.

## 4.2 Symbols and abbreviations

Symbol	Definition	Unit
AL	Application Layer	–
DL	Data Link Layer	–
SM	System Management	–
L+	Power supply connection	–
L-	Ground connection	–

<b>C/Q</b>	Connection for communication (C) or switching (Q) signal	–
<b>V+</b>	Voltage at L+	–
<b>V0</b>	Voltage at L-	–
<b>PS</b>	Power supply	V
<b>ΔPS</b>	Power supply ripple	V
<b>IS</b>	Supply current at V+	A
<b>VRQ</b>	Residual voltage on driver in saturated operating status ON	V
<b>VRQH</b>	Residual voltage on high-side driver in operating status ON	V
<b>VRQL</b>	Residual voltage on low-side driver in saturated operating status ON	V
<b>IQ</b>	Driver current in saturated operating status ON	A
<b>IQH</b>	Driver current on high-side driver in saturated operating status ON	A
<b>IQL</b>	Driver current on low-side driver in saturated operating status ON	A
<b>IQPK</b>	Maximum driver current in unsaturated operating status ON	A
<b>IQPKH</b>	Maximum driver current on high-side driver in unsaturated operating status ON	A
<b>IQPKL</b>	Maximum driver current on low-side driver in unsaturated operating status ON	A
<b>ILL</b>	Input load current at input C/Q to V0	A
<b>VTH</b>	Threshold voltage of receiver with reference to V0	V
<b>VTHH</b>	Threshold voltage of receiver for safe detection of a high signal	V
<b>VTHL</b>	Threshold voltage of receiver for safe detection of a low signal	V
<b>VHYS</b>	Hysteresis of receiver threshold voltage	V
<b>VI</b>	Input voltage at connection C/Q with reference to V0	V
<b>VIL</b>	Input voltage range at connection C/Q for low signal	V
<b>VIH</b>	Input voltage range at connection C/Q for high signal	V
<b>VD+</b>	Voltage drop on the line between the L+ connections on master and device	V
<b>VD-</b>	Voltage drop on the line between the L- connections on master and device	V
<b>VDQ</b>	Voltage drop on the line between the C/Q connections on master and device	V
<b>CQ</b>	Input capacity at C/Q connection	nF
<b>CLC</b>	Capacity for line compensation at C/Q connection	nF
<b>RLC</b>	Resistance for line compensation at C/Q connection	ohm
<b>CHF</b>	Capacity for HF attenuator at C/Q connection	nF
<b>RHF</b>	Resistance for HF attenuator at C/Q connection	ohm
<b>RS</b>	Current-limiting resistance for operation in PHY1	ohm
<b>CS</b>	Capacity for operation in PHY1	uF
<b>OVD</b>	Signal Overload Detect	–
<b>On/Off</b>	Driver's ON/OFF switching signal	–
<b>H/L</b>	High/low signal at receiver output	–
<b>f<sub>DTR</sub></b>	Data transfer rate	bps, bauds
<b>Δf<sub>DTR</sub></b>	Permissible deviation from data transfer rate	%
<b>T<sub>BIT</sub></b>	Bit time	s
<b>t<sub>H</sub></b>	Detection time for high level	s
<b>t<sub>L</sub></b>	Detection time for low level	s
<b>t<sub>ND</sub></b>	Noise suppression time	s
<b>t<sub>DR</sub></b>	Rise time	s



<b>t<sub>DF</sub></b>	Fall time	s
<b>r</b>	Time to reach a stable level with reference to the beginning of the start bit	T <sub>BIT</sub>
<b>s</b>	Time to exit a stable level with reference to the beginning of the start bit	T <sub>BIT</sub>
<b>WURQ</b>	Wake-up request pulse	–
<b>IQ<sub>WU</sub></b>	Amplitude of master's wake-up request current	A
<b>T<sub>WU</sub></b>	Pulse duration of wake-up request	s
<b>T<sub>REN</sub></b>	Receive enable	s
<b>T<sub>PON</sub></b>	Ramp-up time following power ON	s
<b>T<sub>RDL</sub></b>	Wake-up readiness following power ON	s
<b>T<sub>DMT</sub></b>	Delay time for establishment of communication on master	T <sub>BIT</sub>
<b>T<sub>DSIO</sub></b>	Delay time on device for transition to Standard IO mode following wake-up request	s
<b>T<sub>DWU</sub></b>	Wake-up retry delay	s
<b>n<sub>WU</sub></b>	Wake-up retry count	–
<b>T<sub>SD</sub></b>	Device detect time	s
<b>SIO</b>	Standard IO (digital switching mode)	–
<b>R<sub>L<sub>eff</sub></sub></b>	Loop resistance of cable	ohm
<b>C<sub>L<sub>eff</sub></sub></b>	Effective total cable capacity	nF
<b>t<sub>1</sub></b>	Character transfer delay on master	T <sub>BIT</sub>
<b>t<sub>2</sub></b>	Character transfer delay on device	T <sub>BIT</sub>
<b>t<sub>A</sub></b>	Response delay on device	T <sub>BIT</sub>
<b>T<sub>FRAME</sub></b>	Frame duration	T <sub>BIT</sub>
<b>T<sub>FRAMEIDLE</sub></b>	Idle time between two frames	s
<b>T<sub>CYC</sub></b>	Cycle time on frame layer	s
<b>T<sub>OFS</sub></b>	Temporal offset for process data processing on the device with reference to start of cycle	s
<b>ON-REQ</b>	On-request data	–

### 4.3 Conventions for service definitions

#### 4.3.1 General

The service model, service primitives, and time-sequence diagrams used are entirely abstract descriptions; they do not represent a specification for implementation.

#### 4.3.2 Service parameters

Service primitives are used to represent service user/service provider interactions (ISO/IEC 10731). They convey parameters which indicate information available in the user/provider interaction. In any particular interface, not all parameters need be explicitly stated.

The service specification of this PAS uses a tabular format to describe the component parameters of the service primitives. The parameters which apply to each group of service primitives are set out in tables. Each table consists of up to five columns for the

- 1) parameter name,
- 2) request primitive,
- 3) indication primitive,

- 4) response primitive, and
- 5) confirm primitive.

One parameter (or component of it) is listed in each row of each table. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the column.

- M parameter is mandatory for the primitive.
- U parameter is a User option, and may or may not be provided depending on dynamic usage of the service user. When not provided, a default value for the parameter is assumed.
- C parameter is conditional upon other parameters or upon the environment of the service user.
- (blank) parameter is never present.
- S parameter is a selected item.

Some entries are further qualified by items in brackets. These may be

- a) a parameter-specific constraint:
  - “(=)” indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table;
- b) an indication that some note applies to the entry:
  - “(n)” indicates that the following note "n" contains additional information pertaining to the parameter and its use.

### 4.3.3 Service procedures

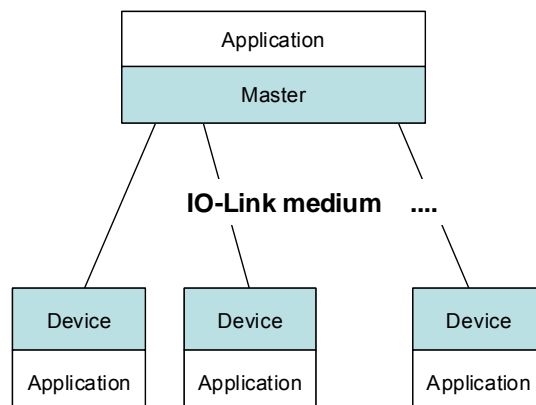
The procedures are defined in terms of

- the interactions between application entities through the exchange of Protocol Data Units, and
- the interactions between an communication layer service provider and an communication layer service user in the same system through the invocation of service primitives.

These procedures are applicable to instances of communication between systems which support time-constrained communications services within the communication layers.

## 5 IO-Link overview

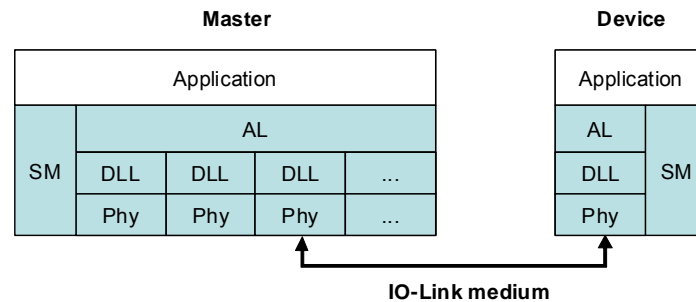
### 5.1 IO-Link topology



**Figure 1 — IO-Link topology**

Figure 1 shows the point-to-point communication topology between a master and several devices, usually sensors and actuators.

## 5.2 Specification outline



**Figure 2 — IO-Link layer structure**

Figure 2 shows the layer structures of the IO-Link master and the IO-Link device. The subsequent clauses in this document define the services and protocols of each layer.

Clause 6 (Phy = Physical layer) describes the physical properties and access to the transmission medium.

NOTE The EMC requirements have been completed

Clause 7 (DLL = Data link layer) describes data transport between master and device.

Clause 8 (AL = Application layer) defines the services and protocols for accessing process and on-request data (parameter settings, events, services).

Clause 9 (SM = System management) describes master and device startup.

Annex A lists error types, Annex B event codes, and Annex C the properties of service PDU indices. Annex D demonstrates the transmission of a service PDU example and Annex E the residual fault probability investigations on IO-Link.

This extended version of the specification contains additional annexes. The new Annex F comprises errata and supplements to the original specification dated November 2007. Annex G shows a form that can be used for a manufacturer's conformity declaration.

## 5.3 General characteristics

The distinguishing features of this system are:

- Single-master/single-device communication in half-duplex mode
- There are 3 baud rates: 4,800 bauds, 38,400 bauds and 230,400 bauds
- There are 2 physical interfaces: 2-wire system and 3-wire system

NOTE For product implementations the 2-wire physic shall not be considered even though it is specified herein. It will not be supported in future releases.

## 5.4 Compatibility

In respect of the binary p-switching or push/pull 3-wire sensors and switches, downward compatibility denotes switching behavior in accordance with standard IEC 60947-5-2.

In respect of the binary actuators, downward compatibility denotes switching behavior.

In respect of the master, downward compatibility denotes behavior in accordance with a digital input module.

Downward compatibility is only relevant for PHY2.

#### 5.4.1 Master

Following a power ON, a master supporting PHY2 shall behave like a digital input type 1 in accordance with IEC 61131-2 (Table IEC 61131 Chapter 5.3.2). This function is mandatory for an IO-Link master and is the default setting.

Configuration settings may be made for the master determining whether, when interacting with non-communicating devices on the interface, it behaves like a digital output. This function is optional for an IO-Link master.

The master may also be configured for PHY1 (if supported).

#### 5.4.2 Sensor

When not in Communication status, where its port is concerned, an IO-Link sensor shall behave like a sensor in accordance with IEC 60947-5-2, insofar as it is able to work sensibly as a switching sensor.

#### 5.4.3 Actuator

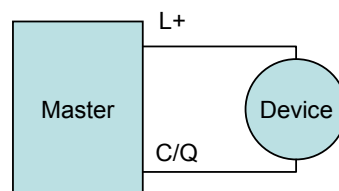
An IO-Link actuator is always in communication status.

## 6 Physical layer

### 6.1 General

#### 6.1.1 Overview

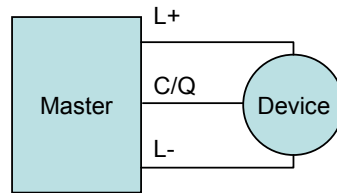
PHY1 combines communication and its power supply on a 2-wire interface as shown in Figure 3. Communication data and energy are transferred in the time division multiplex. A pause for energy transfer is inserted between each data block. Where this connection method is concerned, data transfer is always at a rate of 230 kBd. This transfer rate enables the required response times and energy transfer to be achieved. Support of PHY1 is optional for the master.



**Figure 3 — PHY1**

For product implementations PHY1 shall not be considered even though it is specified herein. It will not be supported in future releases.

PHY2 is a 3-wire system. Communication and the power supply run via separate lines, as illustrated in Figure 4. This removes the multiplexing of power and data we see in PHY1. The master is obliged to support PHY2.

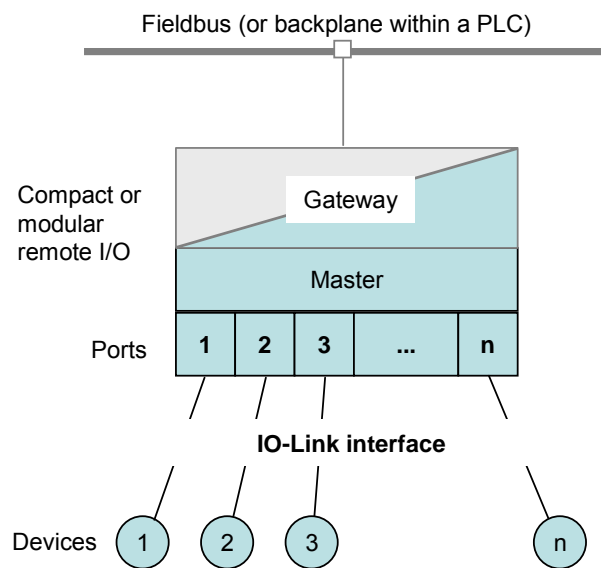


**Figure 4 — PHY2**

NOTE: Figure 3 and Figure 4 only consider the communication power supply.

### 6.1.2 Hierarchical composition

The IO-Link topology is described by means of a point-to-point link between a master and a device, see Figure 5. The master may feature a number of ports for the connection of devices. Only one device may be connected to each port. The master is connected to upper level control systems via a gateway using fieldbus or backplane transmission channels.



**Figure 5 — Hierarchical composition**

## 6.2 Transmitter/Receiver

The specification differentiates between static and dynamic parameters.

Static parameters describe levels and currents separately for master and device in the form of reference schematics.

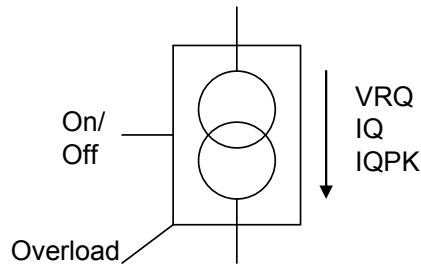
Dynamic parameters describe the signal-transmission process (specifically the receiver and analyzer function).

### 6.2.1 Static parameters

#### 6.2.1.1 General

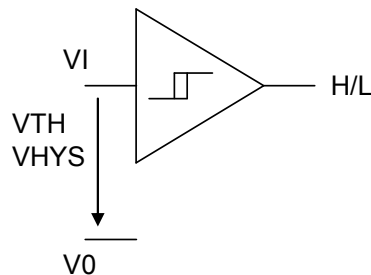
The driver is described by a reference schematic corresponding to Figure 6. On the master, a transmitter comprises a combination of two drivers and one current sink. On the device, in its simplest form, the transmitter takes the form of a p-switching driver. As an option there may be an additional n-switching or non-switching driver (this also introduces the option of push-pull operation).

In saturated operating status ON the descriptive variables are the residual voltage VR and the standard driver current IQ; in unsaturated operating status ON the descriptive variable is the peak current IQPK. The source is controlled by the On/Off signal. An overcurrent event is indicated at the "Overload" output.



**Figure 6 — Driver reference schematics**

The receiver is defined by the reference schematic in accordance with Figure 7. It performs the function of a comparator and is described by its switching thresholds VTH and a hysteresis VHYS between the switching thresholds. The output indicates the logic level (High or Low) at the receiver input.



**Figure 7 — Receiver reference schematics**

Figure 8 and Figure 9 show the reference schematics for the interconnection of master and device separately for PHY1 and PHY2.

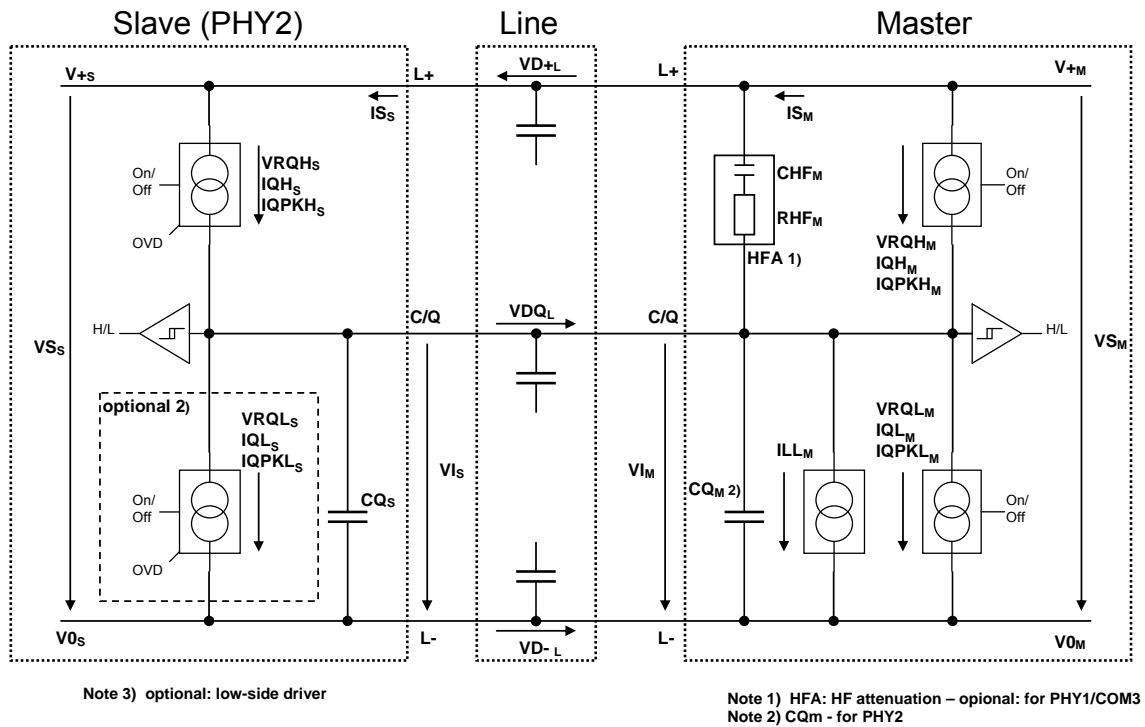


Figure 8 — Master and device reference schematics for PHY2

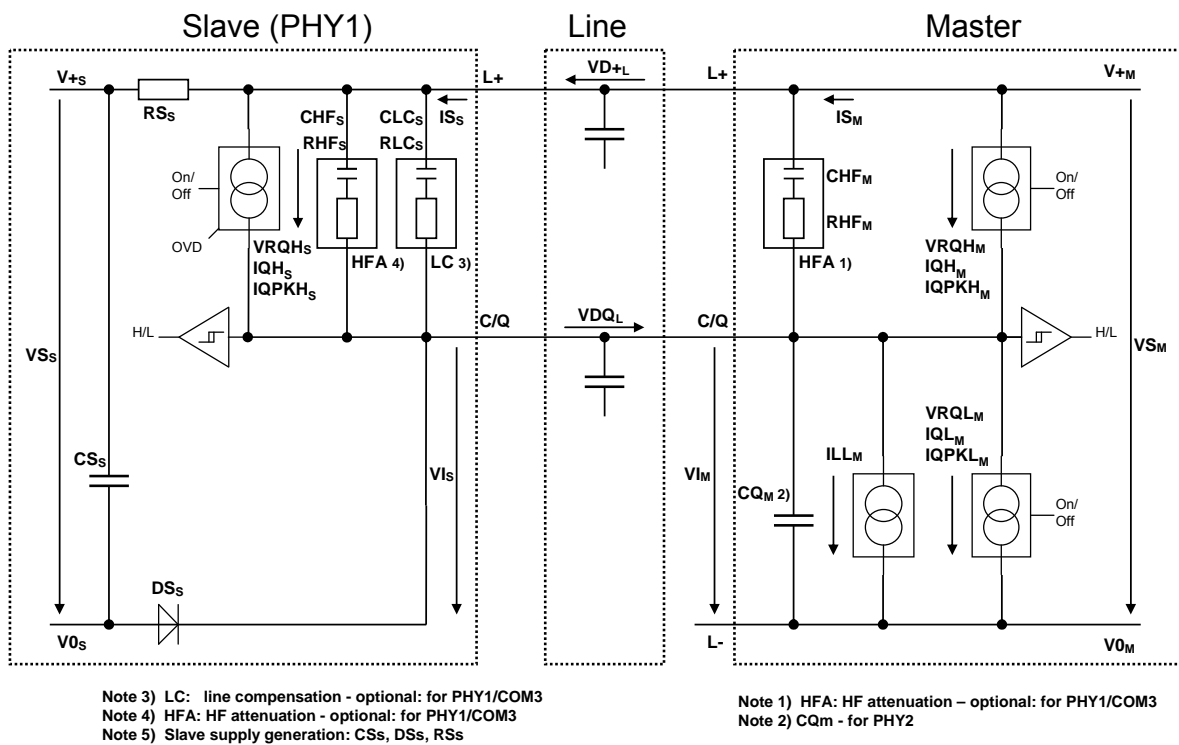


Figure 9 — Master and device reference schematics for PHY1

The illustrations and the parameter tables below contain the voltage level definitions as per Figure 10. The last letter appears in subscript and refers to the master (M), device (S) or line (L).

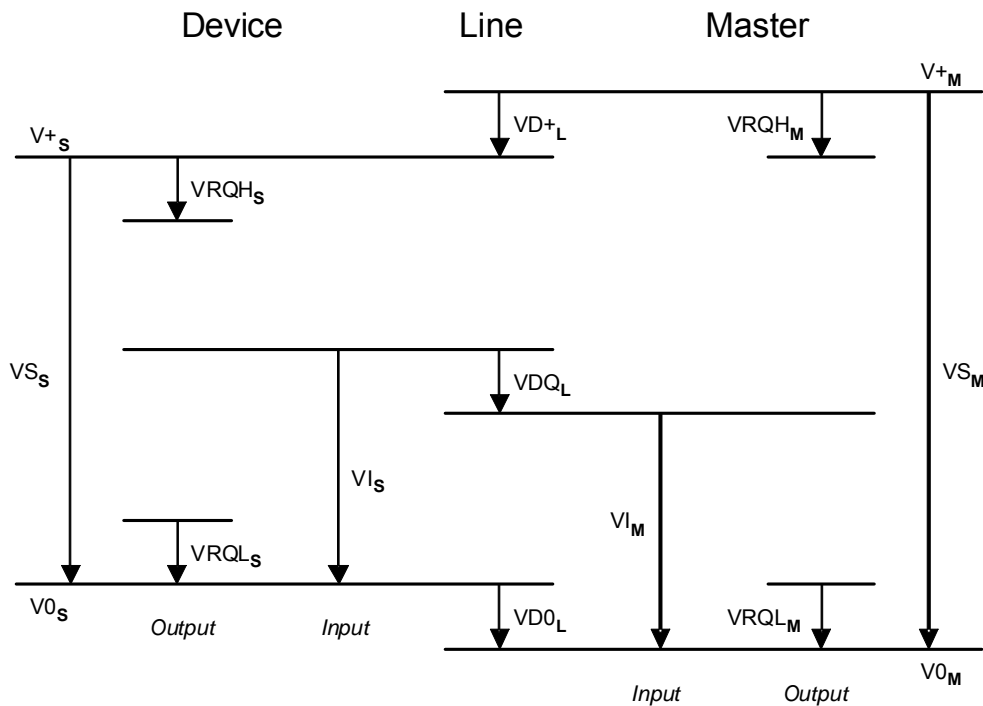


Figure 10 — Voltage level definitions

6.2.1.2 Receiver

The voltage range and switching threshold definitions are the same for master and device. The definitions in Table 1 apply.

Table 1 — Static parameters of receiver

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
VTHH <sub>S,M</sub>	Input threshold "H"	10,5	n/a	13	V	NOTE 1
VTHL <sub>S,M</sub>	Input threshold "L"	8	n/a	11,5	V	NOTE 1
VHYS <sub>S,M</sub>	Hysteresis between input thresholds "H" and "L"	0	n/a	n/a	V	Shall not be negative NOTE 2
VILS <sub>M</sub>	Permissible voltage range "L"	V <sub>0S,M</sub> - 1,0	n/a	n/a	V	With reference to relevant neg. supply voltage
VIHS <sub>M</sub>	Permissible voltage range "H"	n/a	n/a	V <sub>+S,M</sub> + 1,0	V	With reference to relevant pos. supply voltage.

NOTE 1 Thresholds are compatible with definition of Type 1 digital inputs in IEC61131-2.  
NOTE 2 Hysteresis voltage VHYS = VTHH-VTHL

Figure 11 illustrates the switching thresholds for the detection of Low and High signals.



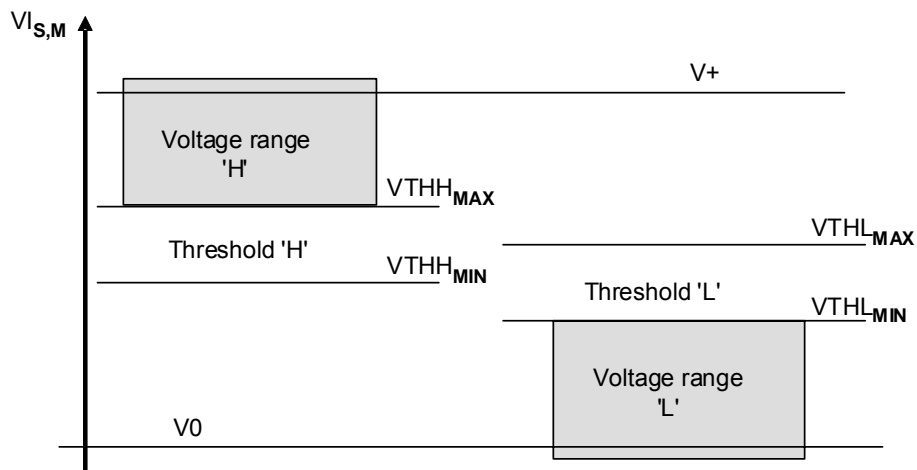


Figure 11 — Switching thresholds

### 6.2.1.3 Master

The definitions in Table 2 are valid for the static parameters of a master.

Table 2 — Static master parameters

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
$V_{SM}$	Supply voltage for devices	20	24	30	V	
$I_{SM}$	Supply current for devices	200	n/a	n/a	mA	> 200 mA external supply required for devices
$I_{LLM}$	Load or discharge current for $0 < V_{IM} < 5V$	n/a	n/a	200	mA	Only for PHY1 NOTE 1
	$15 < V_{IM} < 30V$	50	n/a	n/a	mA	
$I_{LLM}$	Load or discharge current for $0 < V_{IM} < 5V$	0	n/a	15	mA	Only for PHY2 NOTE 2
	$15 < V_{IM} < 30V$	5	n/a	15	mA	
$VRQH_M$	Residual voltage "H"	n/a	n/a	3	V	Voltage drop compared with $V^+_M$ at max. driver current $I_{QH_M}$
$VRQL_M$	Residual voltage "L"	n/a	n/a	3	V	Voltage drop compared with $V_{0M}$ at max. driver current $I_{QL_M}$
$I_{QH_M}$	DC driver current "H"	100	n/a	200	mA	
$I_{QPKH_M}$	Output peak current "H"	500	n/a	n/a	mA	Absolute value NOTE 5
$I_{QL_M}$	DC driver current "L"	100	n/a	200	mA	
$I_{QPKL_M}$	Output peak current "L"	500	n/a	n/a	mA	Absolute value NOTE 5
$C_{QM}$	Input capacitance	n/a	n/a	1,0	nF	f=0-4MHz

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
						NOTE 3
RLC <sub>M</sub>	Line termination	9	10	11	Ω	Affects V+M
CLC <sub>M</sub>		0,9	1,0	1,1	nF	NOTE 4
NOTE 1 Load currents for PHY1 device during communication and communication break						
NOTE 2 Currents are compatible with definition of Type 1 digital inputs in IEC61131-2.						
NOTE 3 Valid for PHY2 operation.						
NOTE 4 The line termination is only necessary for PHY1/COM3 operation (combi master).						
NOTE 5 Wake-up request current – see Wake-up definition.						

#### 6.2.1.4 Device

The definitions in Table 3 are valid for the static parameters of a device.

**Table 3 — Static device parameters**

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
V <sub>SS</sub>	Supply voltage	18	24	30	V	
ΔV <sub>SS</sub>	Ripple	n/a	n/a	1,3	V <sub>pp</sub>	Peak-to-peak absolute value limits shall not be exceeded. fripple = DC – 100kHz Only valid for PHY2
VR <sub>QHs</sub>	Residual voltage "H"	n/a	n/a	3	V	Voltage drop compared with V+s (IEC 60947-5-2)
VR <sub>QLs</sub>	Residual voltage "L"	n/a	n/a	3	V	Only for PHY2 Voltage drop compared with V0s Implementation of push-pull output stages
I <sub>QHs</sub>	DC driver current P-switching output	50	n/a	minimum (I <sub>QPKL<sub>M</sub></sub> )	mA	Min. value due to fallback to operation as input in acc. with IEC 61131 Type 2
I <sub>QLs</sub>	DC driver current N-switching output	0	n/a	minimum (I <sub>QPKH<sub>M</sub></sub> )	mA	Only for PHY2 Only for push-pull output stages
C <sub>Qs</sub>	Input capacitance	0	n/a	1,0	nF	PHY2 only Capacitance between C/Q and L- of device in receive status NOTE 1
C <sub>Ss</sub>	Effective capacity	n/a	n/a	n/a	μF	PHY1 only Capacitance for device supply between L+ and C/Q NOTE 2

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
RHF <sub>S</sub>	HF attenuator	n/a	10	n/a	Ω	PHY1/COM3 operation Effective between L+ and C/Q NOTE 3
CHF <sub>S</sub>		n/a	470	n/a	pF	
RLC <sub>S</sub>	Line termination	n/a	n/a	150	Ω	PHY1/COM3 operation Effective between L+ and C/Q NOTE 4
CLC <sub>S</sub>		n/a	n/a	n/a	μF	

NOTE 1 Maximum of 10 nF capacitive load permissible for push-pull stages on device. In this case the limits shall be defined from the dynamic parameters (in the baud rates supported by the device). The value 1nF is applicable for COM3.

NOTE 2 The value shall be selected so that the device's supply voltage remains within the limits for VSs during communication (frame time). During the communication break (idle time) CSs is loaded via C/Q.

NOTE 3 HF attenuation is required for PHY1/COM3 operation.

NOTE 4 The line termination is required for signal shaping in PHY1/COM3 operation. The value for CLCs is determined by the frame length, cycle time and current consumption ISs of the device.

Typical values are: 2,7 uF for a 4-byte frame, ISs > 50 mA, 4,7 uF for a 6-byte frame, ISs > 50 mA, 10 uF for a 4-byte frame, ISs < 50 mA, 15 uF for a 6-byte frame, ISs < 50 mA

## 6.2.2 Dynamic parameters

### 6.2.2.1 Transmission method

The “Non Return to Zero” (NRZ) code is used for bit-by-bit coding. Here, logic value “1” (DL\_symbol = “ONE”) corresponds to a voltage value of 24 volts between the L+ and C/Q lines. Logic value “0” (DL\_symbol = “ZERO”) corresponds to a voltage value of 0 volts between the L+ and C/Q lines.

The open-circuit level on the C/Q line is 0 V with reference to L- and -24 V with reference to L+. A start bit has logic value “0”.

The UART format is used for character-by-character coding. A UART character is a character limited by a start signal and a stop signal and used for the asynchronous transmission of a telegram. The structure of the UART character defined for IO-Link is illustrated in Figure 12.

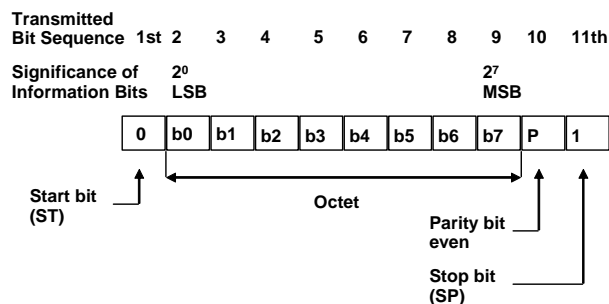


Figure 12 — UART character

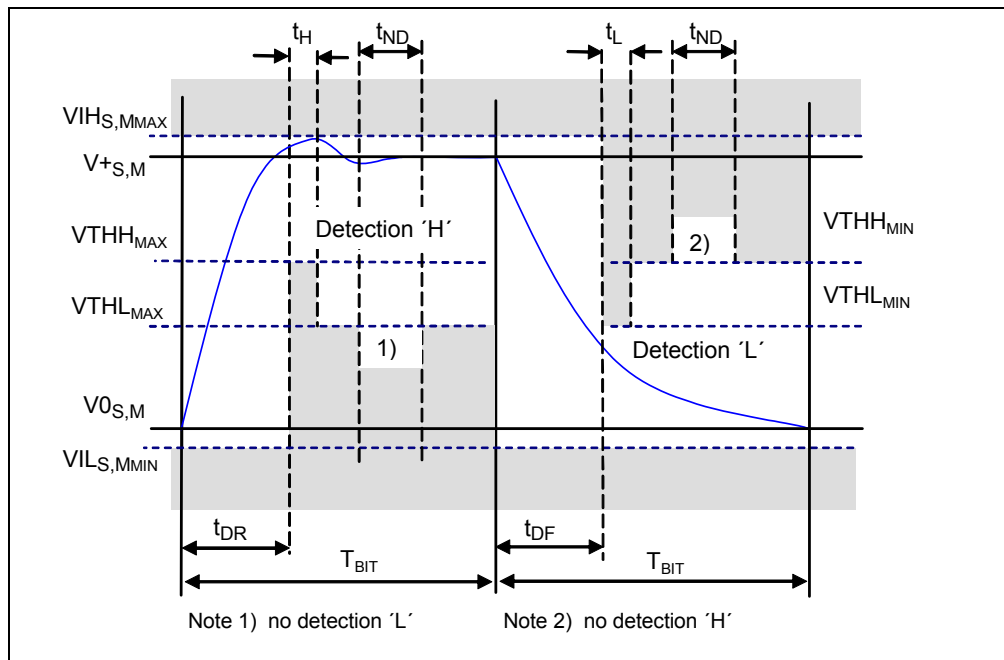
The definition of the UART character is based on ISO/IEC 1177:1985, ISO/IEC 2022:1986.

### 6.2.2.2 Transmission parameters

The dynamic transmission parameters are illustrated in the form of an eye diagram with permissible signal ranges, see Figure 13. These ranges are applicable for both transmitter and receiver on both master and device.

Regardless of boundary conditions, the transmitter shall generate a voltage characteristic on the receiver's C/Q connection that is within the permissible range of the eye diagram.

The receiver shall detect a voltage characteristic on the C/Q connection as valid bits within the permissible range of the eye diagram. Signal characteristics in the "No detection" range shall not produce an invalid bit.



**Figure 13 — Eye diagram for "H" and "L" detection**

Bits are transmitted asynchronously in a bit sequence starting with a start bit and ending with a stop bit (UART frame). This bit sequence is 11 bits long.

In order for a UART frame to be detected, a signal characteristic as illustrated in Figure 14 is required on the receiver. The signal delay time between the C/Q signal and the UART input shall be taken into account. Time  $T_{BIT}$  always indicates the receiver's bit rate.

For every bit  $n$  in the bit sequence ( $n=1\dots 11$ ) of a UART frame, the time  $(n-r)T_{BIT}$  designates the time at the end of which a safe level shall be reached in the "H" or "L" ranges as illustrated in the eye diagram in Figure 13. The time  $(n-s)T_{BIT}$  describes the time, which shall elapse, before the level may change. Reference shall always be made to the eye diagram in Figure 13 where signal characteristics within a bit time are concerned.

This representation permits a variable weighting of the influence parameters bit-width distortion and data-rate tolerance on the device.

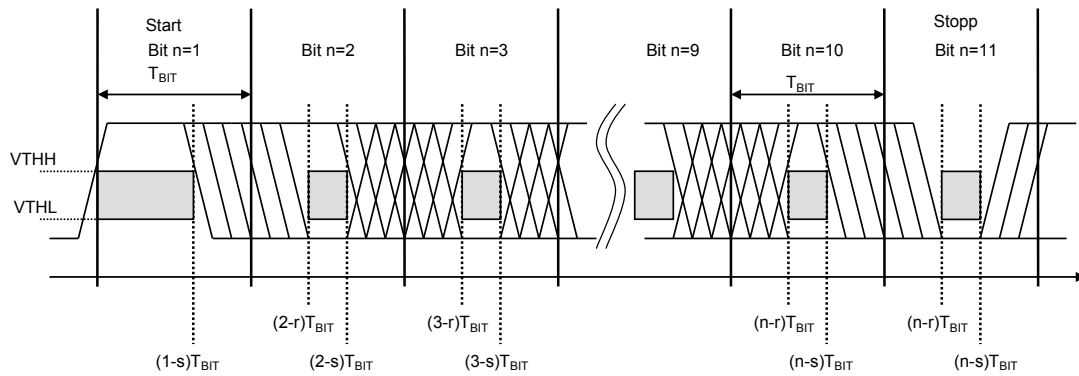


Figure 14 — Eye diagram for safe detection of a UART frame

Table 4 specifies the dynamic transmission parameters.

Table 4 — Transmission parameters

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
$f_{DTR}$	Data transfer rate	n/a	4,8 38,4 230,4	n/a	kBaud	COM1 COM2 COM3
$T_{BIT}$	Bit time at 4,8 kBaud at 38,4 kBaud at 230,4 kBaud		208,33 26,04 4,34		$\mu$ s	
$\Delta f_{DTRM}$	Master data transfer rate accuracy at 4,8 kBaud at 38,4 kBaud at 230,4 kBaud	-0,1 -0,1 -0,1	n/a n/a n/a	+0,1 +0,1 +0,1	% % %	Tolerance of transfer data rate on master $\Delta T_{bit}/T_{bit}$
r	Start of detection time within a bit with reference to the start of the start bit	0,65	n/a	n/a	TBit	Calculated in each case from the end of the bit in question at a scan rate of 8x (NOTE) See F.2.1
s	End of detection time within a bit with reference to the start of the start bit	n/a	n/a	0,22	$T_{BIT}$	Calculated in each case from the end of the bit in question at a scan rate of 8x (NOTE) See F.2.1
$T_{DR}$	Rise time at 4,8 kBaud at 38,4 kBaud at 230,4 kBaud	0 0 0	n/a n/a n/a	0,20 41,7 5,2 869	$T_{BIT}$ $\mu$ s $\mu$ s ns	With reference to the bit time unit

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
$T_{DF}$	Fall time	0	n/a	0,20	$T_{BIT}$	With reference to the bit time unit
	at 4,8 kBaud	0	n/a	41,7	$\mu s$	
	at 38,4 kBaud	0	n/a	5,2	$\mu s$	
	at 230,4 kBaud	0	n/a	869	ns	
$t_{ND}$	Noise suppression time	n/a	n/a	1/16	$T_{BIT}$	Permissible duration of a receive signal above/below the detection threshold without detection taking place
$t_H$	Detection time High	1/16	n/a	n/a	$T_{BIT}$	Duration of a receive signal above the detection threshold for H level
$t_L$	Detection time Low	1/16	n/a	n/a	$T_{BIT}$	Duration of a receive signal below the detection threshold for H level
NOTE For frequency accuracy/edge steepness compensation: Systems with significant edge steepness on the device side may compensate increased edge steepness against reduced frequency accuracy. The summary bit-width corruption on the last bit of the UART byte shall produce a safe level in the range of the scan time in accordance with specification.						

### 6.2.2.3 Wake-up

The wake-up request is sent by the master by way of a definitive communication request and also initiates the necessary changeover of the devices to a ready-to-receive state.

The wake-up request (WURQ) is a current event induced by the master. The wake-up request comprises the following phases:

- Injection of a current  $I_{Q_{WU}}$  by the master on the basis of the level at the C/Q connection. For an input signal equivalent to logic "1" this is a current source; for an input signal equivalent to logic "0" it is a current sink.
- Response time of the device until it is ready to receive.

The exact sequence of events is illustrated in Figure 15.

The WakeUpRequest precedes the attempt to establish communication. This latter process is part of the Data link layer specification (7.2.2.1.3).

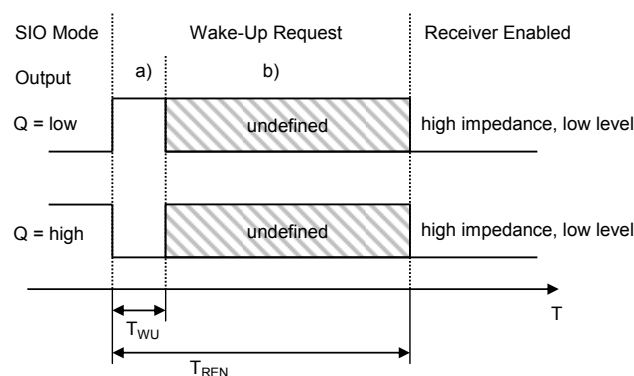


Figure 15 — Wake-up request

Table 5 specifies the current and time parameters associated with the wake-up request.

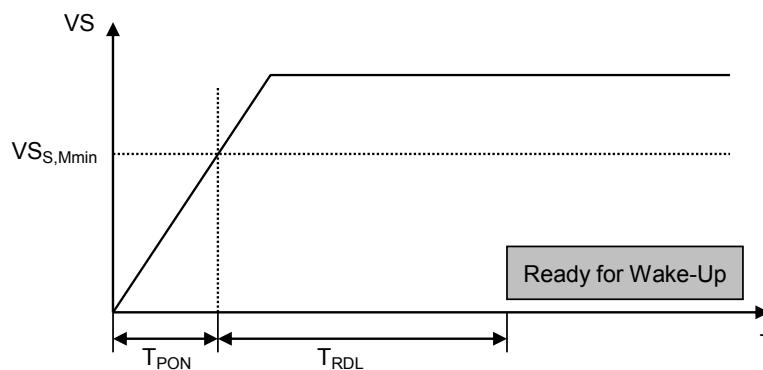
**Table 5 — Wake-up request definitions**

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
I <sub>QWU</sub>	Amplitude of master's wake-up request current	I <sub>QPKL<sub>M</sub></sub> or I <sub>QPKH<sub>M</sub></sub>	n/a	n/a	mA	Sign of current following switching status of device
T <sub>WU</sub>	Pulse duration of wake-up request	75	n/a	85	μs	Master property
T <sub>REN</sub>	Receive enable delay	n/a	n/a	500	μs	Device property

## 6.2.3 Power supply

### 6.2.3.1 Power-on requirements

The power-on response of a device is defined by the ramp-up time of the power supply and the internal time requirement until the device is ready for operation, see Figure 16. A wake-up can only be detected once the device is ready to operate.



**Figure 16 — Power-on timing**

Table 6 specifies the parameters for the power-on response.

**Table 6 — Power-on definitions**

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
T <sub>PON</sub>	Ramp-up time V <sub>S<sub>M</sub></sub> /V <sub>S<sub>S</sub></sub>	n/a	n/a	5	ms	Time for valid voltage following power-on (PHY1 only) NOTE 1
T <sub>RDL</sub>	Wake-up readiness following power-on	n/a	n/a	300	ms	Device ramp-up time until ready for wake-up detection NOTE 2
NOTE 1 On PHY1, the ramp-up time is essentially defined by the input capacitance effective on the device side (CSs and CLCs).						
NOTE 2 Equivalent to time delay before availability in IEC 60947-5-2.						

### 6.2.3.2 IO-Link-powered device

In PHY1, communication and power transmission from master to device take place on the same line in accordance with the time division multiplex procedure. The multiplex ratio is determined by the device in the exportable parameter “Minimum cycle time”.

In PHY2, power transmission and communication take place on separate lines. There is no time division multiplex procedure.

The maximum supply current for devices is defined by the maximum driver currents of the master in Table 2.

### 6.2.3.3 Separately-powered device

In both PHY1 and PHY2, the device may be supplied with power via an external electrically-isolated current supply interface. The communication section of the device shall always be powered from the same voltage source as the master.

## 6.2.4 Medium

### 6.2.4.1 Connector

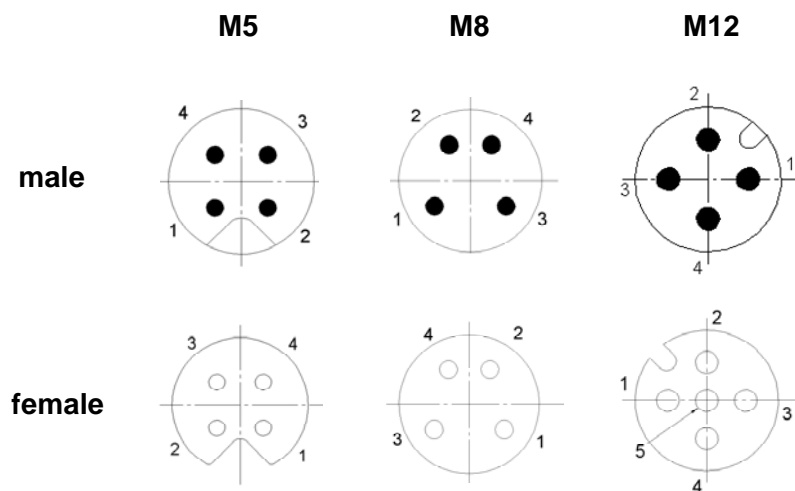
The master and device pin assignment is based on IEC 60947-5-2.

The IO-Link plug connection is female on the master and male on the device. Table 7 lists the pin assignment and Figure 17 illustrates the layout for M12, M8 and M5 connections (see F.2.2).

**Table 7 — Pin assignment**

Pin	Signal	Designation	Remark
1	L+	Power supply (+)	
2	n/a	n/a	Not specified for IO-Link (see F.2.2)
3	L-	Power supply (-)	Not used on PHY1
4	C/Q	Communication signal	On PHY1 also power supply (-)
5	n/a	n/a	Not specified for IO-Link. Shall not be connected on the master side

NOTE M12 is always 5 pins on the master side (female).



**Figure 17 — Pin layout front view**



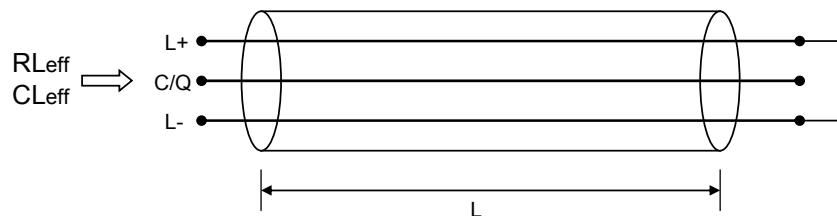
### 6.2.4.2 Cable

To ensure functional reliability, standard cables with parameters complying with Table 8 shall be used.

**Table 8 — Cable parameters**

Parameter	Minimum	Typical	Maximum	Unit
Length	0	n/a	20	m
Overall ohmic loop resistance $RL_{eff}$				Ohm
PHY1	n/a	n/a	3,0	
PHY2	n/a	n/a	6,0	
Effective line capacitance $CL_{eff}$	n/a	n/a	3,0	nF

The loop resistance  $RL_{eff}$  and the effective line capacitance  $CL_{eff}$  may be measured as illustrated in Figure 18.



**Figure 18 — Reference schematic for effective line capacitance and loop resistance**

Table 9 shows the assignment of cable conductors and their color codes.

**Table 9 — Cable conductor assignment**

Signal	Designation	Color	Remark
L-	Power supply (-)	blue	Not used on PHY1
C/Q	Communication signal	black	On PHY1 also power supply (-)
L+	Power supply (+)	brown	
NOTE Corresponding to IEC 60947-5-2			

## 6.3 EMC requirements

The EMC requirements of this specification are only relevant for the functionality of the IO-Link interface of the device that incorporates this interface. The primary function of the device and the relevant EMC requirements for that functionality are not in the scope of this specification. For this purpose the device specific product standards shall be applied.

To ensure proper operating conditions of the IO-Link interface, the test configurations defined in section 6.3.5 (IO-Link Masters) or 6.3.6 (IO-Link Devices) shall be maintained during all EMC tests. The tests required in the EUTs product standard may alternatively be performed in SIO-mode.

### 6.3.1 Operating conditions

The evaluation of the IO-Link communication should be done during the startup phase with the cycle times as given in Table 10. This leads in most cases to the minimal time requirements for performing the tests. Alternatively a manufacturer may decide to do the evaluation during normal operation of the device, but then he must confirm that the required number of frames specified below is evaluated during each test.

### 6.3.2 Performance criteria

#### a) Performance Criterion A

The communication element operating at an average cycle time as defined in Table 10 shall produce no more than six detected frame errors in the number of frames given in Table 10.

No interruption of communication is allowed.

**Table 10 — Cycle times**

Baud Rate	TCycle	Frames
4,8 kBaud	18 ms	21 000
38,4 kBaud	2,3 ms	35 000
230 kBaud	0,4 ms	55 000

NOTE Performance Criterion A is defined on the basis of IEC61158-2, which permits one residual error in 20 years at 1600 frames/sec ( $T_{\text{Cycle}} = 0,625$  ms). This is equivalent to a residual fault probability (RFW) as shown in formula (1).

$$\text{RFW}_{\text{IEC61158}} = \frac{0,625 \text{ ms}}{20 \text{ years}} \approx 10^{-12} \quad (1)$$

Based on the different error detection mechanisms one can calculate the bit error rate (BER) of IEC61158-2 (formula (2)) and IO-Link (XOR6 N=4) with the help of Figure E.1 in this specification (formula (3)).

$$\text{BER}_{\text{IEC61158}} = 10^{\frac{\log(\text{RFW}_{\text{IEC61158}}) - 3,92}{3,95}} \quad (2)$$

$$\text{BER}_{\text{IO-Link}} = 10^{\frac{\log\left(\frac{T_{\text{Cycle}}}{20 \text{ Years}}\right) - 2,71}{3,95}} \quad (3)$$

Derived from the definition of performance criterion A in IEC 61158-2 with a maximum of 6 detected frame errors within 100 000 frames, the number of required frames for EMC tests on IO-Link is calculated with the help of formula (4).

$$n_{\text{frames}_{\text{IO-Link}}} = 10^5 * \frac{\text{BER}_{\text{IEC61158}}}{\text{BER}_{\text{IO-Link}}} \quad (4)$$

#### b) Performance Criterion B

The error rate of criterion A shall also be satisfied after but not during the test. No change of actual operating state (e.g. permanent loss of communication) or stored data is allowed.

### 6.3.3 Required immunity tests

Table 11 shows the EMC tests to be performed.

**Table 11 — EMC test levels**

Phenomena Basic Standard	Test Level	Performance Criterion	Remarks
Electrostatic discharges (ESD) IEC 61000-4-2	Air discharge: ± 8 kV  Contact discharge: ± 4 kV	B	See NOTE 1
Radio-frequency electromagnetic field. Amplitude modulated IEC 61000-4-3	80 – 1000 MHz 10 V/m  1400 – 2000 MHz 3 V/m  2000 – 2700 MHz 1 V/m	A	See NOTE 1 + 2
Fast transients (Burst) IEC 61000-4-4	± 1 kV  ± 2 kV	A  B	See NOTE 3
Surge IEC 61000-4-5	Not required for IO-Link interconnect		IO-Link interconnect is limited to max. 20 m
Radio-frequency common mode IEC 61000-4-6	0,15 – 80 MHz 10 VEMF	A	See NOTE 2 + 4
Voltage dips and interruptions IEC 61000-4-11	Not required for IO-Link interconnect		
NOTE 1 As this phenomenon influences the entire device under test, an existing device specific product standard takes precedence over the test levels specified here.			
NOTE 2 The test shall be performed with a step size of 1 % and a dwell of 1 sec. If a single frame error occurs at a certain frequency, that frequency shall be tested until the number of frames according Table 10 has been transmitted or until 6 frame errors have occurred.			
NOTE 3 Depending on the baud rate the test time varies. The test time shall be at least one minute (with the transmitted frames and the permitted errors increased accordingly). At least the number of frames according Table 10 shall be transmitted during the test.			
NOTE 4 This phenomenon is expected to influence most probably the EUTs internal analogue signal processing and only with a very small probability the functionality of the IO-Link communication. Therefore an existing device specific product standard takes precedence over the test levels specified here.			

### 6.3.4 Required emission tests

The definition of emission limits is not in the scope of this specification. The requirements of the device specific product family or generic standards apply, usually for general industrial environments the IEC 61000-6-4.

All emission tests shall be performed at the fastest possible communication rate with the fastest cycle time.

### 6.3.5 Test configurations for IO-Link masters

For the test of IO-Link masters the followings rules apply:

- In the following test setup diagrams only the IO-Link and the power supply cables are shown. All other cables shall be treated as required by the relevant product standard.
- Grounding of the master and the devices according to the relevant product standard or manual.
- Where not otherwise stated, the IO-Link cable shall have an overall length of 20 m. Excess length laid as an inductive coil with a diameter of 0,3 m, where applicable mounted 0,1 m above RefGND.
- Where applicable, the auxiliary devices shall be placed 10 cm above RefGND.

- A typical test configuration consists of the IO-Link master and two IO-Link devices, except for the RF common mode test, where only one device shall be used.

### 6.3.5.1 Electrostatic discharges

Figure 19 shows the test setup for electrostatic discharge.

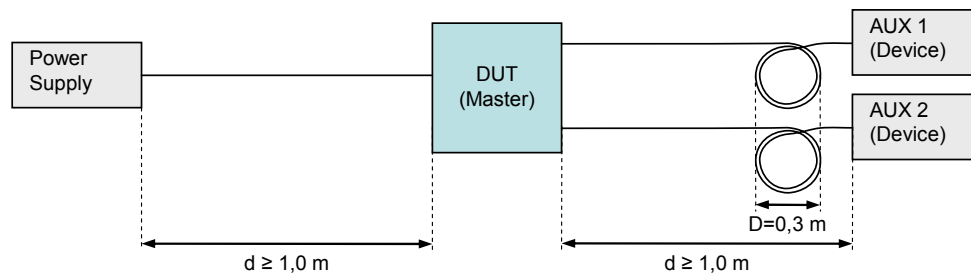


Figure 19 — Test setup for electrostatic discharge

### 6.3.5.2 Radio-frequency electromagnetic field

Figure 20 shows the test setup for radio-frequency electromagnetic field.

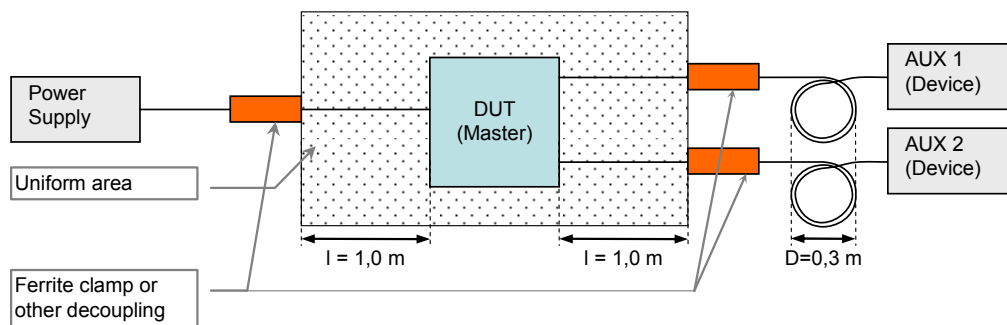


Figure 20 — Test setup for RF electromagnetic field

### 6.3.5.3 Fast transients (Burst)

Figure 21 shows the test setup for fast transients.

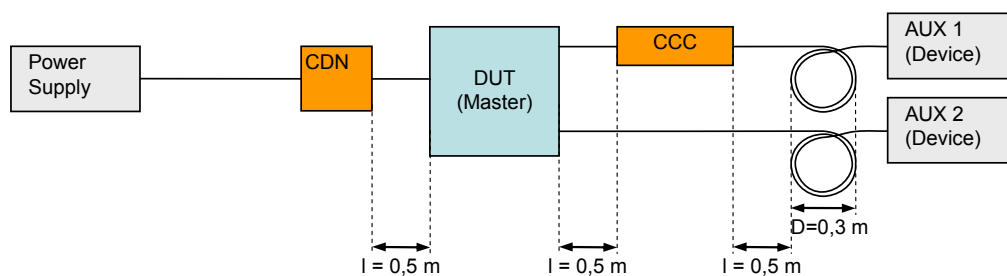


Figure 21 — Test setup for fast transients

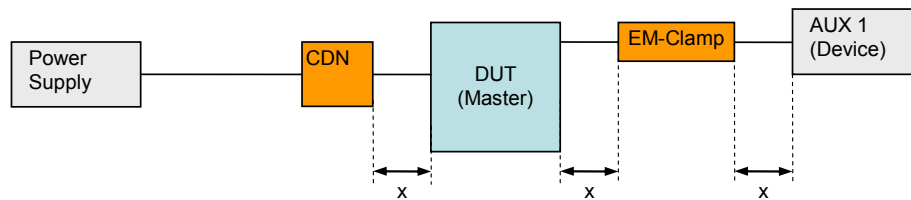
NOTE 1 No coupling into IO-Link line to AUX 2 is required

NOTE 2 CDN: Coupling/Decoupling Network

NOTE 3 CCC: Capacitive coupling clamp

### 6.3.5.4 Radio-frequency common mode

Figure 22 shows the test setup for radio frequency common mode.



**Figure 22 — Test setup for RF common mode**

NOTE 1  $0,1 \text{ m} < x < 0,3 \text{ m}$

NOTE 2 IO-Link overall cable length = 1 m

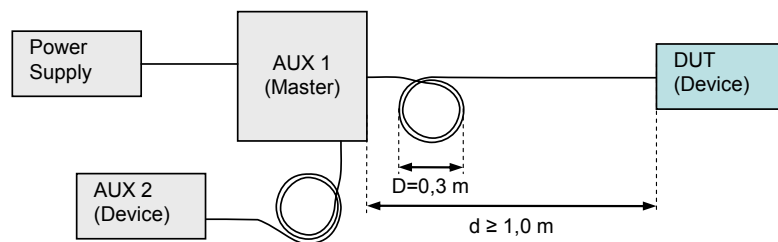
### 6.3.6 Test configurations for IO-Link devices

For the test of IO-Link devices the followings rules apply:

- In the following test setup diagrams only the IO-Link and the power supply cables are shown. All other cables shall be treated as required by the relevant product standard.
- Grounding of the master and the devices according to the relevant product standard or manual.
- Where not otherwise stated, the IO-Link cable shall have an overall length of 20 m. Excess length laid as an inductive coil with a diameter of 0,3 m, where applicable mounted 0,1 m above RefGND.
- Where applicable, the auxiliary devices shall be placed 10 cm above RefGND.

#### 6.3.6.1 Electrostatic discharges (device)

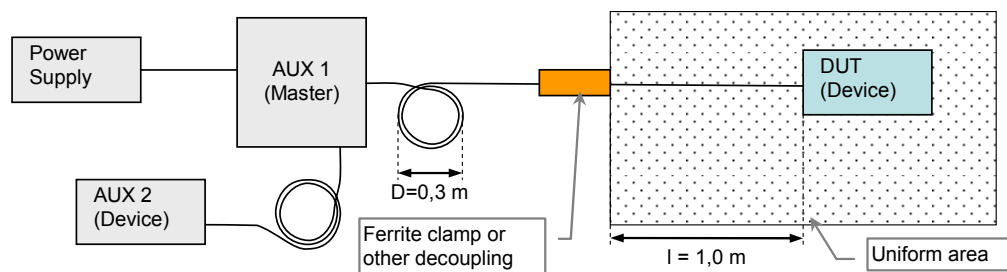
Figure 23 shows the test setup for electrostatic discharge.



**Figure 23 — Test setup for electrostatic discharge (device)**

#### 6.3.6.2 Radio-frequency electromagnetic field (device)

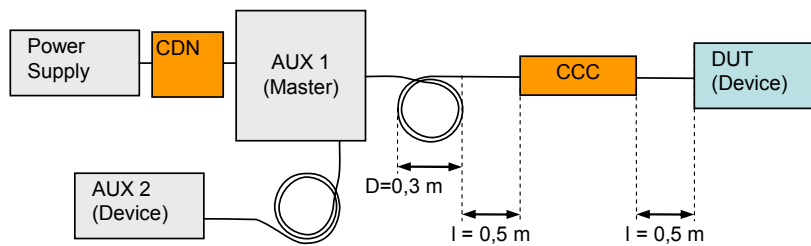
Figure 24 shows the test setup for radio-frequency electromagnetic field.



**Figure 24 — Test setup for RF electromagnetic field (device)**

#### 6.3.6.3 Fast transients / burst (device)

Figure 25 shows the test setup for fast transients.



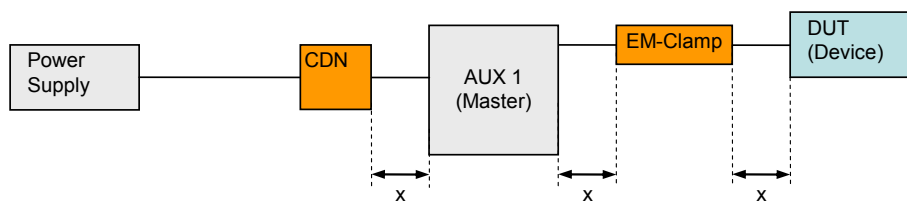
**Figure 25 — Test setup for fast transients (device)**

NOTE 1 CDN: Coupling/Decoupling Network, here only used for decoupling

NOTE 2 CCC: Capacitive coupling clamp

#### 6.3.6.4 Radio-frequency common mode (device)

Figure 26 shows the test setup for radio frequency common mode.



**Figure 26 — Test setup for RF common mode (device)**

NOTE 1  $0,1 \text{ m} < x < 0,3 \text{ m}$

NOTE 2 IO-Link overall cable length = 1 m

## 7 Data link layer

### 7.1 Data link layer services

#### 7.1.1 Services depending on station type

This chapter describes the services the data link layer provides to the application layer and system management via its upper interface. Table 12 lists the assignment of master and device to their roles as client or server in the context of the execution of individual DL services.

**Table 12 — Assignment of services to master and device**

Service name	Master	Device
DL-ReadParam	I	(R)
DL-WriteParam	I	(R)
DL-SPduTransport	I R	I R
DL-PDOutputUpdate	LR	
DL-PDOutputTransport	LI	
DL-PDInputUpdate		LR
DL-PDInputTransport		LI
DL-PDCycle	LI	LI
DL-SetMode	LR	
DL-Mode	LI	

Service name	Master	Device
DL-Event	R (R)	I R
DL-Control	I	R
<b>Key</b> I Initiator of service R Receiver (Responder) of service (R) Receiver (Responder) of service (but no interaction with application) LR Local service triggered by the application layer LI Local service triggered by the data link layer		

### 7.1.2 DL-ReadParam

The DL-ReadParam service is used to read out a parameter value from the device via the "Parameter" data channel. The parameters of the service primitives are listed in Table 13.

**Table 13 — DL-ReadParam**

Parameter name	.req	.cnf
Argument	M	
Address	M	
Result (+)		S
Value		
Result (-)		S
ErrorInfo		M

#### **Argument**

The service-specific parameters of the service request are transmitted in the argument.

#### **Address**

This parameter specifies the address of the requested parameter, i.e., the parameter addresses within the "Parameter" data channel.

Permitted values: 0 .. 31

#### **Result (+):**

This selection parameter indicates that the service request has been executed successfully.

#### **Value**

This parameter specifies read parameter values.

Parameter type: OctetString

#### **Result (-):**

This selection parameter indicates that the service request has failed.

#### **ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: NO\_COMM, STATE\_CONFLICT

### 7.1.3 DL-WriteParam

The DL-WriteParam service is used to write a parameter value to the device via the "Parameter" data channel. The parameters of the service primitives are listed in Table 14.

**Table 14 — DL-WriteParam**

Parameter name	.req	.cnf
Argument	M	
Address	M	
Value	M	
Result (+)		S
Result (-)		S
ErrorInfo		M

#### Argument

The service-specific parameters of the service request are transmitted in the argument.

#### Address

This parameter specifies the address of the requested parameter, i.e., the parameter addresses within the "Parameter" data channel. The values 0 is not permitted.

Permitted values: 1 .. 31, in accordance with parameter access rights

#### Value

This parameter specifies the parameter value to be written.

Parameter type: OctetString

#### Result (+):

This selection parameter indicates that the service request has been executed successfully.

#### Result (-):

This selection parameter indicates that the service request has failed.

#### ErrorInfo

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: NO\_COMM, STATE\_CONFLICT

### 7.1.4 DL-SPduTransport

The DL-SPduTransport service is used to transport a Service PDU. This service is used by the master to send a service request on the master application layer to the device. It is used by the device to send a service response to the master on the device application layer. The parameters of the service primitives are listed in Table 15.

**Table 15 — DL-SPduTransport**

Parameter name	.req	.ind	.cnf
Argument	M	M	
SPDU	M	S(=)	
Status		S	
Result (+)			S



Parameter name	.req	.ind	.cnf
Result (-)			S
ErrorInfo			M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**SPDU**

This argument specifies the service primitive encoded on the application layer.

Parameter type: OctetString

**Status**

This parameter specifies a fault condition identified during the service. This selection parameter only occurs on the master's data link layer.

Parameter type: Unsigned8

Permitted values: CHECKSUM\_ERROR

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**Result (-):**

This selection parameter indicates that the service request has failed.

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: NO\_COMM, STATE\_CONFLICT

**7.1.5 DL-PDOutputUpdate**

The master's application layer uses the DL-PDOutputUpdate service to update the output data (process data from master to device) on the data link layer. The parameters of the service primitives are listed in Table 16.

**Table 16 — DL-PDOutputUpdate**

Parameter name	.req	.cnf
Argument	M	
OutputData	M	
Result (+)		S
TransportStatus		M
Result (-)		S
ErrorInfo		M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**OutputData**

This argument specifies the process data provided by the application layer.

Parameter type: OctetString

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**TransportStatus**

This parameter indicates whether the data link layer is in a state permitting data to be transferred to the communication partner(s).

Parameter type: TRANSPORTSTATUS\_TYPE

Permitted values: YES, NO

**Result (-):**

This selection parameter indicates that the service request has failed.

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: NO\_COMM, STATE\_CONFLICT

**7.1.6 DL-PDOutputTransport**

The data link layer on the device uses the DL-PDOutputTransport service to show the application layer the content of output data (process data from master to device) and transmits this data to the application layer. The parameters of the service primitives are listed in Table 17.

**Table 17 — DL-PDOutputTransport**

Parameter name	.ind
Argument	M
OutputData	M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**OutputData**

This argument specifies the process data to be transmitted to the application layer.

Parameter type: OctetString

**7.1.7 DL-PDInputUpdate**

The device's application layer uses the DL-PDInputUpdate service to update the input data (process data from device to master) on the data link layer. The parameters of the service primitives are listed in Table 18.

**Table 18 — DL-PDInputUpdate**

Parameter name	.req	.cnf
Argument	M	
InputData	M	
Result (+)		S
TransportStatus		M
Result (-)		S
ErrorInfo		M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**InputData**

This argument specifies the process data provided by the application layer.

Parameter type: OctetString

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**TransportStatus**

This parameter indicates whether the data link layer is in a state permitting data to be transferred to the communication partner(s).

Parameter type: TRANSPORTSTATUS\_TYPE

Permitted values: YES, NO

**Result (-):**

This selection parameter indicates that the service request has failed.

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: NO\_COMM, STATE\_CONFLICT

**7.1.8 DL-PDInputTransport**

The data link layer on the master uses the DL-PDInputTransport service to show the application layer the content of input data (process data from device to master) and transmit this data to the application layer. The parameters of the service primitives are listed in Table 19.

**Table 19 — DL-PDInputTransport**

Parameter name	.ind
Argument	M
InputData	M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**InputData**

This argument specifies the process data to be transmitted to the application layer.

Parameter type: OctetString

**7.1.9 DL-PDCycle**

The data link layer uses the DL-PDCycle service to indicate the end of a process data cycle to the application layer. The parameters of the service primitives are listed in Table 20.

**Table 20 — DL-PDCycle**

Parameter name	.ind
Argument	M

**Argument**

No service-specific parameters associated with the service indication are transmitted in the argument.

**7.1.10 DL-SetMode**

The DL-SetMode service is used to switch the data link layer's state machine. In addition, system management sends the characteristic values required for operation to the data link layer. The parameters of the service primitives are listed in Table 21.

**Table 21 — DL-SetMode**

Parameter name	.req	.cnf
Argument	M	
Mode	M	
ValueList	U	
Result (+)		S
RealMode		M
Result (-)		S
ErrorInfo		M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**Mode**

This parameter specifies the requested mode of the IO-Link master's DL on an individual port.

Parameter type: Enum

Permitted values: Inactive, ComPHY1, ComPHY2, ComStop, DigInput, DigOutput,

**ValueList**

This parameter specifies the relevant operating parameters.

Parameter type: Record

**ParamName**

This record element specifies the name of the parameter to be set.

Parameter type: Unsigned8

Permitted values: MasterCycleTime, FrameType, PdInputLength, PdOutputLength, OnReqDataLengthPerFrame

**MasterCycleTime:** Sent to the startup handler

**FrameType:** Sent to the frame handler

**PdInputLength, PdOutputLength:** Sent to the process data handler

**OnReqDataLengthPerFrame:** Sent to the on-request data handler (important for decoding on-request data (1 byte or 2 bytes))

**Value**

This record element specifies the parameter content.

Parameter type: specific

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**RealMode** (see F.2.3)

**Result (-):**

This selection parameter indicates that the service request has failed.

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: NOT\_CONNECTED, STATE\_CONFLICT

**7.1.11 DL-Mode**

The DL uses the DL-Mode local service to show system management that an operating status has been reached. The parameters of the service primitives are listed in Table 22.

**Table 22 — DL-Mode**

Parameter name	.ind
Argument	M
RealMode	M
ParamList	U

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**RealMode**

Indicates the status of the startup handler.

Parameter type: Unsigned8 (see F.2.4)

Permitted values: Inactive, EstablishComm, InitMaster, InitDevice, Operate, Stop

**ParamList**

This parameter specifies the relevant operating parameters.

Parameter type: Record

**7.1.12 DL-Event**

The Event service indicates a pending status or error message. The source of the event may be local or remote. The event may be triggered by the AL or by an application. The parameters of the service primitives are listed in Table 23.

**Table 23 — DL-Event**

Parameter name	.req	.ind
Argument	M	M
Instance	M	M
Mode	M	M
Type	M	M
PDvalid	M	M
Local	M	M
EventCode	M	M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**Instance**

This parameter specifies the event source.

Permitted values: AL, App, Process, User

**Type**

This parameter specifies the event category.

Permitted values: ERROR, WARNING, MESSAGE

**Mode**

This parameter specifies the event type.

Permitted values: SINGLESHOT, APPEARS, DISAPPEARS

**PDvalid**

This parameter indicates whether, given the conditions of the event that has occurred, the device is going to continue to be able to provide valid process data values.

Permitted values: valid, invalid

**EventCode**

This parameter contains a code identifying the event.

Parameter type: Unsigned16

**Local**

This parameter indicates that the event was generated in the local communication section.

Permitted values: local, remote

**7.1.13 DL-Control**

The Master shall use the Control service to convey control information to the device or to the device application. The parameters of the service primitives are listed in Table 24.

**Table 24 — DL-Control**

Parameter name	.req	.ind
Argument	M	M
ControlCode	M	M(=)

**Argument**

The service-specific parameters are transmitted in the argument.

**ControlCode**

This parameter contains a code identifying the event.

Permitted values: PdOutValid, PdOutInvalid

**7.2 Data link layer protocol****7.2.1 Overview****7.2.1.1 Structure of the data link layer**

Figure 27 and Figure 28 illustrate the structure and interfaces of the data link layer for master and device.

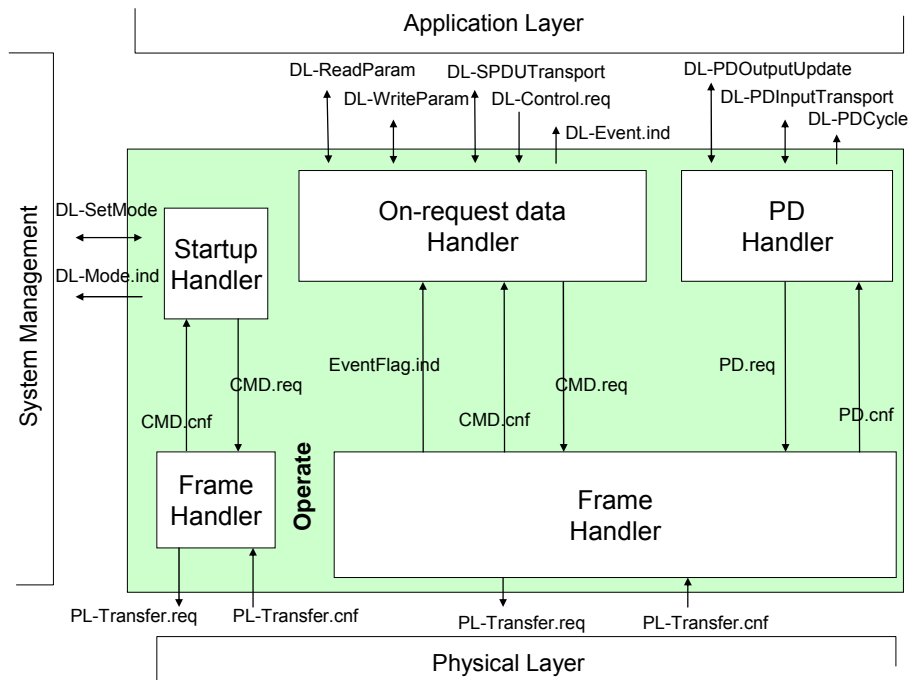


Figure 27 — Structure and interfaces of the data link layer (master)

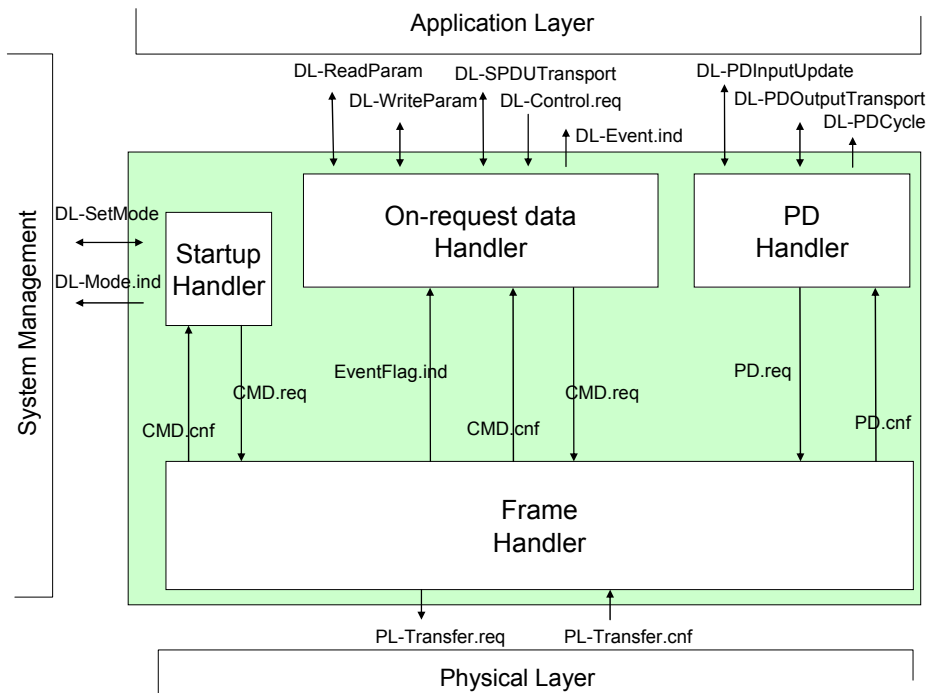


Figure 28 — Structure and interfaces of the data link layer (device)

7.2.1.2 Service primitives on the master data link layer

Table 25 lists the service primitives and their parameters, which are transmitted from the master’s application layer to the master’s data link layer.

**Table 25 — AL service primitives sent to the master's DL**

Primitive name	Source	Associated parameters	Functions
DL-ReadParam.req	AL	Address	see 7.1
DL-WriteParam.req	AL	Address, Value	
DL-SPduTransport.req	AL	SPDU	
DL-PDOutputUpdate.req	AL	OutputData	
DL-Control	AL	ControlCode	

Table 26 lists the service primitives and their parameters, which are transmitted from the master's data link layer to the master's application layer.

**Table 26 — DL service primitives sent to the master's AL**

Primitive name	Source	Associated parameters	Functions
DL-ReadParam.cnf (+)	DL	Value	
DL-ReadParam.cnf (-)	DL	ErrorInfo	
DL-WriteParam.cnf (+)	DL		
DL-WriteParam.cnf (-)	DL	ErrorInfo	
DL-SPduTransport.cnf (-)	DL	ErrorInfo	
DL-SPduTransport.ind	DL	SPDU	
DL-SPduTransport.ind	DL	Status	
DL-PDOutputUpdate.cnf(+)	DL	TransportStatus	
DL-PDOutputUpdate.cnf(-)	DL	ErrorInfo	
DL-PDInputTransport.ind	DL	InputData	
DL-Event.ind	DL	Instance, Mode, Type, PDValid, Local, EventCode	
DL-PDCycle.ind	DL		

Table 27 lists the service primitives and their parameters, which are transmitted from the master's system management to the master's data link layer.

**Table 27 — SM service primitives sent to the master's DL**

Primitive name	Source	Associated parameters	Functions
DL-SetMode.req	AL	TargetMode, ValueList	see 7.1

Table 28 lists the service primitives and their parameters, which are transmitted from the master's system management to the master's data link layer.



**Table 28 — DL service primitives sent to the master's SM**

Primitive name	Source	Associated parameters	Functions
DL-SetMode.cnf (+)	DL	RealMode	see 7.1
DL-SetMode.cnf (-)	DL	ErrorInfo	
DL-Mode.ind	DL	RealMode, ParamList	

### 7.2.1.3 Local variables on the master's data link layer

The master's data link layer contains the local variables listed in Table 29.

**Table 29 — Local variables on the master's DL**

Variable Name	Description
DL_nRetry	Counter for retries affecting service calls to the physical layer.
DL_ReadCommParamStatus	Status variable, which controls the processing of the parameter list for reading out parameter values from the device. Permissible values: UNDONE, DOING, DONE.
DL_WriteCommParamStatus	Status variable, which controls the processing of the parameter list for writing parameter values to the device. Permissible values: UNDONE, DOING, DONE.

## 7.2.2 Startup handler

Startup is divided into a phase during which the master is initialized and a phase during which the device is initialized. The parameters for communication are defined for both. Startup is initiated by the master.

The startup handler is controlled by system management. The application charges system management with controlling startup.

### 7.2.2.1 Master startup

#### 7.2.2.1.1 Overview

The master DL state machine is illustrated in Figure 29. Startup of the master is initiated from SM using DL-SetMode.

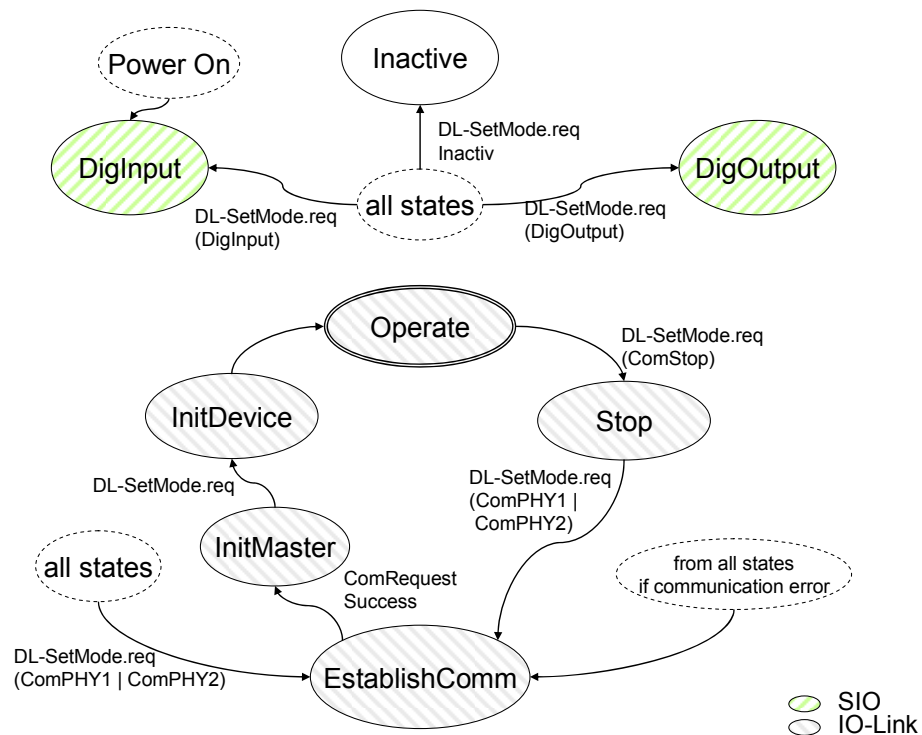


Figure 29 — Master DL state diagram

7.2.2.1.2 General description of states

The states defined for the master’s data link layer are summarized in Table 30.

Table 30 — Master DL states

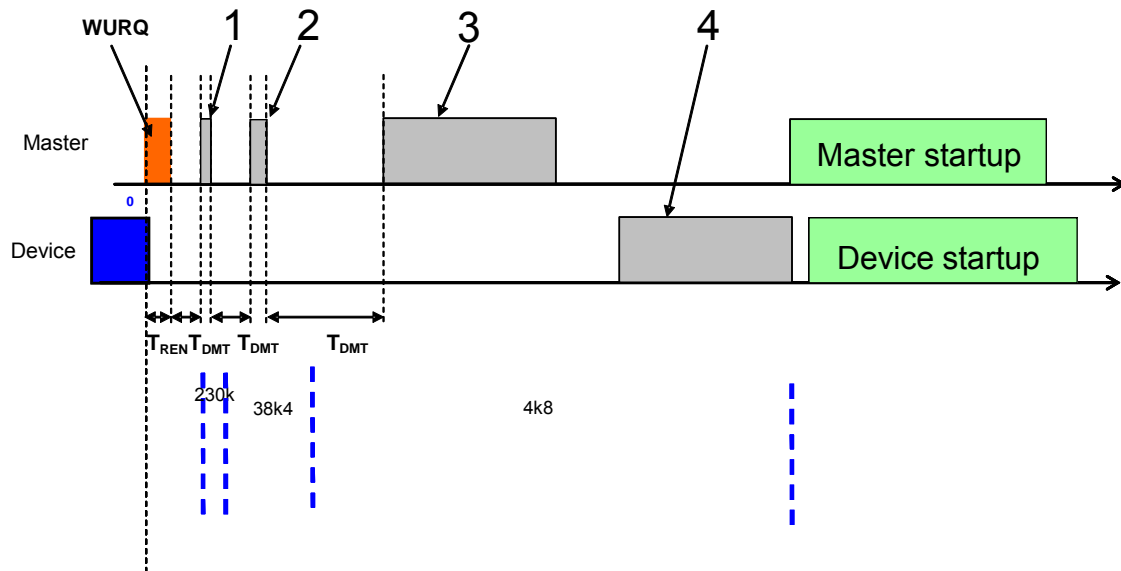
State Name	Description
INACTIVE	In the INACTIVE state, the port is disabled and not considered to be a process data bit. The “Inactive” state is exited when a DL-SetMode service is received with Mode=ComPHY1 or Mode=ComPHY2.
DIGINPUT	In the DIGINPUT state, the port operates as a standard input and is, therefore, considered to be a process data bit. The DIGINPUT state is exited when a DL-SetMode service is received with Mode=ComPHY1 or Mode=ComPHY2.
DIGOUTPUT	In the DIGOUTPUT state, the port operates as a standard output and is, therefore, considered to be a process data bit. The DIGOUTPUT state is exited when a DL-SetMode service is received with Mode=ComPHY1 or Mode=ComPHY2.
ESTABLISHCOMM	In the ESTABLISHCOMM state, communication is initiated by the master sending a wake-up request and then implemented by executing a Type 0 Read frame (parameter=Min Cycle Time).
INITMASTER	In the INITMASTER state, the master reads communication parameters from the device.
INITDEVICE	In the INITDEVICE state, the master writes communication parameters to the device.
OPERATE	In the OPERATE state, the on-request and process data handlers are activated in order for data exchange to take place. Start of cyclic operation.  When the DL-SetMode (DIGINPUT or DIGOUTPUT or INACTIVE) state takes effect, the CMD fallback is transmitted to the device and the state changes.  If the on-request and process data handlers are terminated due to errors, the state machine shall switch to the ESTABLISHCOMM state.
STOP	In the STOP state, writing master command “Device startup” to address 0 shall force a changeover from OPERATE to ESTABLISHCOMM.  This command shall stop the on-request and process data handlers. Any open services shall be aborted.

### 7.2.2.1.3 Description of the “EstablishComm” state

A WakeUp triggers the following actions on the data link layer.

The master shall attempt to establish communication by sending a wake-up request and then executing a Type 0 Read frame (parameter = Min Cycle Time).

The master follows the sequence illustrated in Figure 30 when attempting to establish communication.



**Figure 30 — Successful establishment of communication**

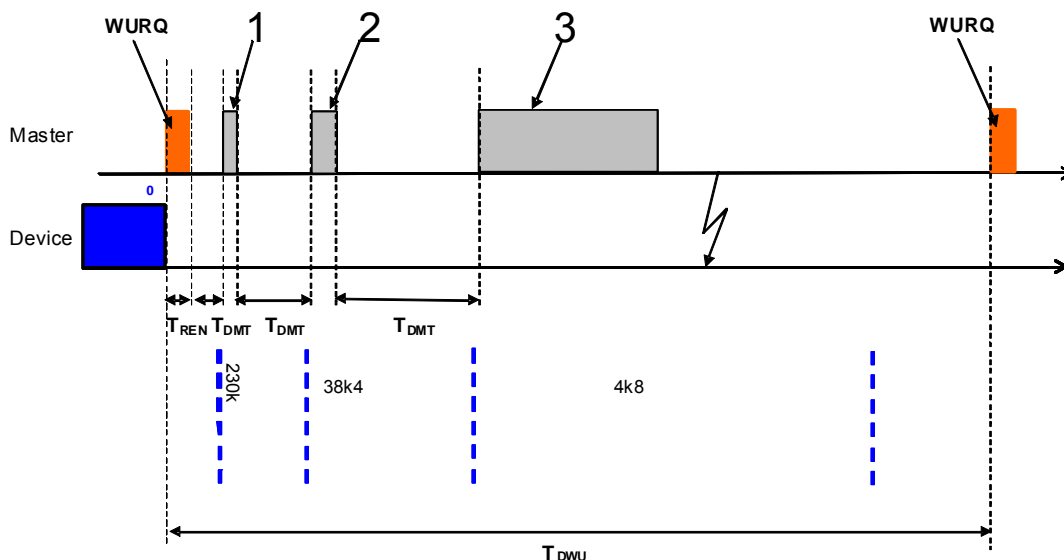
Establishment of communication takes in the following states once the wake-up request has been sent:

- 1) Master telegram in COM3 (optional)
- 2) Master telegram in COM2
- 3) Master telegram in COM1
- 4) Response from device in COM1

Depending on the connected device, a valid response may be even be received after COM3 or COM2. The sequence of states from the higher to lower baud rates supported by the master is mandatory. If the attempt succeeds (i.e., if a valid response is received), startup shall commence.

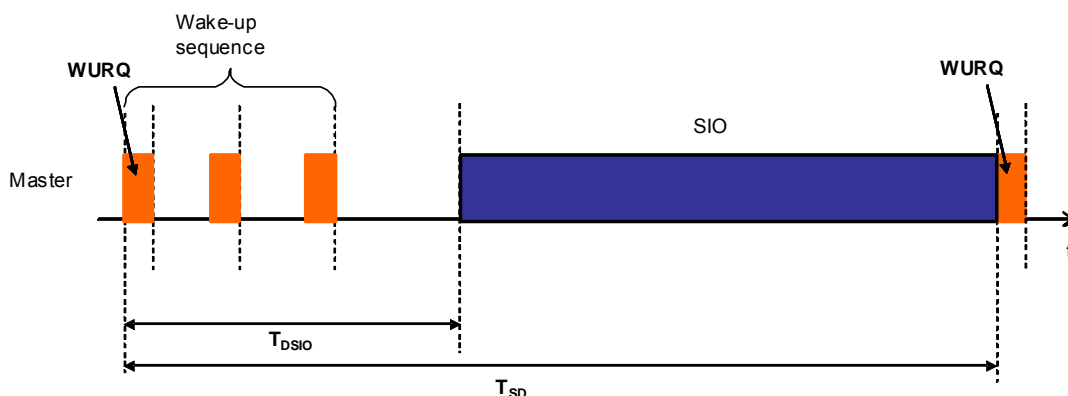
Following the wake-up request and/or establishment of communication, the master shall wait a time of  $T_{DMT}$  of the next baud rate before switching to the next baud rate.

Once time  $T_{DWU}$  has elapsed, the master may send a new wake-up request as shown in Figure 31. The wake-up sequence comes to an end after  $n_{wu}+1$  attempts.



**Figure 31 — Failed attempt to establish communication**

Once time  $T_{DSIO}$  has elapsed, the device may reset its C/Q to SIO mode and once time  $T_{SD}$  has elapsed, a new wake-up sequence shall begin, see Figure 32.



**Figure 32 — Communication establishment retries**

The master shall behave like a standard input during the time between failed retries and a new device detect.

In addition to the definitions in Table 5, the definitions in Table 31 are valid for the wake-up procedure and its parameters. All times are defined from the point of view of the master.

**Table 31 — WakeUp procedure and level definitions**

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
$T_{DMT}$	Master telegram delay	27	n/a	37	$T_{BIT}$	Bit time of subsequent data rate
$T_{DSIO}$	Standard IO delay	60	n/a	300	ms	After $T_{DSIO}$ the device falls back to switching mode (if supported)

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
T <sub>DWU</sub>	Wake-up retry delay	30	n/a	50	ms	After TDWU the master resends the wake-up request
n <sub>WU</sub>	Wake-up retry count	2	2	2		Number of wake-up retries
T <sub>SD</sub>	Device detect time	0,5	n/a	1	s	Time between 2 wake-up sequences Configuration characteristic of the IO-Link master

The master's data link layer shall exit the wake-up procedure once it finds a communicating IO-Link device. At this point, a DL-SetMode confirmation indicating the mode identified is sent as feedback .

If communication cannot be established after n<sub>WU</sub> retries, the negative confirmation "NOT\_CONNECTED" is sent back to SM.

#### 7.2.2.1.4 Description of the InitMaster state

The master shall use Type 0 frames to read the device data required for device startup. This transfer shall take place with direct access to the parameter page. The time between the start of two frames shall be T<sub>initcyc</sub> as listed in Table 32.

**Table 32 — Startup sequence**

Parameter	Designation	Minimum	Typical	Maximum	Unit	Remark
T <sub>INITCYC</sub>	Startup cycle time applying for PHY2	100	n/a	n/a	T <sub>BIT</sub>	Bit time of data rate
T <sub>INITCYC</sub>	Startup cycle time applying for PHY1	1000	n/a	n/a	T <sub>BIT</sub>	Bit time of data rate

The startup parameters in this phase are listed in Table 33. Addresses 2 to 6 shall be read in ascending order.

**Table 33 — StartUp parameters in InitMaster**

Address	Parameter name
02h	Min Cycle Time
03h	Frame Capability
04h	IO-Link Revision ID
05h	Process Data In
06h	Process Data Out

#### 7.2.2.1.5 Description of the InitDevice state

The startup parameters in this phase are listed in Table 34.

**Table 34 — StartUp parameters in InitDevice**

Address	Parameter name
00h	Master Command
01h	Master Cycle Time

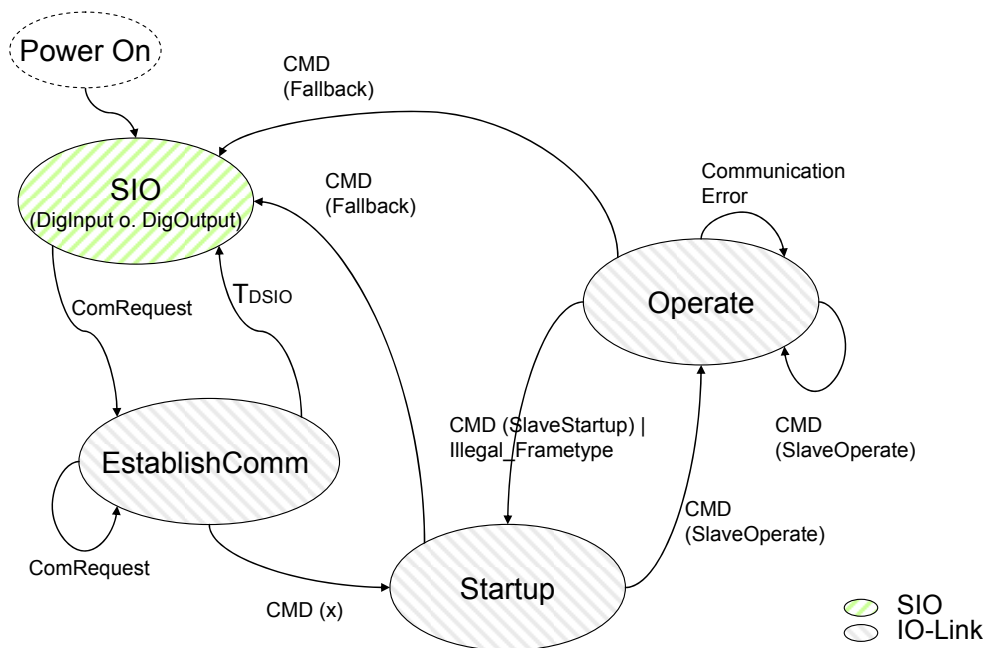
The master starts by writing address 1 in the direct parameter page, followed by address 0. The time between the start of two frames shall be  $T_{initcyc}$ .

InitDevice ends with the writing of the master command “DeviceOperate” to address 0. This induces a transition to the Operate state.

**7.2.2.2 Device startup**

**7.2.2.2.1 Overview**

The device DL state machine is illustrated in Figure 33. Once the device power supply has been connected, if Standard IO is supported, the device shall switch to Standard IO mode.



**Figure 33 — Device DL state diagram**

**7.2.2.2.2 General description of states**

The states defined for the device’s data link layer are summarized in Table 35.

**Table 35 — Device DL states**

State Name	Description
SIO	In the IO state, the port operates as a standard switching signal and is considered to be a process data bit.

State Name	Description
ESTABLISHCOMM	The device switches to the ESTABLISHCOMM state when a wake-up request is received. If a frame is then executed successfully, the device shall switch to the STARTUP state. Otherwise, once the monitoring time TDSIO has elapsed, it shall switch back to the SIO state.
STARTUP	Data exchange takes place in the STARTUP state. The process data is still not active at this point. During device startup, the time between the start of two frames may be $T_{initcyc}$ . The device uses the DeviceOperate command to initialize its communication with the set parameters.
OPERATE	In the OPERATE state, the on-request and process data handlers are activated in order for data exchange to take place. The device switches back to the SIO state when a master command with FALLBACK is received. The device switches to the STARTUP state when a master command with DEVICESTARTUP or an incorrect frame type is received. The on-request and process data handlers are stopped and any open services shall be aborted. On receiving a DEVICEOPERATE master command the device shall reattempt to initialize communication with the set parameters.

### 7.2.2.2.3 Description of startup

On completion of startup a device is able to communicate. From this point onwards, cyclic frames shall be generated on the data link layer. Initially, only the on-request data channel is available.

Devices, which fall back to SIO mode, may be reset to the SIO state from the parameterization phase by sending a command as a final transmission operation.

Cyclic process data communication then may be started via the application on the master. A device shall be addressed from the next cycle onwards with the cyclic mode frame type.

The master shall follow the MasterCycleTime after sending the DeviceOperate command for the first time. The accuracy of the MasterCycleTime shall be from 0% to +10%.

### 7.2.2.3 Elements of the parameter data channel

The parameter data channel comprises 32 byte type parameters, which, depending on their addresses, may be read and write or read-only parameters. The parameters and their descriptions appear in Table 36.

**Table 36 — Addresses on the parameter data channel**

Address	Parameter name	Access	Implementation	Description
00h	Master Command	R/W	mandatory	Master command to set operating states. NOTE 1
01h	Master Cycle Time	R/W	mandatory	Actual cycle duration used by the master to address the device. May be used as a parameter to monitor process data transfer.
02h	Min Cycle Time	R	mandatory	Minimum cycle duration supported by a device. This is a performance feature of the device and is determined by its device function.
03h	Frame Capability	R	mandatory	Information about implemented options in relation to frames and physical configuration
04h	IO-Link Revision ID	R	mandatory	Version ID for the implemented IO-Link specification (shall be set to 10 for this version)
05h	Process Data In	R	mandatory	Number and structure of input data (process data from device to master)

Address	Parameter name	Access	Implementation	Description
06h	Process Data Out	R	mandatory	Number and structure of output data (process data from master device master)
07h	Vendor ID 1	R	mandatory	Unique vendor ID (defined by IO-Link consortium)
08h	Vendor ID 2	R	mandatory	
09h	Device ID 1	R	mandatory	Unique device ID allocated by a vendor
0Ah	Device ID 2	R	mandatory	
0Bh	Device ID 3	R	mandatory	
0Ch	Function ID 1	R/W	optional	Vendor specific unique ID for each function type 0000h : not used, no validation 0001h .. 7FFFh : reserved 8000h .. FFFFh : vendor specific
0Dh	Function ID 2	R/W	optional	
0Eh .. 0Fh		n/a	n/a	Reserved for future use
10 .. 1Bh		n/a	optional	Device-specific parameter
1Ch	Offset Time	R/W	optional	Time between start of cycle and point in time when process data is processed on the device. May be used to synchronize/desynchronize multiple ports.
1D .. 1Fh		n/a	n/a	Reserved for future use
NOTE 1 The value of Master Command need not to be available for read access.				

A device shall reply the value 0 upon read access to parameter addresses, which are not implemented (i.e. reserved parameter addresses or not supported optional parameters).

### 7.2.2.3.1 Master Command

Permissible values for the Master Command parameter are listed in Table 37.

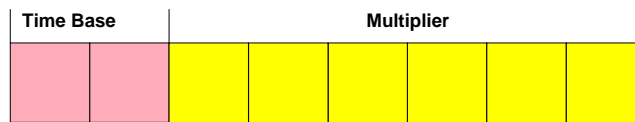
**Table 37 — Values of Master Command**

Value	Master command	Description
00h .. 59h	reserved	
5Ah	FALLBACK	Transition from communication to SIO mode. The device shall execute this transition after 3 and before 4 master cycle times.
5Bh .. 96h	reserved	
97h	DeviceStartup	Switches the device from Operate to Startup
98h	Process data Output Operate,	Process output data valid
99h	DeviceOperate	Process output data invalid or not available
9Ah .. 9Fh	reserved	
A0h .. FFh	Vendor-specific	

### 7.2.2.3.2 Min Cycle Time

The structure of the Min Cycle Time parameter is illustrated in Figure 34.





**Figure 34 — Min Cycle Time**

#### Bits 0 to 5: Multiplier

These bits contain a 6-bit factor for calculating the Min Cycle Time. Permissible values for Multiplier are 0 to 63.

#### Bits 6 to 7: Time Base

These bits contain the time base for calculating the Min Cycle Time.

The permissible combinations for Time Base and Multiplier are listed in Table 38, along with the resulting values for Min Cycle Time.

If a value of 00h is assigned to this byte, the device will not be restricted in respect of cycle time (device has no Min Cycle Time).

**Table 38 — Values of Min Cycle Time**

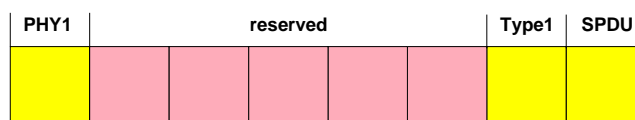
Time Base	Meaning for Time Base	Calculation	Min Cycle Time
00b	0,100 ms	Multiplier * Time Base	0,000 .. 6,300 ms
01b	0,400 ms	6,400 ms + Multiplier * Time Base	6,400 .. 31,600 ms
10b	1,600 ms	32,000 ms + Multiplier * Time Base	32,00 .. 132,800 ms
11b	6,400 ms	134,400 ms + Multiplier * Time Base	134,400 .. 537,600 ms

#### 7.2.2.3.3 Master Cycle Time

See 7.2.2.3.2.

#### 7.2.2.3.4 Frame Capability

The structure of the Frame Capability parameter is illustrated in Figure 35.



**Figure 35 — Frame Capability**

#### Bit 0: SPDU

This bit indicates whether or not the Service PDU data channel is supported. Permissible values for SPDU are listed in Table 39.

**Table 39 — Values of SPDU**

Value	Meaning
0	Service PDU not supported
1	Service PDU supported

**Bit 1: Type1**

This bit indicates whether frame type 1 is supported for on-request data on the data link layer. This is also valid for exclusive on-request data exchange with a process data length of 0. Permissible values for Type1 are listed in Table 40.

**Table 40 — Values of Type1**

Value	Meaning
0	Type 0 only supported
1	Type 1 supported

**Bits 2 to 6: reserved**

These bits are reserved and shall be set to zero in this IO-Link version.

**Bit 7: PHY1**

This bit indicates which variant of the Medium Attachment Unit is supported on the physical layer. Permissible values for Type1 are listed in Table 41.

**Table 41 — Values of PHY1**

Value	Meaning
0	PHY2 supported
1	PHY1 supported

**7.2.2.3.5 IO-Link Revision ID**

The IO-Link Revision ID parameter is the two-digit version number of the IO-Link specification in accordance with which the device has been implemented. Its structure is illustrated in Figure 36.

**Figure 36 — IO-Link Revision ID****Bits 0 to 3: MinorRev**

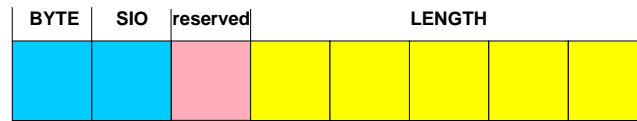
These bits contain the minor digit of the version number, e.g., 0 for IO-Link Version 1.0. Permissible values for MinorRev are 0 to 9.

**Bits 4 to 7: MajorRev**

These bits contain the major digit of the version number, e.g., 1 for IO-Link Version 1.0. Permissible values for MajorRev are 0 to 9.

### 7.2.2.3.6 Process Data In

The structure of the Process Data In parameter is illustrated in Figure 37.



**Figure 37 — Process Data In**

#### Bits 0 to 4: LENGTH

This bit contains the length of the input data (process data from device to master) in the length unit designated in the Byte parameter. Permissible values for LENGTH are 0 to 31, the meaning of LENGTH depends on BYTE and is defined in Table 43.

#### Bit 5: reserved

This bit is reserved and shall be set to zero in this IO-Link version.

#### Bit 6: SIO

This bit indicates whether the device has a switching signal in Standard IO mode. Permissible values for SIO are listed in Table 42.

**Table 42 — Values of SIO**

Value	Meaning
0	SIO mode not supported
1	SIO mode supported

#### Bit 7: BYTE

This bit indicates the length unit for Length. Permissible values for BYTE and the resulting meaning of the proces data length in conjunction with LENGTH are listed in Table 43.

**Table 43 — Values of BYTE**

BYTE	LENGTH	Meaning
0	0	no process data (NOTE)
0	1	1 bit process data, structured in bits
0	...	...
0	16	16 bit process data, structured in bits
0	17 .. 31	n/a
1	0,1	n/a
1	2	3 byte process data, structured in bytes
1	...	
1	31	32 byte process data, structured in bytes

NOTE: no process data is useful with SIO=1 only.

### 7.2.2.3.7 Process Data Out

The structure of the Process Data Out is the same as specified in 7.2.2.3.6, with the only exception, that BIT 6 is reserved.

### 7.2.2.3.8 Vendor ID 1, 2

These bytes contain the vendor ID. The vendor ID is regulated by the IO-Link consortium and shall be unique all over the world.

### 7.2.2.3.9 Device ID 1, 2, 3

These bytes contain the device-type ID. The device ID shall be unique for all device types manufactured by a single vendor.

### 7.2.2.3.10 Offset Time

The structure of the Offset Time parameter is the same as illustrated in Figure 34.

#### Bits 0 to 5: Multiplier

These bits contain a 6-bit factor for calculating the Offset Time.

#### Bits 6 to 7: Time Base

These bits contain the time base for calculating the Offset Time.

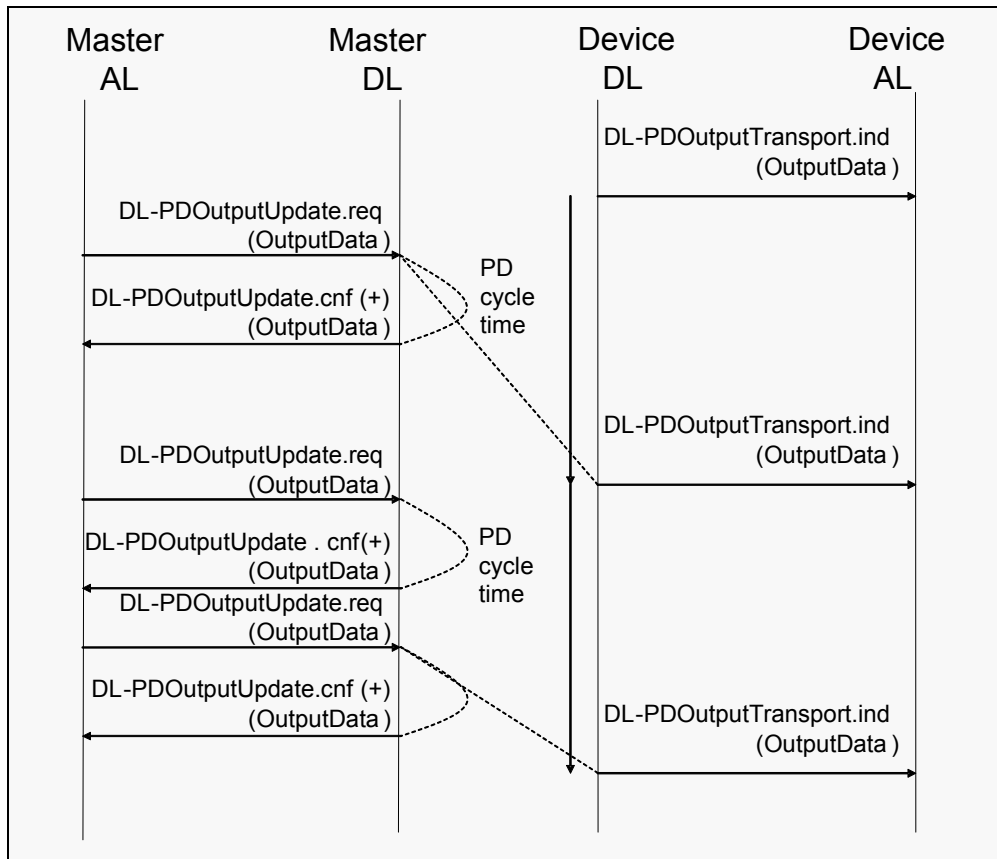
The permissible combinations for Time Base and Multiplier are listed in Table 44, along with the resulting values for Min Cycle Time. Setting both Multiplier and Time Base to zero is a special case where synchronization is deactivated by means of Offset Time.

**Table 44 — Values of Offset Time**

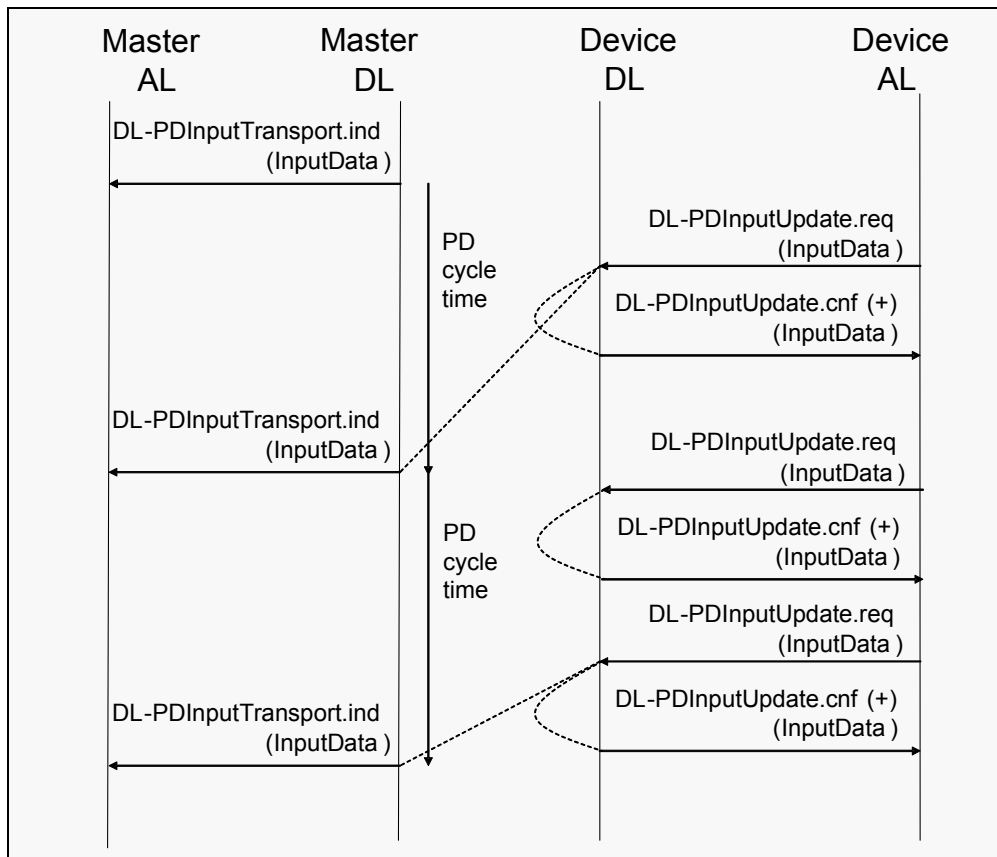
Time Base	Meaning of Time Base	Calculation	Offset Time
00b	0,010 ms	Multiplier * Time Base	0,010 .. 0,630 ms (NOTE)
01b	0,040 ms	0,640 ms + Multiplier * Time Base	0,640 .. 3,160 ms
10b	0,640 ms	3,200 ms + Multiplier * Time Base	3,200 .. 43,520 ms
11b	10,240 ms	44,160 ms + Multiplier * Time Base	44,160 .. 689,280 ms
NOTE A value of 0 indicates no synchronization.			

## 7.2.3 Process data handler

The sequences for transporting process data for output data using the DL-OutputUpdate and DL-OutputTransport services are illustrated in Figure 38 and for input data using the DL-InputUpdate and DL-InputTransport services in Figure 39.



**Figure 38 — Output data transport sequence**



**Figure 39 — Input Data Transport Sequence**

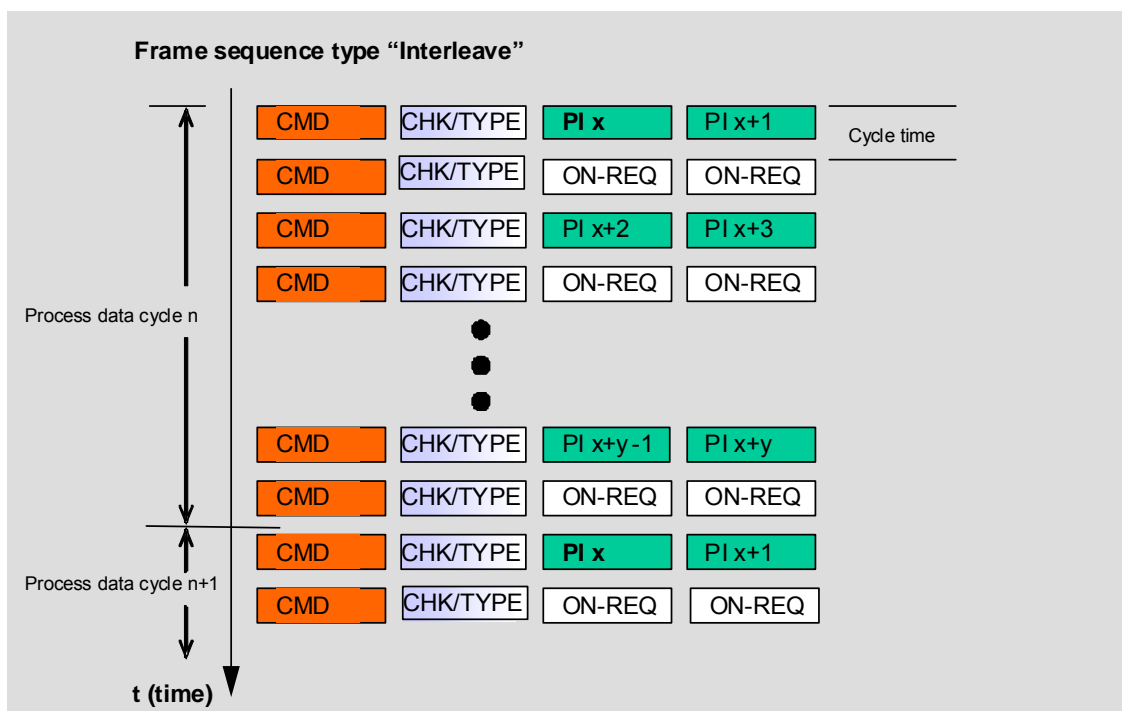
A process data cycle is complete when all process data has been transferred between master and device. A cycle may last for several frames.

For frame types 2.1 to 2.5, all process data is transmitted within one frame.

NOTE: The temporal length of the process data cycle is in this case the same as the cycle time.

For frame type 1, process data and service data is transmitted alternately in multiple frames. Figure 40 illustrates this from the point of view of the master call for a device with output data. Within a process data cycle, in the process data, all input data shall be read first and then all output data shall be written.

The length of the process data cycle is the sum of all cycle times (see 7.2.5.5) required to transmit all process data.



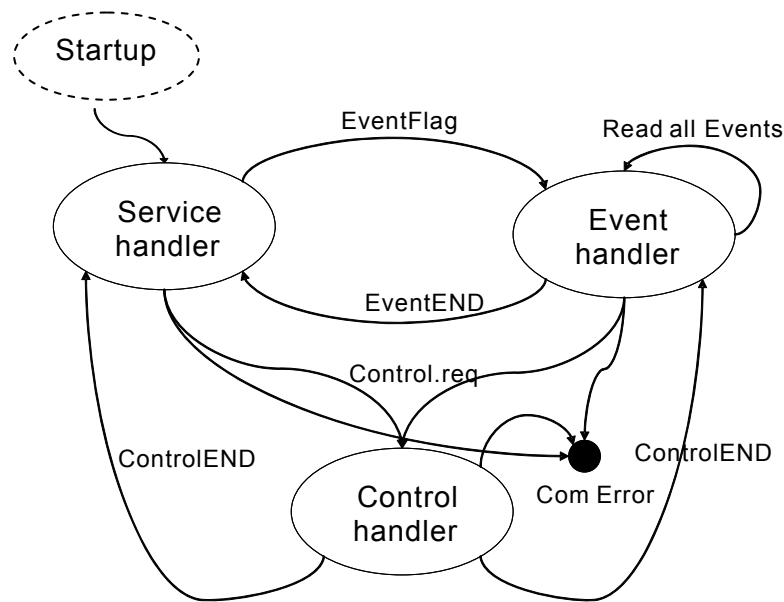
**Figure 40 — Interleave mode sequence**

NOTE The temporal length of the process data cycle is in this case a multiple of the cycle time.

## 7.2.4 On-request data handler

### 7.2.4.1 On-request data handler on the master data link layer

The state machine of the master data link layer's on-request data handler is illustrated in Figure 41 (see F.2.5).



**Figure 41 — On-request handler on the master**

The on-request data handler is a subordinate state machine of the Operate state. It starts in the service handler.

If an EventFlag.ind is received from the frame handler, the state machine shall switch to the event handler. Once event handling is complete, it reverts to the service handler.

If a Control.req is received in the service handler or the event handler, the state machine shall switch to the control handler. Once the control request has been handled, it shall revert to the previously active state.

#### 7.2.4.2 On-request data handler on the device data link layer

The device decodes CMD.ind and derives from it the data channel and memory address. The data channels are entirely independent.

In the event of valid access, the corresponding state machine is addressed on the relevant data channel.

The device shall respond to requests for read access to non-implemented ranges by sending back the value zero. It shall ignore requests for write access to non-implemented addresses.

Requests for access sent in frame types without addressing on the process data channel shall also be ignored.

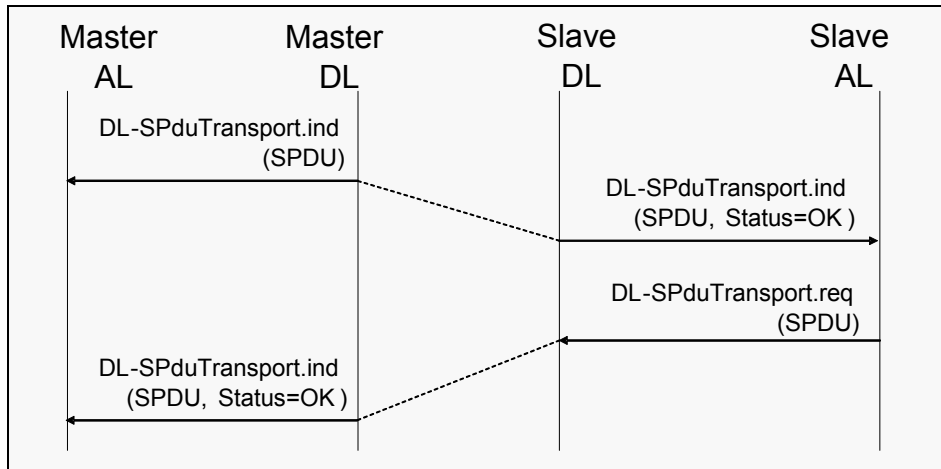
Where non-implemented SPDUs on the device are concerned, the device shall send NO\_SERVICE in response to CMD.ind (R, SPDU, FlowCTRL=IDLE).

An error message is not generated.

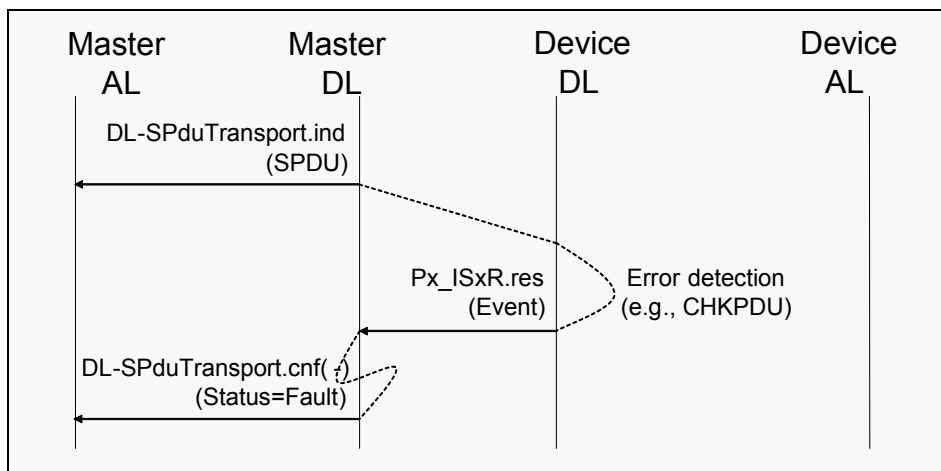
NOTE CMD.ind (R, SPDU, FlowCTRL=IDLE) is the default telegram if there are no on-request data pending for transmission.

**7.2.4.3 Service handler**

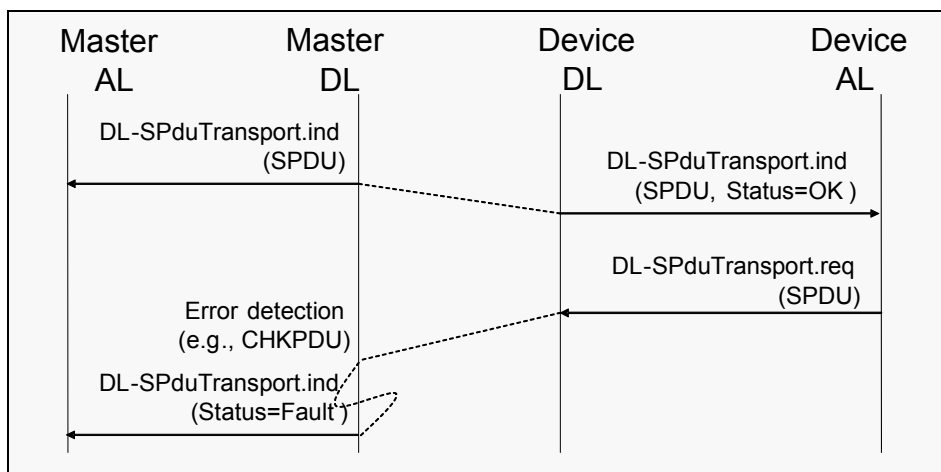
Error-free sequences for the transmission of Service PDUs using the DL-SPduTransport service are illustrated in Figure 42. Figure 43 and Figure 44 illustrate the sequences for error detection on the master and device (see F.2.6).



**Figure 42 — Error-free Service PDU transport sequence**



**Figure 43 — Service PDU transport sequence with error on device**



**Figure 44 — Service PDU transport sequence on master**



7.2.4.3.1 Service handler on the master

The state machine of the service handler on the master is illustrated in Figure 45.

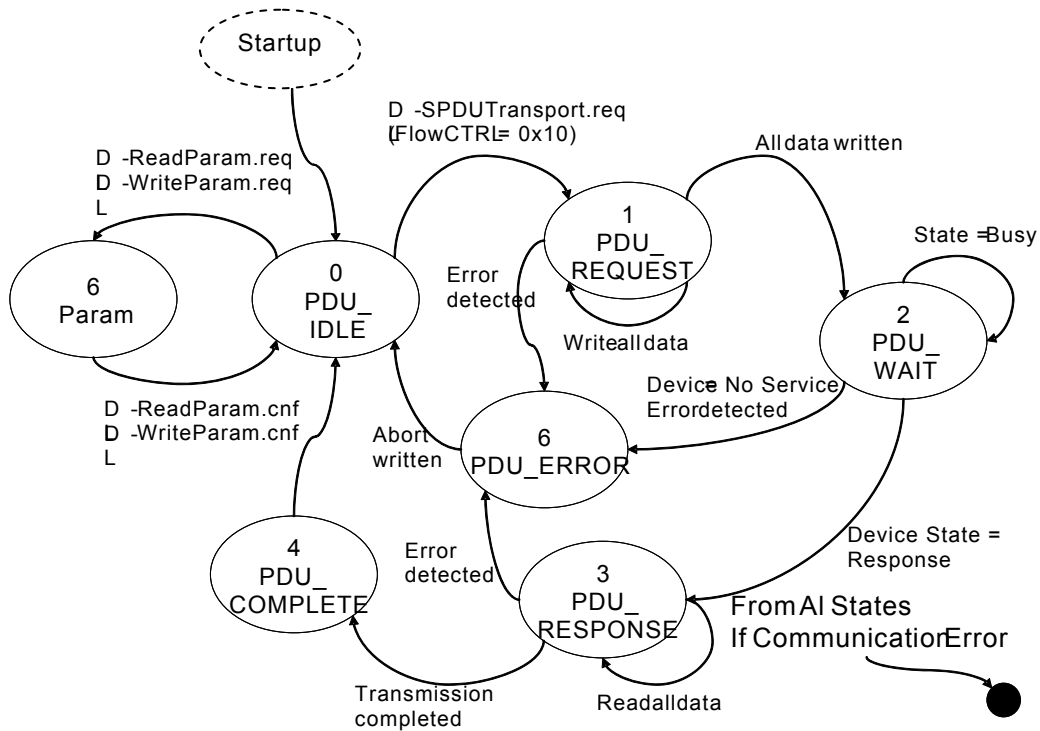


Figure 45 — Service handler on the master

The states defined for the service handler on the master are summarized in Table 45 (F.2.7).

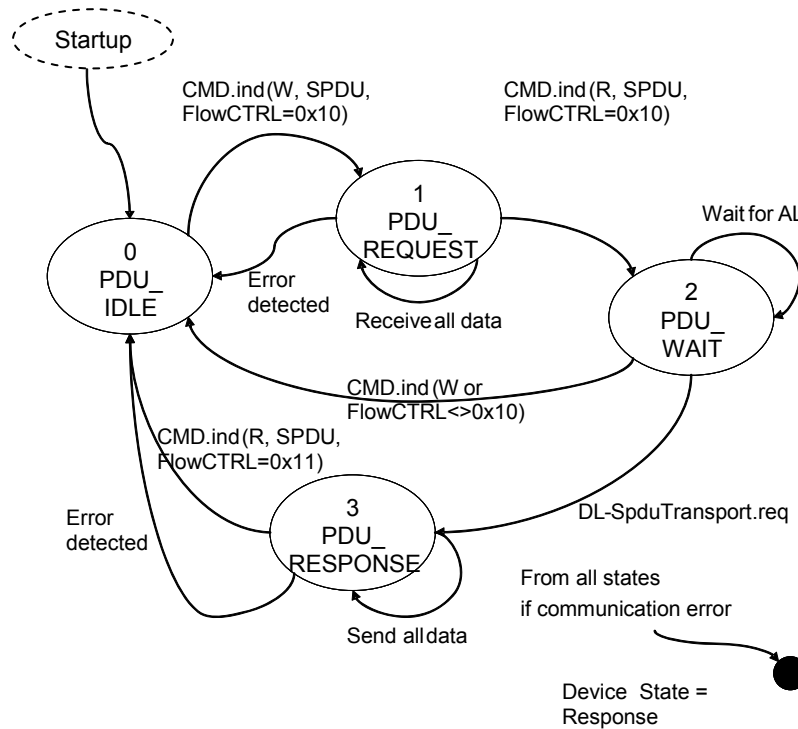
Table 45 — Master DL states

State name	Description
PDU_IDLE	If a DL-ReadParam.req or DL-WriteParam.req is sent, go to PARAM. If a DL-SPDUTransport.req is sent, go to PDU_REQUEST. Otherwise remain in PDU_IDLE and generate CMD.req (R,SPDU, FlowCTRL=IDLE) (NOTE 1).
PDU_REQUEST	Master starts with CMD.req (W,SPDU, FlowCTRL=START, data) and repeats this call with active Flow Control and continuous data until all data has been transmitted to the device. If the master data link layer detects an error in the SPDU transmission, it shall switch to PDU_ERROR. (NOTE 2)
PDU_WAIT	Master generates CMD.req (R,SPDU, FlowCTRL=START) and evaluates CMD.cnf. If CMD.cnf(+) == Busy (0x01), remain in PDU_WAIT If CMD.cnf(+) == Service_Response, go to PDU_RESPONSE If CMD.cnf(+) == NO_SERVICE, generate DL-SPDUTransport.ind(-) and go to PDU_ERROR If the master data link layer detects an error in the SPDU transmission, it shall switch to PDU_ERROR. (NOTE 2)

State name	Description
PDU_RESPONSE (see F.2.7)	<p>Master starts with CMD.req (R,SPDU, FlowCTRL=COUNT, Data) and repeats this call with active Flow Control and continuous data until all data including CHKPDU has been transmitted to the master.</p> <p>Once all data has been received, it checks CHKPDU.</p> <p>Otherwise, it switches to PDU_ERROR.</p> <p>If the master data link layer detects an error in the SPDU transmission, it shall switch to PDU_ERROR.</p> <p>(NOTE 2)</p>
PDU_COMPLETE	<p>Master sends CMD.req (R,SPDU, FlowCTRL=IDLE).</p> <p>If the device responds without a set event flag, the master shall generate a DL-SPDUTransport.ind(+).</p> <p>Otherwise, the master shall generate a DL-SPDUTransport.ind(-).</p> <p>It shall switch to PDU_IDLE afterwards.</p> <p>(NOTE 2)</p>
PDU_ERROR	<p>The master sends CMD.req (W, SPDU, FlowCTRL=ABORT) once to the device to prevent the device generating an event even though the master has detected an error during processing and switches to the output state.</p>
PARAM	<p>If DL-ReadParam.req(Address)</p> <p style="padding-left: 40px;">Generate CMD.req (R,Parameter,Address)</p> <p style="padding-left: 40px;">Generate DL-ReadParam.cnf on the basis of CMD.cnf</p> <p>If DL-WriteParam.req(Address,Value)</p> <p style="padding-left: 40px;">Generate CMD.req (W, Parameter, Address, Value)</p> <p style="padding-left: 40px;">Generate DL-WriteParam.cnf on the basis of CMD.cnf</p> <p>Go to PDU_IDLE.</p> <p>(NOTE 2)</p>
NOTE 1	<p>A CMD.req (R,SPDU, FlowCTRL=IDLE) is a default telegram is no on-request data is transmitted (see F.2.7).</p> <p>NOTE 2 If, in a state other than PDU_IDLE, an event indicates that a DL error has occurred, a negative confirmation shall be generated for the service in progress and the device shall switch to PDU_IDLE.</p>

#### 7.2.4.3.2 Service handler on the device data link layer

The state machine of the service handler on the device is illustrated in Figure 46.



**Figure 46 — Service handler on the device**

The states defined for the service handler on the device are summarized in Table 46.

**Table 46 — Device DL states**

State name	Description
PDU_IDLE	Following startup, the service handler starts in the PDU_IDLE state. Events are handled in this state.  When the master starts SPDU transmission, i.e., when a CMD.ind (W, SPDU, FlowCTRL=START) is received, the device switches to the PDU_REQUEST state.  The device always supports a CMD.req (R,SPDU, FlowCTRL=IDLE), even if it does not support any Service PDUs. It responds with CMD.res (NO_SERVICE).
PDU_REQUEST	Service PDU data from the master is received in the PDU_REQUEST state.  The device service handler detects the end using a CMD.ind (R, SPDU, FlowCTRL=START). At this point, it switches to the PDU_WAIT state.
PDU_WAIT	In the PDU_WAIT state, the device service handler waits for SPDU data to be made available on the application layer.  If a DL-SpduTransport.req is received from the application layer, the device service handler shall switch to the PDU_RESPONSE state.  When a CMD.ind (W, ...) or (... , FlowCTRL<>START) is received, the device service handler detects an abort/restart of the master and switches to the PDU_IDLE state. The DL sends an Abort.ind to the AL. For all CMD.ind except (W, FlowCTRL=ABORT), it also sends an Event.ind (SINGLESLOT, ERROR, DL, S_PDU_FLOW)
PDU_RESPONSE	Service PDU data is sent to the master in the PDU_RESPONSE state.  The device service handler detects the end using a CMD.ind (R, SPDU, FlowCTRL=IDLE). At this point, it switches to the PDU_IDLE state.
NOTE If, in a state other than PDU_IDLE, an event indicates that a DL error has occurred, a negative confirmation shall be generated for the service in progress and the device shall switch to PDU_IDLE.	

**7.2.4.3.3 Transmission of Service PDUs**

A Service PDU is transmitted via the “Service PDU” on-request data channel. A number of telegrams are typically required for this purpose. The master transfers a Service PDU by sending

a service request to the device via the “Service PDU” on-request data channel. It then fetches the device’s response via the “Service PDU” on-request data channel.

NOTE The mapping of Service PDU transmission to the DL\_Command and CHK/TYPE bytes is illustrated in Annex B (see F.2.8).

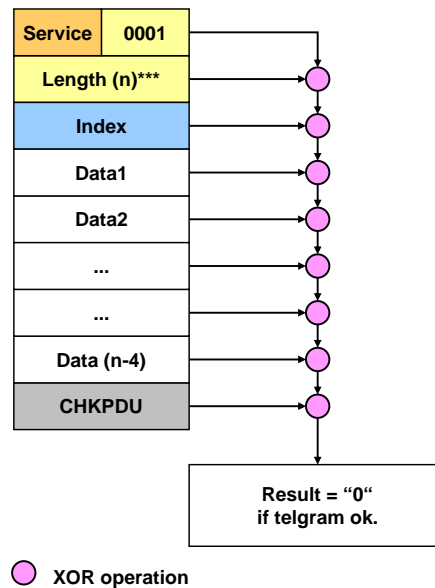
In the “Service PDU” on-request data channel, on the data link layer, the address controls data flow (referred to henceforth as FlowCTRL). The 4 low-order bits of the 5-bit-wide parameter count the frames “rolling” with overflow. The master uses FlowCTRL in conjunction with the elements of the Service PDU (e.g., the Length element) to control the data flow of a service. Permissible values for FlowCTRL are defined in Table 47.

**Table 47 — Definition for FlowCTRL**

FlowCTRL	Meaning
0x00 .. 0x0F	COUNT Frame counter within a PDU. Increments beginning with 1 following start with START. Jumps back from 15 to 0 in the event of an overflow.
0x10	START Start of a Service PDU, i.e., start of a request or a response. For the start of a request, any previously incomplete services may be rejected. For a start request associated with a response, a device shall send “No Service” until its application returns response data.
0x11	IDLE No request for Service PDU transmission.
0x12 .. 0x1E	reserved
0x1F	ABORT Abort entire service. The master responds by rejecting received response data. The device responds by rejecting received request data and may generate an abort.

#### 7.2.4.3.4 Data integrity of Service PDU transmission

The “CHKPDU” element provides data integrity protection. CHKPDU is generated by exclusively ORing the bytes in a Service PDU, see Figure 47 (see F.2.9). This means that all bytes in the Service PDU, including the exclusively-ORed CHKPDU, produce a value of 0.



**Figure 47 — Calculation of CHKPDU**

#### 7.2.4.4 Event handler

##### 7.2.4.4.1 The event transmission sequence

The device DL writes an event to the event buffer and then sets the event flag, which is sent to the master in the next frame in the CHECK/STAT byte.

There are two event-coding methods, with or without detailed information. A device always uses one of these two methods. For the purpose of differentiation, the device uses the EventDetails bit in the status code. If the bit is set, the device shall provide detailed information for up to 6 individual events.

If the event flag is set, the master shall switch from the service handler to the event handler. The event handler starts by reading the status code. Once the status code has been read the device shall not change the content of the fields to which it has written data in the event buffer. It may write data to the free fields in the event buffer, but not change the status code.

If the event detail bit is set, the master shall read the event details of the events indicated in the status code from the event buffer. Once it has read an event detail, it shall generate a DL event indication. When event handling is complete, the master shall reset the value of the status code to the value it read at the start and revert to the service handler.

Write access to the status code indicates the end of event processing to the device. The device then resets the event flag and may now change the content of the fields in the event buffer. The device does not have to evaluate the content of the master data.

Figure 48 illustrates the sequence for transmitting an event from an application associated with a device, which does not support event details.

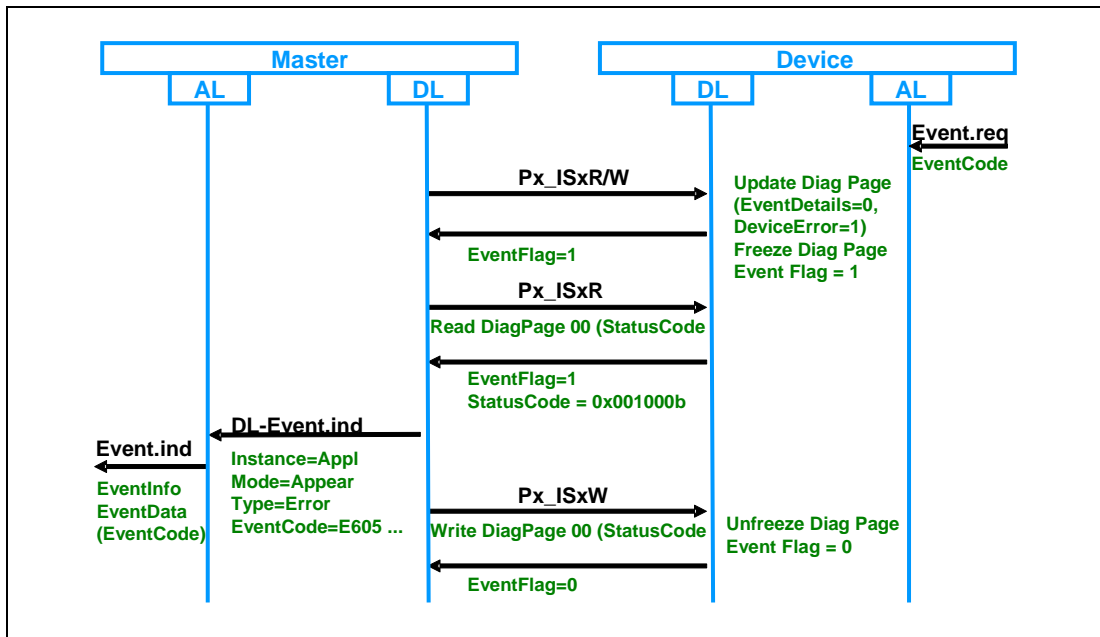


Figure 48 — Event transmission without event details

Figure 49 illustrates the sequence for transmitting an event from an application associated with a device, which does support event details.

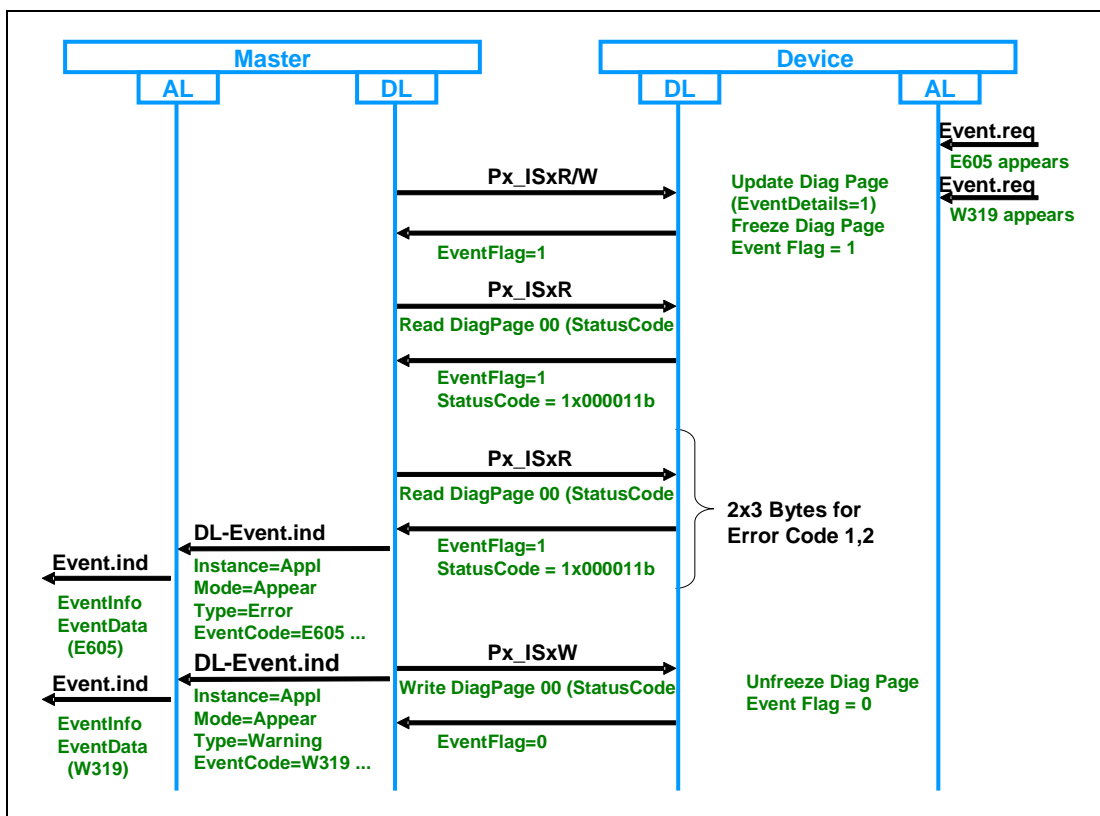


Figure 49 — Event transmission with event details

### 7.2.4.4.2 Coding event information

#### 7.2.4.4.2.1 Status code without event details

The status code entry is coded as a bit field. The reset bit 7, i.e. the value is 0, in the status code entry indicates that summary event information is pending.

#### Bits 0 .. 4: EventCode

Codes for the DL service are listed in Table 48.

**Table 48 — Event**

Service	Bit 4 Communi- cation error	Bit 3 Device Error	Bit 2 Parameter Error	Bit 1 Device Warning	Bit 0 Device Message
Mode = SINGLESHOT, Type = MESSAGE, Instance = Application EventCode = A_MESSAGE	n/a	n/a	n/a	n/a	1
Mode = SINGLESHOT, Type = WARNING, Instance = Application, EventCode = A_WARNING	n/a	n/a	n/a	1	n/a
Mode = SINGLESHOT, Type = ERROR, Instance = Application EventCode = A_PARAMETER	n/a	n/a	1	n/a	n/a
Mode = SINGLESHOT, Type = ERROR, Instance = Application EventCode = A_DEVICE	n/a	1	n/a	n/a	n/a
Mode = SINGLESHOT, Type = ERROR, Instance = unknown EventCode = S_COMM	1	n/a	n/a	n/a	n/a

#### Bit 5: reserved

This bit is reserved and shall be set to zero in this IO-Link version.

#### Bit 6: Invalid Process Data

Setting the Invalid Process Data bit indicates that the device is unable to make any valid process data available.

#### Bit 7: Event Details

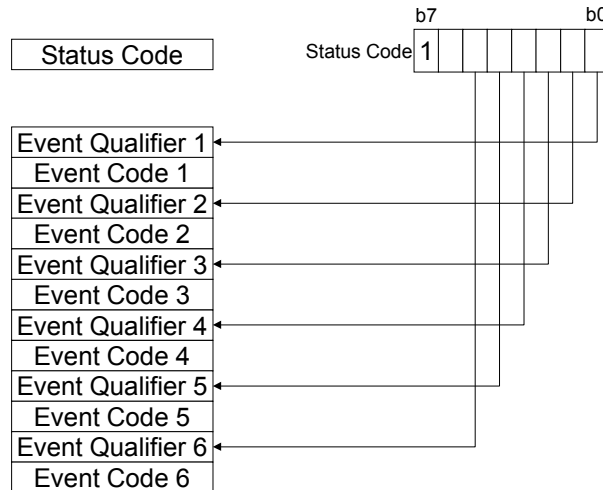
This bit indicates that no detailed event information is available. It shall always be 0 from the status code for this type.

#### 7.2.4.4.2.2 Status code with event details

The status code entry is coded as a bit field. Setting bit 7, i.e. the value is 1, in the status code indicates that, for each event, detailed information is available in the form of an event qualifier and an event code.

#### Bits 0 to 5: Event\_1 to Event\_6

Every bit is linked to an event. Bit 0 is linked to event 1, bit 2 to event 2, etc. (see Figure 50). A set bit indicates that the corresponding event qualifier and the event code have been entered in valid formats in the table. A bit with value 0 indicates an invalid entry.



**Figure 50 — Meaning of Event\_1 to Event\_6**

**Bit 6: Invalid Process Data**

Setting the Invalid Process Data bit indicates that the device is unable to make any valid process data available.

**Bit 7: Event Details**

This bit indicates that detailed event information is available. It shall always be set in the status code for this type.

**7.2.4.4.2.3 Event qualifier**

The event qualifier is illustrated in Figure 51.



**Figure 51 — Event qualifier**

**Bits 0 to 2: INSTANCE**

These bits indicate the instance triggered by the event (event source) and, therefore, the instance forming the basis for the evaluation of the event. Permissible values for INSTANCE are listed in Table 49.

**Table 49 — Values of INSTANCE**

Value	Meaning
0	unknown
1	Phy
2	DL
3	AL
4	Application
5 .. 7	reserved



**Bit 3: reserved**

This bit is reserved and shall be set to zero.

**Bits 4 to 5: TYPE**

These bits indicate the event category. Permissible values for TYPE are listed in Table 50.

**Table 50 — Values of TYPE**

Value	Meaning
0	reserved
1	Information
2	Warning
3	Error

**Bits 4 to 5: MODE**

These bits indicate the event mode. Permissible values for MODE are listed in Table 51.

**Table 51 — Values of MODE**

Value	Meaning
0	reserved
1	Event single shot
2	Event disappears
3	Event appears

**7.2.4.4.2.4 Event Code**

The event code entry contains the ID for an actual event. Permissible values for Event Code are listed in Annex B. The data type of the event code entry is Unsigned16.

**7.2.4.4.2.5 Event channel addresses**

The entries are listed and described in Table 52 (see F.2.10).

The event channel's address range comprises 32 read and write bytes. 19 of these bytes are used for read operations and 1 byte for write operations. The remaining bits are reserved for future expansions.

**Table 52 — Event buffer addresses**

Address	Parameter Name	Description
00h	Status Code	Summary of status and error information. Also used to control read access for individual messages.
01h	Event Qualifier 1	Contains information about the type, mode and source of the first event.
02h	Event Code 1	16-bit event code of the first event.
03h		
04h	Event Qualifier 2	Contains information about the type, mode and source of the second event
05h	Event Code 2	16-bit event code of the second event
06h		

Address	Parameter Name	Description
07h	Event Qualifier 3	Contains information about the type, mode and source of the third event
08h	Event Code 3	16-bit event code of the third event
09h		
0Ah	Event Qualifier 4	Contains information about the type, mode and source of the fourth event
0Bh	Event Code 4	16-bit event code of the fourth event
0Ch		
0Dh	Event Qualifier 5	Contains information about the type, mode and source of the fifth event
0Eh	Event Code 5	16-bit event code of the fifth event
0Fh		
10h	Event Qualifier 6	Contains information about the type, mode and source of the sixth event
11h	Event Code 6	16-bit event code of the sixth event
12h		
13 .. 1Fh		reserved for future use

#### 7.2.4.5 Control handler

The control handler sends the control code contained in Control.req to the cyclic frame handler by sending CMD.req via the “Parameter” data channel. The permissible control codes are listed in Table 53.

**Table 53 — Control codes**

Control code	Master command	Description
PDOOutValid	Process data OutPut Operate,	Process output data valid
PDOOutInvalid	DeviceOperate	Process output data invalid or missing

#### 7.2.5 Frame handler

##### 7.2.5.1 Overview of the frame handler

A frame handler is always active (startup of frame handler or cyclic frame handler). It encodes commands and data in telegrams and sends these to the remote IO-Link device via the physical layer. It receives telegrams from the remote IO-Link device via the physical layer and forwards their content to the required handlers in the form of a confirmation. When the EventFlag is set, the cyclic frame handler generates an Event.ind.

The master frame handler shall realize a retry strategy upon a failed frame, i.e. upon receiving an incorrect checksum, or no checksum at all, from a device. In case of a failed frame the master shall repeat the master frame two times. If the retries are not successful, a negative confirmation shall be provided and the master shall re-initiate the communication by the startup handler via the ESTABLISHCOMM state beginning with Wak-up.

During the startup phase, the startup handler is active, see Figure 52. The individual steps within the startup sequence are controlled by system management.

### Frame handler startup

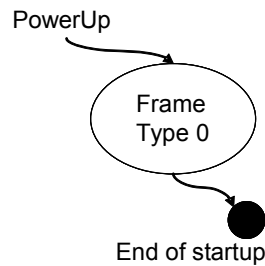


Figure 52 — Frame handler startup

In the Operate state, the on-request data handler and the process data handler are active, see Figure 53. They receive their service requests from the application layer.

### Cyclic frame handler

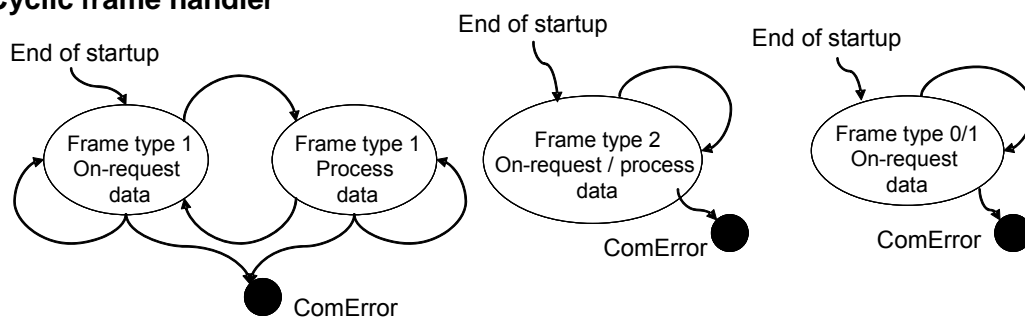


Figure 53 — Cyclic frame handler

#### 7.2.5.2 Frame handler service description

This chapter describes the services the frame handler provides to other internal handlers on the data link layer.

CMD.req = ((R|W), (Event | Parameter | SPDU), (Address | FlowCTRL), [0|1|2]Data byte)

CMD.cnf = ([0|1|2]Data byte)

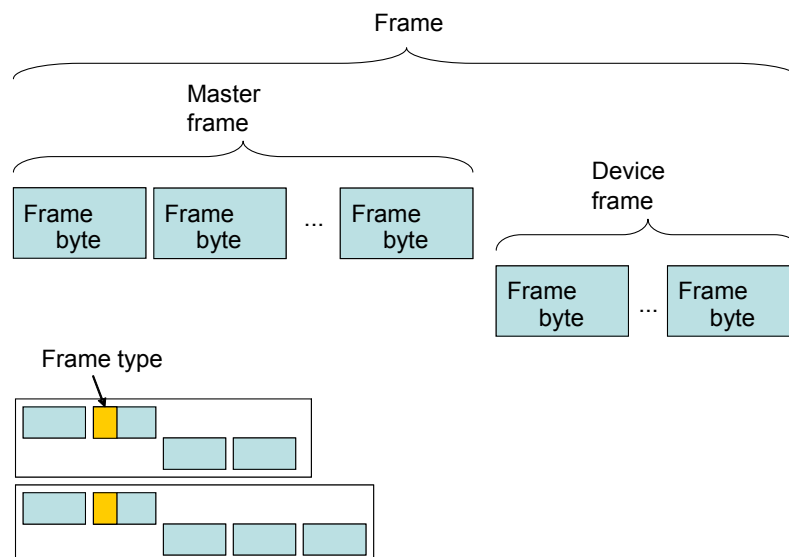
PD.req = ((R|W), (PD), (Address), [0|1|2]Data byte)

PD.cnf = ([0|1|2]Data byte)

#### 7.2.5.3 General structure and encoding

The master and device exchange data in frames. The master telegram contains the CMD byte, the CHECK/TYPE byte and optional data (process data and/or on-request data). The device telegram contains optional data and the CHECK/STAT byte.

Various frame types can be selected to meet the particular needs of an actuator or sensor (scan rate, process data volume). The length of master and device telegrams may vary depending on the type of telegram and the data transmission direction, see Figure 54 (see F.2.11).

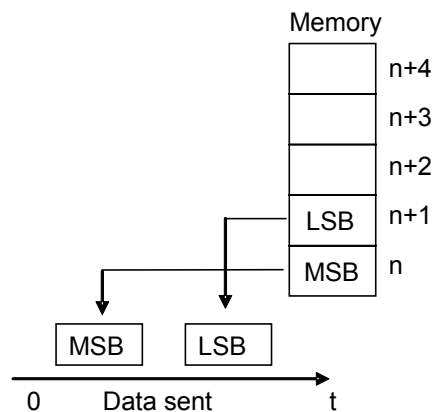


**Figure 54 — Frame structure**

Process data and on-request data use separate channels. The principle of a separate channel for each is used to transmit data qualities to other data qualities in a specified schedule without adverse effects. This ensures a defined dynamic response, in particular in respect of the process data transmitted.

Encoding in accordance with IEC 61158-6, Chapter 6.2 is used to transmit user data. The following data types defined in respect of their transfer syntax are used: Boolean, Integer8, Integer16, Integer32, Unsigned8, Unsigned16, Unsigned32, Floating32, OctetString, VisibleString.

Where the transmission of words is concerned, "big endian" encoding is used (the high-order byte is sent first, followed by the low-order byte, see Figure 55).



**Figure 55 — Transmission data format**

#### 7.2.5.3.1 DL Command

The master specifies the user data to be transmitted in DL Command. This includes the specification of the transmission direction, the data channel and the address of the data on the data channel. The structure of the command is illustrated in Figure 56.



**Figure 56 — DL Command byte**

#### Bits 0 to 4: Address

These bits contain the address, i.e., the byte offset, of the user data on the specified data channel (see also Table 54). If a Service PDU is addressed via the data channel, these bits are used for flow control within the Service PDU. The address, i.e. the position of the user data within the Service PDU, is then only contained indirectly.

#### Bits 5 to 6: Data Channel

These bits contain the data channel for access to the user data. Permissible values for the Data Channel parameter are listed in Table 54.

**Table 54 — Values of Data Channel**

Value	Meaning
0	Process data
1	Parameter data
2	Diagnosis data
3	Service PDU

#### Bit 7: R/W

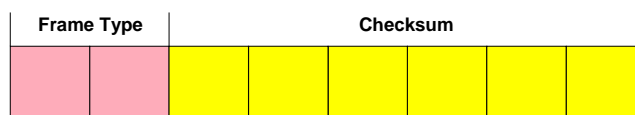
This bit indicates the direction of the user data transmitted via the data channel, i.e., read access (transmission of user data from device to master) or write access (transmission of user data from master to device). Permissible values for the R/W parameter are listed in Table 55.

**Table 55 — Values of R/W**

Value	Meaning
0	Write access
1	Read access

#### 7.2.5.3.2 CHECK/TYPE

The frame type is transmitted together with the checksum in the CHECK/TYPE byte. The structure of CHECK/TYPE is illustrated in Figure 57.



**Figure 57 — CHECK/TYPE byte**

**Bits 0 to 5: Checksum**

These bits contain a 6-bit telegram checksum to ensure data integrity, see also 7.2.5.3.5 and Annex E.

**Bits 6 to 7: Frame type**

These bits contain the frame type. In the frame type, the master specifies how the telegrams in the frame are structured. Permissible values for the Frame type parameter are listed in Table 56.

**Table 56 — Values of Frame type**

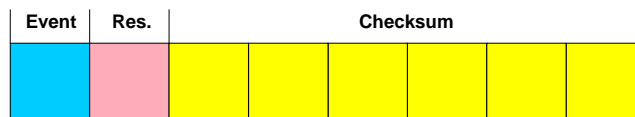
Value	Meaning
0	Type 0
1	Type 1
2	Type 2 (NOTE)
3	reserved
NOTE Subtypes corresponding to the I/O data configuration and determining the process data direction	

**7.2.5.3.3 User data**

The length of user data may vary from 0 to 3 bytes depending on Frame type and R/W.

**7.2.5.3.4 CHECK/STAT**

The event is transmitted together with the checksum in the CHECK/STAT byte. The structure of CHECK/STAT is illustrated in Figure 58.

**Figure 58 — CHECK/STAT byte****Bits 0 to 5: Checksum**

These bits contain a 6-bit telegram checksum to ensure data integrity, see also 7.2.5.3.5 and Annex E.

**Bit 6: Reserved**

This bit shall be set to zero.

**Bit 7: Event**

This bit contains the event flag. The Event data quality implements an active message channel from device to master. The device may report events (errors, limit values) by sending events. Permissible values for the R/W parameter are listed in Table 57.

**Table 57 — Values of Event**

Value	Meaning
0	No Event
1	Event

### 7.2.5.3.5 Calculating the telegram checksum

The telegram checksum provides data integrity protection for data transmission from master to device and from device to master. Block signals is used. Each UART character is protected by a parity bit. All UART characters in a telegram are exclusively-ORed bit-by-bit. The resulting checksum byte is compressed from 8 to 6 bits in accordance with the conversion formula below and added to the CHECK byte on the transmitter. The receiver reverses the procedure.

A start value of 52 hex is used for checksum calculation. This start value is used for exclusive-ORing with the first byte.

Compression from 8 to 6 bytes is performed as follows:

$$D5_6 = D7_8 \text{ xor } D5_8 \text{ xor } D3_8 \text{ xor } D1_8$$

$$D4_6 = D6_8 \text{ xor } D4_8 \text{ xor } D2_8 \text{ xor } D0_8$$

$$D3_6 = D7_8 \text{ xor } D6_8$$

$$D2_6 = D5_8 \text{ xor } D4_8$$

$$D1_6 = D3_8 \text{ xor } D2_8$$

$$D0_6 = D1_8 \text{ xor } D0_8$$

### 7.2.5.4 DLPDU-specific structure and encoding

#### 7.2.5.4.1 Frame

Frames map the logic data channels for IO-Link communication. Each frame type represents a different type of data channel. The specification of different frame types meets the varying requirements of sensors and actuators in respect of their process data width and temporal conditions.

The frame types and their structures are described below.

#### 7.2.5.4.2 Frame types

##### Overview

Frame names are structured as follows:

XPnD\_XSmd

Where:

X = Address [ , I]

P = Process data

n = Number of process data [0, 1, 2]

D = Direction of process data [R, W]

S = On-request data

m = Number of on-request data [0, 1, 2]

d = Direction of on-request data [R, W]

If 2 bytes of process data or 2 bytes of on-request data are sent in one telegram, the byte with the lower address is sent first. Also, if 2 bytes of process data or 2 bytes of on-request data are sent in one telegram, an uneven address may be used. In the event of 1-byte access by a 2 byte

data transfer, the responder shall write the fill byte to the telegram. The receiver shall ignore the surplus byte.

**Frame type 0**

Frame type 0 only transmits on-request data. 1 byte of user data is read (frame P0\_IS1R) or written (frame P0\_IS1W) per cycle. This frame is illustrated in Figure 59.

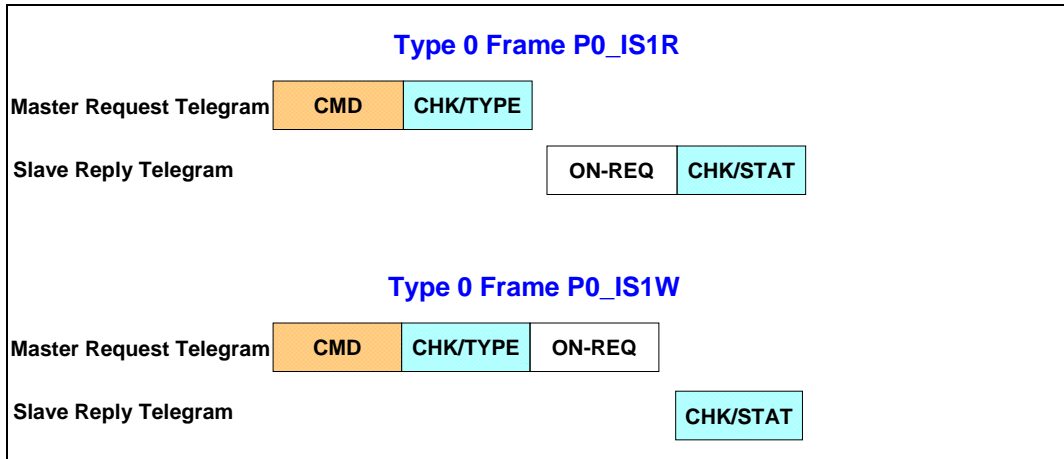


Figure 59 — Frame type 0

**Frame type 1**

Type 1 is optional for devices.

2 bytes of user data are read (frames P0\_IS2R and IP2R\_S0) or written (frames P0\_IS2W and IP2W\_S0) per cycle. This frame is illustrated in Figure 60 for process data and in Figure 61 for on-request data.

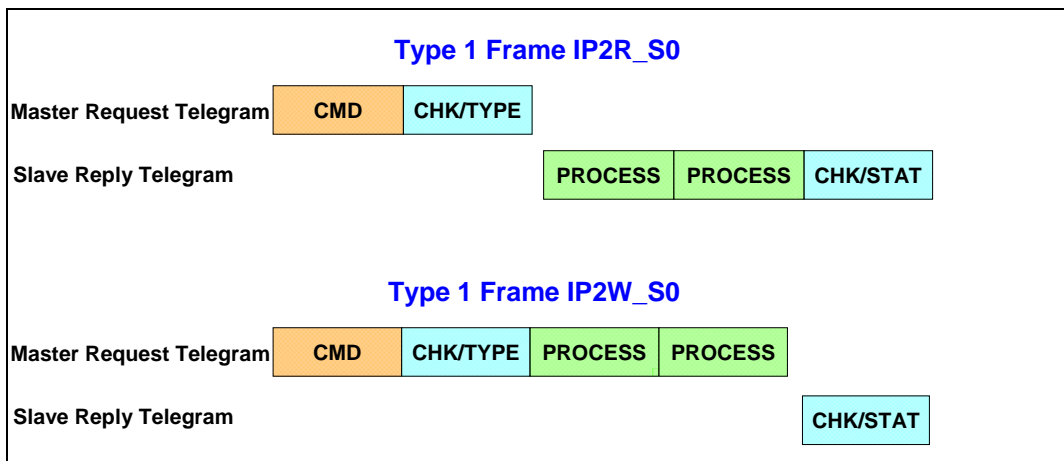


Figure 60 — Frame type 1 (process data)



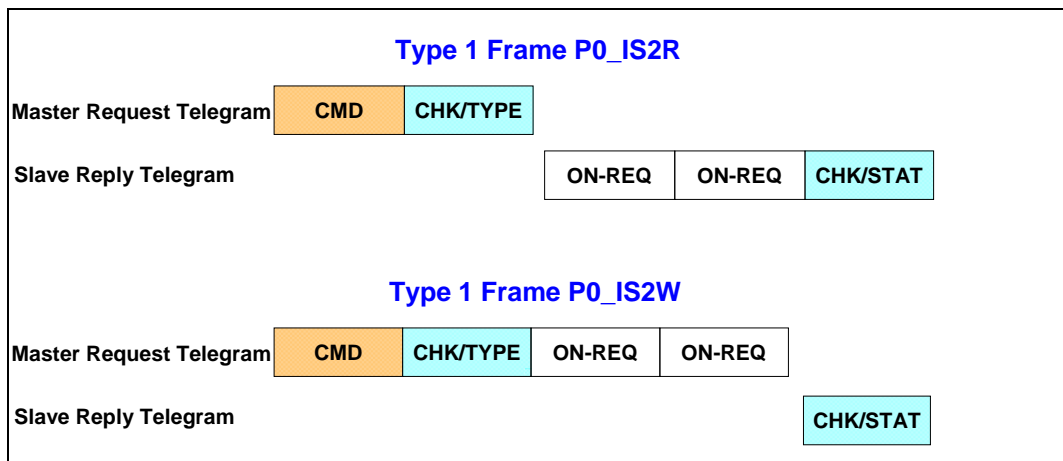


Figure 61 — Frame type 1 (on-request data)

For write access to on-request data (frame P0\_IS2W) via the Parameter and Event data channels, only the first byte of on-request data is ever evaluated. In other words, the device has to reject the second byte.

## Frame type 2

Type 2 is optional for devices.

Frame type 2 transmits process data and on-request data at the same time. The number of process and service data read or written in each cycle is determined by the subtype. The process data address is specified implicitly starting at 0.

Frame subtype 2.1 transmits 1 byte of read process data and 1 byte of read or write on-request per cycle. This frame is illustrated in Figure 62.

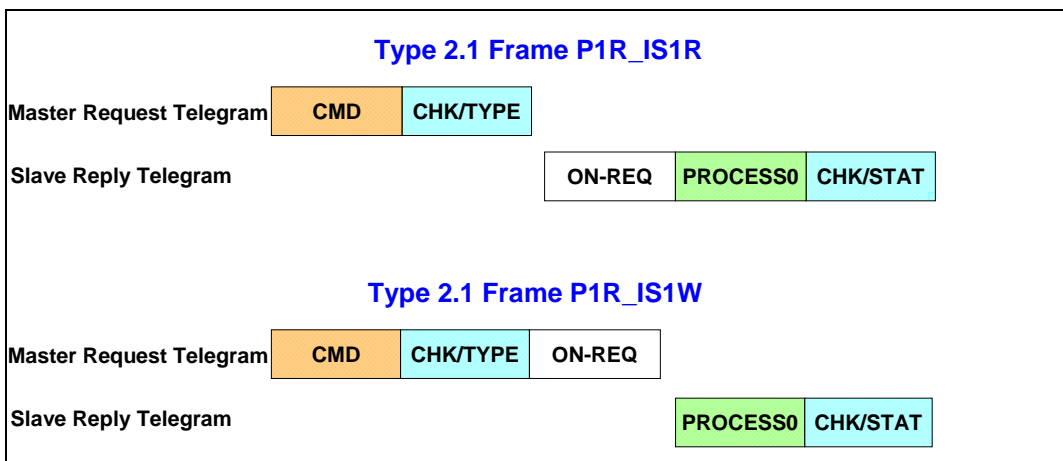


Figure 62 — Frame type 2.1

Frame subtype 2.2 transmits 2 bytes of read process data and 1 byte of read or write service data per cycle. This frame is illustrated in Figure 63.

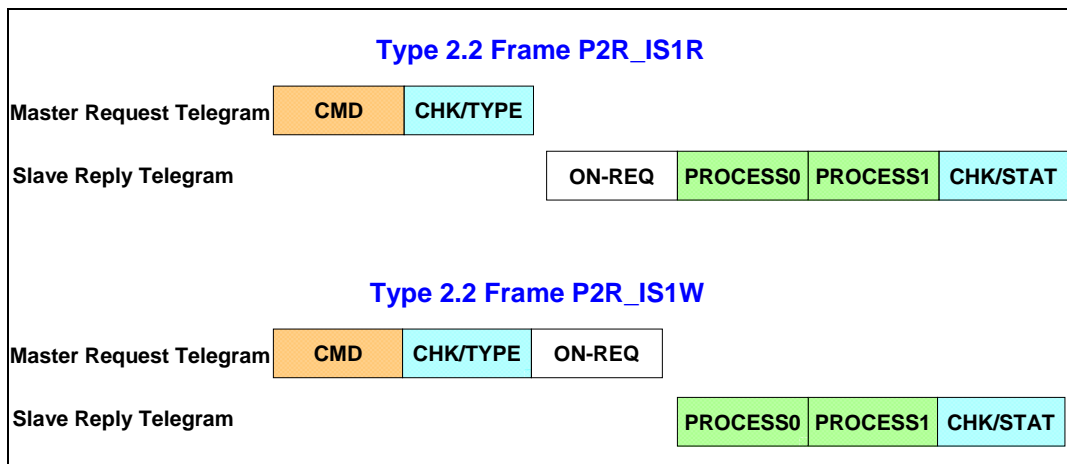


Figure 63 — Frame type 2.2

Frame subtype 2.3 transmits 1 byte of write process data and 1 byte of read or write service data per cycle. This frame is illustrated in Figure 64.

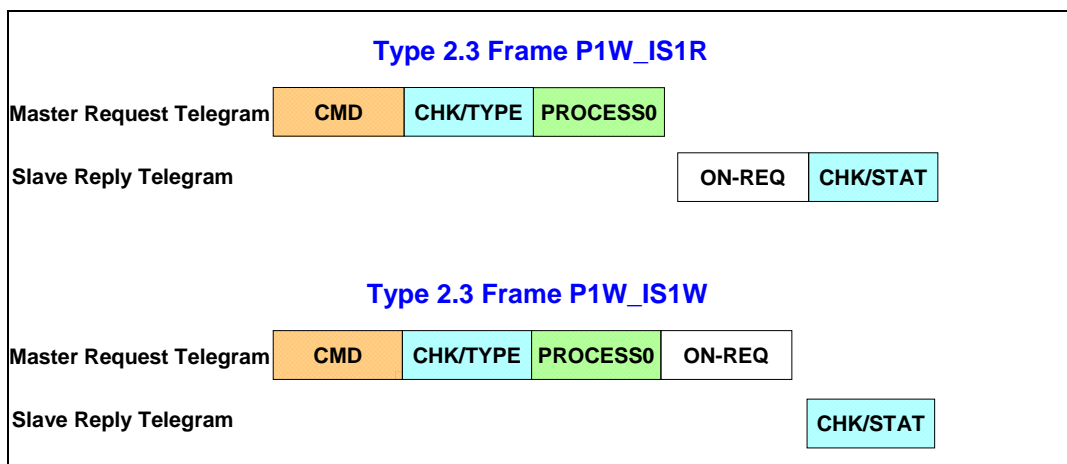


Figure 64 — Frame type 2.3

Frame subtype 2.4 transmits 2 bytes of write process data and 1 byte of read or write on-request data per cycle. This frame is illustrated in Figure 65.

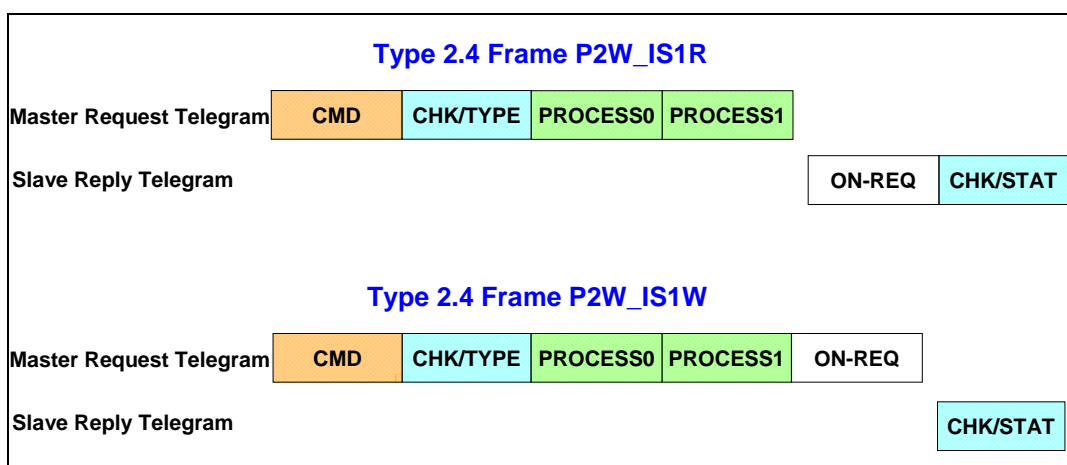
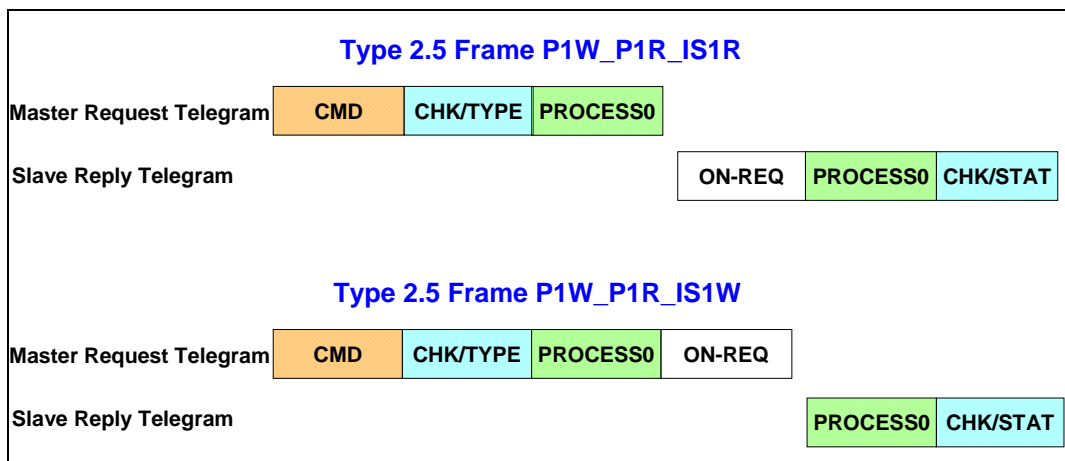


Figure 65 — Frame type 2.4

Frame subtype 2.5 transmits 1 byte of write and read process data and 1 byte of read or write on-request data per cycle. This frame is illustrated in Figure 66.



**Figure 66 — Frame type 2.5**

### 7.2.5.5 Timer operation

#### 7.2.5.5.1 General

The master and device are characterized by a cycle time. This time comprises the frame duration and the cycle's idle time TIDLE. The idle time is determined by the application function of the device and/or the power supply charging time in a 2-wire topology.

#### 7.2.5.5.2 Bit time

The bit time  $T_{BIT}$  is the time it takes to transmit a single bit. It is the inverse value of the data transmission rate:

$$T_{BIT} = 1/(\text{data transmission rate}) \quad (5)$$

Values for  $T_{BIT}$  are specified in 6.2.2.2 (see F.2.12) .

#### 7.2.5.5.3 Character transmission delay on master

The master's character transmission delay is the duration of the pause between the end of the stop bit of a UART character and the start of the start bit of the next UART character. The master shall transmit the UART characters without any gaps.

$$0 \leq t_1 \leq 1 T_{BIT} \quad (6)$$

#### 7.2.5.5.4 Character transmission delay on device

The device's character transmission delay is the duration of the pause between the end of the stop bit of a UART character and the start of the start bit of the next UART character. The device shall transmit the UART characters with a maximum pause of 3 bit times.

$$0 \leq t_2 \leq 3 T_{BIT} \quad (7)$$

NOTE The master does not have to check  $t_2$ . However, it shall start error processing if a frame is not completed within the cycle time.

### 7.2.5.5.5 Response time on device

The device's response time is the duration of the pause between the stop bit of the last UART character sent by the master being received and the start bit of the first UART character being sent. The device shall observe a pause lasting at least 1 but no more than 10 bit times.

$$1 T_{\text{BIT}} \leq t_{\text{A}} \leq 10 T_{\text{BIT}} \quad (8)$$

NOTE The master does not have to check  $t_{\text{A}}$ . However, it shall start error processing if a frame is not completed within the cycle time.

### 7.2.5.5.6 Frame time

Communication between a master and an associated device takes place in a fixed schedule called the frame time (see F.2.13).

$$T_{\text{FRAME}} = (m+n) \times 11 \times T_{\text{BIT}} + t_{\text{A}} + (m-1) \times t_1 + (n-1) \times t_2 \quad (9)$$

where  $m$  is the number of UART characters sent by the master to the device and  $n$  is the number of UART characters sent by the device to the master within a frame.

Figure 67 illustrates the timings during a telegram frame.

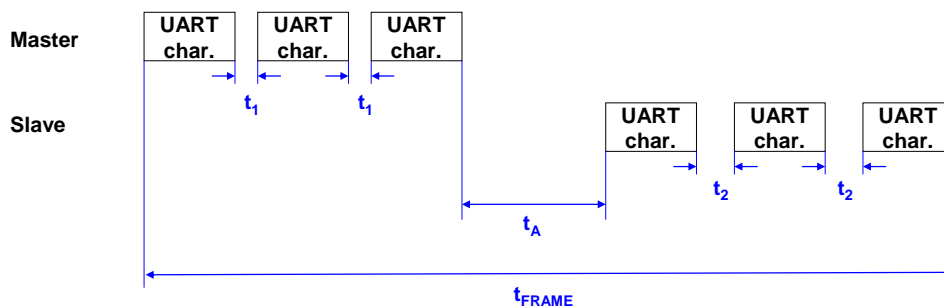


Figure 67 — Frame timing

### 7.2.5.5.7 Cycle time

The cycle time  $t_{\text{CYC}}$  is a configured feature. It is determined by the master's internal implementation and number of ports. The device's "Min Cycle Time" parameter may be used as a basis for deriving the cycle time.

$$t_{\text{CYC}} = t_{\text{FRAME}} + t_{\text{IDLE}} \quad (10)$$

The device's writeable "Master Cycle Time" parameter may be used as a basis for the application on the device to derive time conditions for the safe state.

Table 58 shows cycle times, which should be typically configured for the specified operating modes of the physical layer.

Table 58 — Typical cycle times

Phy operating mode	$t_{\text{CYC}}$
COM1	18,0 ms
COM2	2,3 ms

Phy operating mode	$t_{CYC}$
COM3	2,3 ms

#### 7.2.5.5.8 Idle time

The idle time  $t_{IDLE}$  results from the configured cycle time and the frame time. With reference to a port, it comprises the time between the end of the telegram from the device and the start of the next telegram from the master.

NOTE: The idle time shall be long enough for the internal processing on the device to be completed (and, for PHY1, shall allow time for a reload to take place).

#### 7.2.5.5.9 Offset time

The offset time  $T_{OFS}$  is a configured feature (see F.2.14). It determines when data processing in the device application should start with reference to the start of the cycle, i.e., with reference to the start of the master telegram. The offset enables:

- Device applications to be synchronized with the master cycle within certain limits
- Multiple device applications to be synchronized with one another on different master ports
- Multiple device applications to run with a defined offset on different master ports

Figure 68 illustrates the timings during a telegram cycle.

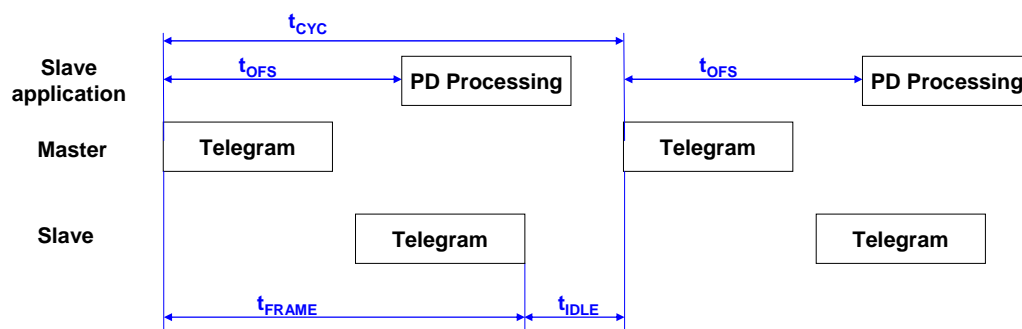


Figure 68 — Cycle timing

#### 7.2.5.6 Error control procedures

The following qualities are detected on the UART layer:

##### Parity errors

Parity errors are handled in the same way as checksums, i.e., if a parity error occurs when a byte is received, error processing takes place as if a checksum error had been detected (even if the checksum in this frame is error-free). This means that duplicate bit errors in different bytes, which may have rectified themselves in the checksum, can also be detected and dealt with.

##### Framing errors

Handled in the same way as parity errors.

##### Timeout

Interpreted as a lack of synchronism within a telegram and, therefore, handled at telegram level.

##### Supplementary notes:

In this context, a collision is understood to occur whenever the master and device send simultaneously due to an error. This error is interpreted as a faulty frame.

## 8 Application layer

### 8.1 Application layer service

#### 8.1.1 Services depending on station type

Table 59 lists the assignment of master and device to their roles as client or server in the context of the execution of individual AL services.

**Table 59 — Assignment of services to master and device**

Service name	Master	Device
Read	I	R
Write	I	R
Abort	I	R
Reject	LI	
GetInput	LR	
NewInput	LI	
SetInput		LR
PDCycle		LI
GetOutput		LR
NewOutput		LI
SetOutput	LR	
Event	R (R)	I R
Control	I	R
Key I Initiator of service R Receiver (Responder) of service (R) Receiver (Responder) of service (but no interaction with application) LR Local service triggered by the application LI Local service triggered by the application layer		

#### 8.1.2 Services for on-request data access

##### 8.1.2.1 On-request data objects

The Read and Write services are used to access on-request data objects addressed via indices. As well as device-specific on-request data objects, standard on-request data objects are specified in the “IO-Link System Description” document.

##### 8.1.2.2 Read

The Read service is used to read on-request data from an IO-Link device connected to a specific port. The parameters of the service primitives are listed in Table 60.

**Table 60 — Read**

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M		
Port	M			
Index	M	M		

Parameter name	.req	.ind	.rsp	.cnf
Subindex	M	M		
Result (+)			S	S(=)
Port				M
Data			M	M(=)
Result (-)			S	S(=)
Port				M
ErrorInfo			M	M(=)

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**Port**

This parameter specifies the port number for the requested on-request data.

Parameter type: Unsigned8

**Index**

This parameter specifies the address of the requested on-request data object on the remote IO-Link device. Index 0 in conjunction with subindex 0 for all direct parameters from addresses 0 to 15 or in conjunction with subindices 1 to 16 for the individual parameter addresses from 0 to 15 addresses the "Parameter" data channel and always supplies a positive result. Index 1 in conjunction with subindex 0 for all direct parameters from addresses 16 to 31 or in conjunction with subindices 1 to 16 for the individual parameter addresses from 16 to 31 addresses the "Parameter" data channel and always supplies a positive result.

Parameter type: Unsigned16

**Subindex**

This parameter specifies the element number of the requested on-request data within a structured on-request data object. An element number with a value of 0 indicates the entire object.

Parameter type: Unsigned8

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**Port**

This parameter specifies the port number for the requested on-request data.

Parameter type: Unsigned8

**Data**

This parameter specifies read values of the requested on-request data.

Parameter type: OctetString

**Result (-):**

This selection parameter indicates that the service request has failed.

**Port**

This parameter specifies the port number for the requested on-request data.

Parameter type: Unsigned8

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: NO\_COMM, NOT\_CONNECTED, NO\_DATA, CONFIG\_FAULT or equivalent to Annex A.

### 8.1.2.3 Write

The Write service is used to write on-request data to an IO-Link device connected to a specific port. The parameters of the service primitives are listed in Table 61.

**Table 61 — Write**

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M		
Port	M			
Index	M	M		
Subindex	M	M		
Data	M	M(=)		
Result (+)			S	S(=)
Port				M
Result (-)			S	S(=)
Port				M
ErrorInfo			M	M(=)

#### Argument

The service-specific parameters of the service request are transmitted in the argument.

##### Port

This parameter specifies the port number for the on-request data to be written.

Parameter type: Unsigned8

##### Index

This parameter specifies the address of the on-request data object to be written on the remote IO-Link device. Index 0 always supplies a negative result. Index 1 in conjunction with subindex 0 for all direct parameters from addresses 16 to 31 or in conjunction with subindices 1 to 16 for the individual parameter addresses from 16 to 31 addresses the "Parameter" data channel and always supplies a positive result.

Parameter type: Unsigned16

##### Subindex

This parameter specifies the element number of the on-request data to be written within a structured on-request data object. An element number with a value of 0 indicates the entire object.

Parameter type: Unsigned8

##### Data

This parameter specifies the values of the on-request data to be written.

Parameter type: OctetString

#### Result (+):

This selection parameter indicates that the service request has been executed successfully.

##### Port

This parameter specifies the port number for the on-request data to be written.

Parameter type: Unsigned8

#### Result (-):

This selection parameter indicates that the service request has failed.

##### Port

This parameter specifies the port number for the requested on-request data.



Parameter type: Unsigned8

#### **ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: NO\_COMM, NOT\_CONNECTED, NO\_DATA, CONFIG\_FAULT or equivalent to Annex A.

### **8.1.2.4 Abort**

The Abort service is used to abort a current Read or Write service on a specific port. The call to this service suppresses the response to a Read or Write service in progress on the master. The parameters of the service primitives are listed in Table 62.

**Table 62 — Abort**

Parameter name	.req	.ind
Argument	M	M
Port	M	

#### **Argument**

The service-specific parameters of the service request are transmitted in the argument.

#### **Port**

This parameter specifies the port number for the on-request data to be written.

Parameter type: Unsigned8

### **8.1.2.5 Reject**

The Reject service is used to reject the Read or Write service requested by the application on a specific port if the requested service cannot be executed in the application layer's current state. There is no response to the requested Read or Write service. The parameters of the service primitives are listed in Table 63.

**Table 63 — Reject**

Parameter name	.req	.ind
Argument	M	M
Port	M	

#### **Argument**

The service-specific parameters of the service request are transmitted in the argument.

#### **Port**

This parameter specifies the port number for the on-request data to be written.

Parameter type: Unsigned8

## **8.1.3 Services for process data access**

### **8.1.3.1 GetInput**

The GetInput local service reads the input data in the process data provided by the local data link layer on an IO-Link device connected to a specific port. The parameters of the service primitives are listed in Table 64.

**Table 64 — GetInput**

Parameter name	.req	.cnf
Argument	M	
Port	M	
Result (+)		S
Port		M
InputData		M
Result (-)		S
Port		M
ErrorInfo		M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**Port**

This parameter specifies the port number for the process data to be read.

Parameter type: Unsigned8

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**Port**

This parameter specifies the port number for the process data to be read.

Parameter type: Unsigned8

**InputData**

This parameter contains the process data values of the requested input data for the specified port.

Parameter type: OctetString

**Result (-):**

This selection parameter indicates that the service request has failed.

**Port**

This parameter specifies the port number for the process data to be read.

Parameter type: Unsigned8

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: NO\_COMM, NOT\_CONNECTED, NO\_DATA, GONFIG\_FAULT or equivalent to Annex A.

**8.1.3.2 NewInput**

The NewInput local service indicates the receipt of up-to-date input data in the process data of an IO-Link device connected to a specific port. The parameters of the service primitives are listed in Table 65.

**Table 65 — NewInput**

Parameter name	.ind
Argument	M
Port	M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**Port**

This parameter specifies the port number for the received process data.

Parameter type: Unsigned8

**8.1.3.3 SetInput**

The SetInput local service updates the input data in the process data on an IO-Link device. The parameters of the service primitives are listed in Table 66.

**Table 66 — SetInput**

Parameter name	.req	.cnf
Argument	M	
Port	M	
InputData	M	
Result (+)		S
Result (-)		S
ErrorInfo		M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**Port**

This parameter specifies the port number for the process data to be written.

**InputData**

This parameter contains the process data values of the input data to be transmitted.

Parameter type: OctetString

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**Result (-):**

This selection parameter indicates that the service request has failed.

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: STATE\_CONFLICT or equivalent to Annex A.

**8.1.3.4 PDCycle**

The PDCycle local service indicates the end of a process data cycle. The device application may use this service to transmit new input data to the AL with SetInput. The parameters of the service primitives are listed in Table 67.

**Table 67 — PDCycle**

Parameter name	.ind
Argument	
Port	M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**Port**

This parameter specifies the port number for the process data to be written.

Parameter type: Unsigned8

**8.1.3.5 GetOutput**

The GetOutput local service reads the output data in the process data provided by the local data link layer on an IO-Link device. The parameters of the service primitives are listed in Table 68.

**Table 68 — GetOutput**

Parameter name	.req	.cnf
Argument	M	
Port		M
Result (+)		S
InputData		M
Result (-)		S
ErrorInfo		M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**Port**

This parameter specifies the port number for the process data to be written.

Parameter type: Unsigned8

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**OutputData**

This parameter contains the process data values of the requested output data for the specified port.

Parameter type: OctetString

**Result (-):**

This selection parameter indicates that the service request has failed.

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: NO\_COMM, NOT\_CONNECTED, NO\_DATA, GONFIG\_FAULT or equivalent to Annex A.

### 8.1.3.6 NewOutput

The NewOutput local service indicates the receipt of up-to-date output data in the process data on an IO-Link device. The parameters of the service primitives are listed in Table 69.

**Table 69 — NewOutput**

Parameter name	.ind
Argument	M
Port	M

#### Argument

The service-specific parameters of the service request are transmitted in the argument.

#### Port

This parameter specifies the port number for the process data to be written.

Parameter type: Unsigned8

### 8.1.3.7 SetOutput

The SetOutput local service updates the output data in the process data on an IO-Link Master. The parameters of the service primitives are listed in Table 70.

**Table 70 — SetOutput**

Parameter name	.req	.cnf
Argument	M	
Port	M	
OutputData	M	
Result (+)		S
Port		M
Result (-)		S
Port		M
ErrorInfo		M

#### Argument

The service-specific parameters of the service request are transmitted in the argument.

#### Port

This parameter specifies the port number for the process data to be written.

Parameter type: Unsigned8

#### OutputData

This parameter contains the process data in the output data to be written for the specified port.

Parameter type: OctetString

#### Result (+):

This selection parameter indicates that the service request has been executed successfully.

#### Port

This parameter specifies the port number for the process data to be written.

Parameter type: Unsigned8

**Result (-):**

This selection parameter indicates that the service request has failed.

**Port**

This parameter specifies the port number for the process data to be written.

Parameter type: Unsigned8

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: STATE\_CONFLICT or equivalent to Annex A.

**8.1.4 Event**

The Event service indicates a pending status or error message. The source of the event may be local or remote. The event may be triggered by a communication layer or by an application. The parameters of the service primitives are listed in Table 71.

**Table 71 — Event**

Parameter name	.req	.ind
Argument	M	M
Port		U
Instance	M	M
Mode	M	M
Type	M	M
PDvalid	M	M
Local		M
EventCode	M	M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**Port**

This parameter specifies the port number for the event data.

**Instance**

This parameter specifies the event source.

Permitted values: unknown, PhL, DL, AL, Application

**Type**

This parameter specifies the event category.

Permitted values: ERROR, WARNING, MESSAGE

**Mode**

This parameter specifies the event mode.

Permitted values: SINGLESHOT, APPEARS, DISAPPEARS

**PDvalid**

This parameter indicates whether, given the conditions of the event that has occurred, the device is going to continue to be able to provide valid process data values.

Permitted values: valid, invalid

**EventCode**

This parameter contains a code identifying the event.

Parameter type: Unsigned16

Permitted values: see Annex A.

**Local**

This parameter indicates that the event was generated in the local communication section.

Permitted values: local, remote

**8.1.5 Control**

The Control service contains control information relating to the master application on the device or the device application. The parameters of the service primitives are listed in Table 72.

**Table 72 — Control**

Parameter name	.req	.ind
Argument	M	M
ControlCode	M	M

**Argument**

The service-specific parameters are transmitted in the argument.

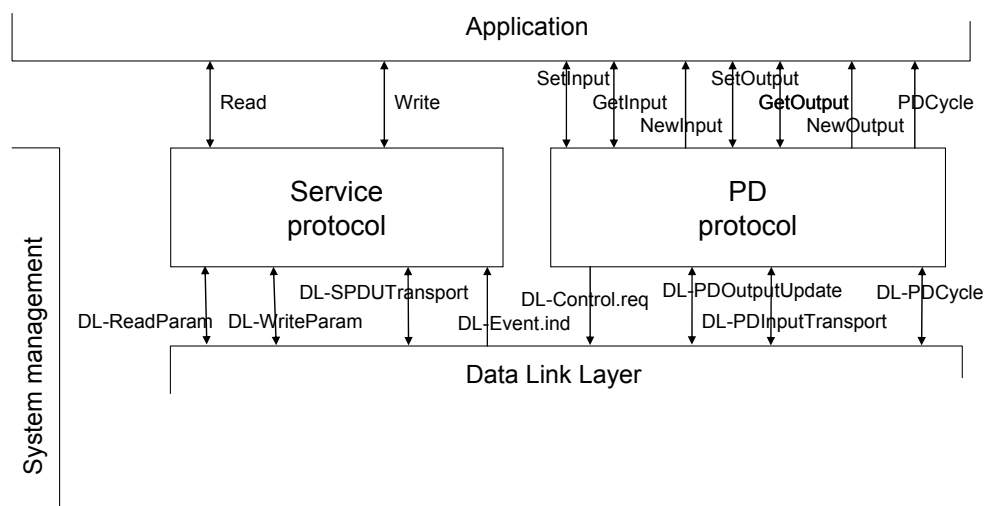
**ControlCode**

This parameter contains a code identifying the event.

Permitted values: PdOutValid, PdOutInvalid

**8.2 Application layer protocol****8.2.1 Overview**

The following Figure 69 illustrates the structure and interfaces of the application layer.

**Figure 69 — Structure and interfaces of the AL****8.2.2 The process data cycle**

The sequence for transporting process data is illustrated in Figure 70 (see F.2.15) for output data and in Figure 71 for input data.

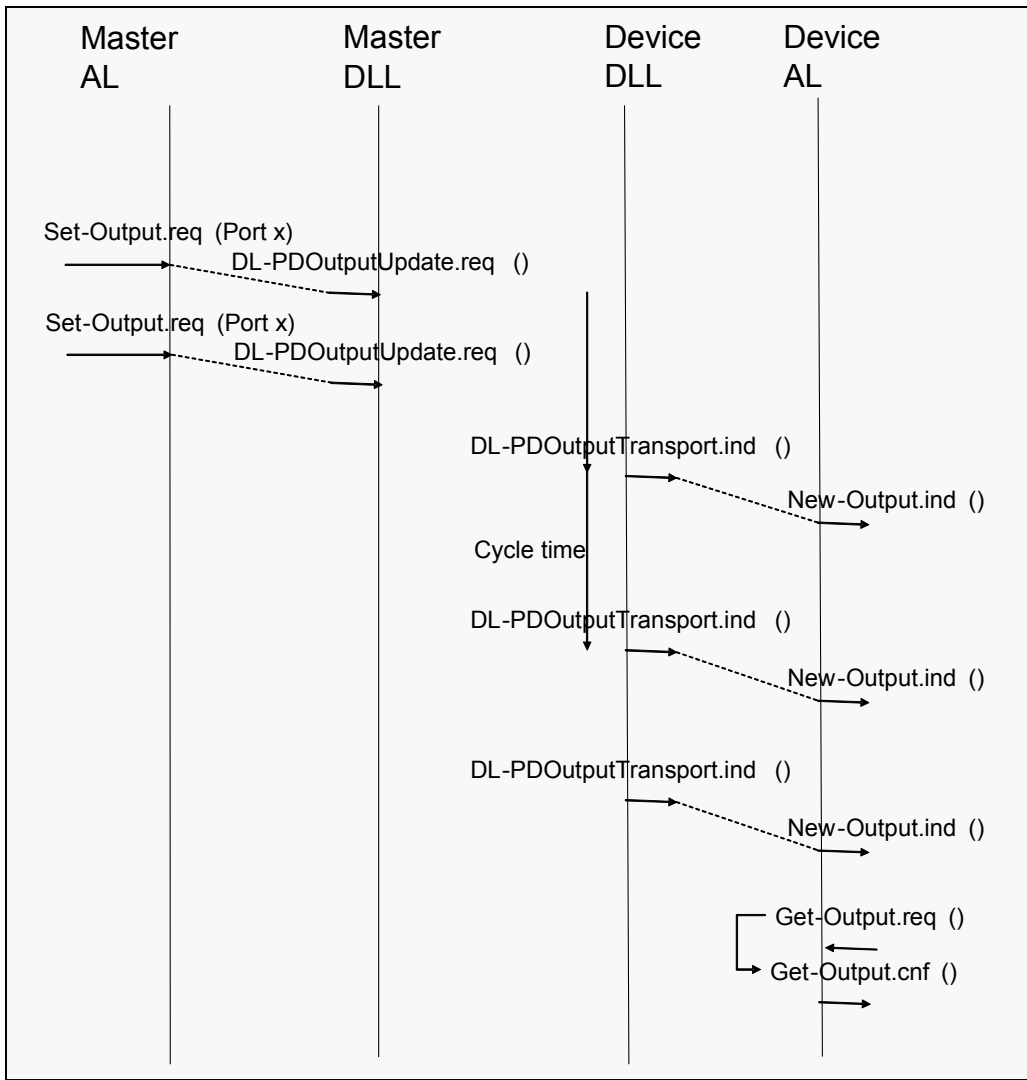


Figure 70 — Output data transmission



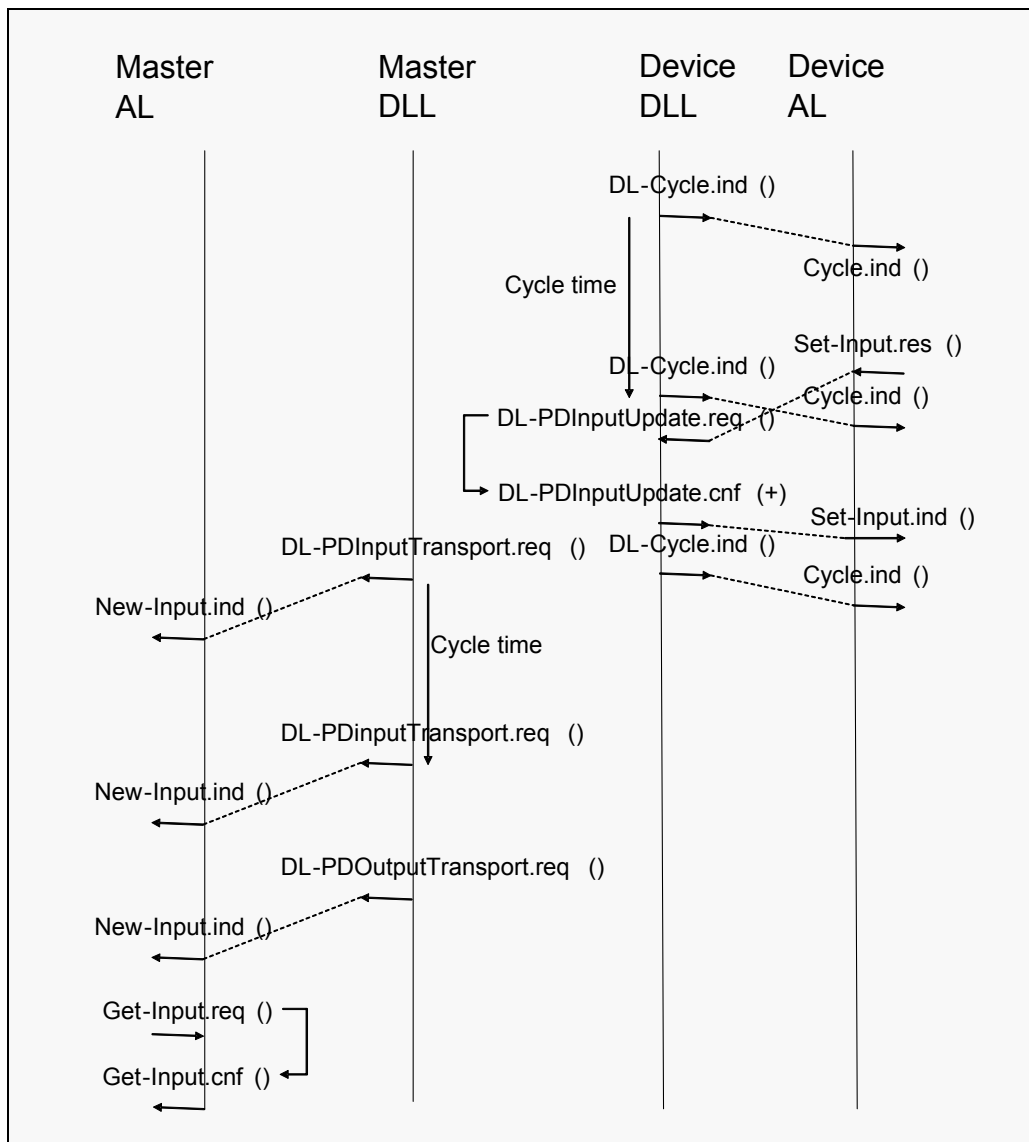
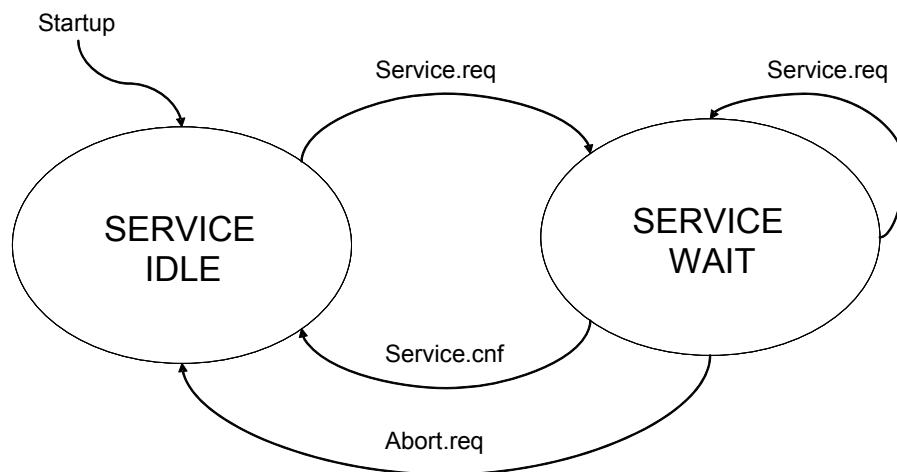


Figure 71 — Input data transmission

### 8.2.3 On-request data transfer

#### 8.2.3.1 State machine on the master application layer

The state machine of the application layer on the master is illustrated in Figure 72.



**Figure 72 — Master AL state machine**

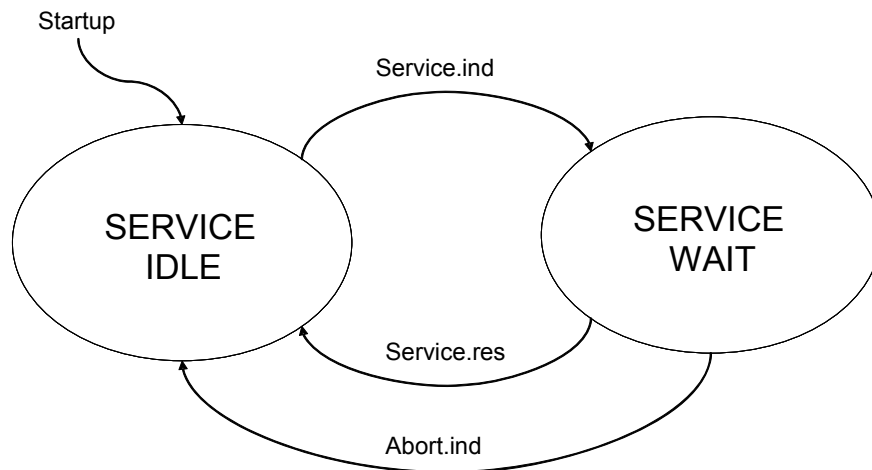
The states defined for the application layer on the master are summarized in Table 73.

**Table 73 — Master AL states**

State Name	Description
SERVICEIDLE	<p>The AL waits for a service request from the application.</p> <p>When a service request is received, it is encoded. A DL-SPduTransport.req is sent to the DL and the AL switches to the SERVICEWAIT state.</p> <p>Any DL-SPduTransport.ind received shall be ignored</p>
SERVICEWAIT	<p>In the SERVICEWAIT state, the AL expects to receive a DL-SPduTransport.ind from the DL.</p> <p>When the indication is received, it is decoded. A service confirmation is sent to the application and the AL switches to the SERVICEIDLE state.</p> <p>In the event of communication errors or aborts, a negative service confirmation is sent to the application and the AL switches to the SERVICEIDLE state.</p> <p>If communication is aborted by the local application sending an Abort.req, the AL switches to the SERVICEIDLE state without a service confirmation.</p> <p>If a service request from the application is received, a Reject.ind is sent in response.</p>

### 8.2.3.2 State machine on the device application layer

The state machine of the application layer on the device is illustrated in Figure 73.



**Figure 73 — Device AL state machine**

The states defined for the application layer on the device are summarized in Table 74.

**Table 74 — Device AL states**

State Name	Description
SERVICEIDLE	<p>In the SERVICEIDLE state, the AL expects to receive a DL-SPduTransport.ind from the DL.</p> <p>When the indication is received, it is decoded. A service indication is sent to the application and the AL switches to the SERVICEWAIT state.</p> <p>If a service response from the application is received, a Reject.ind is sent in response.</p>
SERVICEWAIT	<p>The AL waits for a service response from the application.</p> <p>When a service response is received, it is encoded. A DL-SPduTransport.req is sent to the DL and the AL switches to the SERVICEIDLE state.</p> <p>Any DL-Abort.ind received is forwarded to the application and the AL switches to the SERVICEIDLE state.</p>

**NOTE** The device application shall generate a negative service response, if a not supported index or subindex was tried to access.

### Event transfer

The master application layer forwards events displayed by the data link layer and events generated on the application layer to the application.

The device application layer forwards events requested by the application and events generated on the application layer to the data link layer.

Figure 74 illustrates the master and device sequences (see F.2.16).

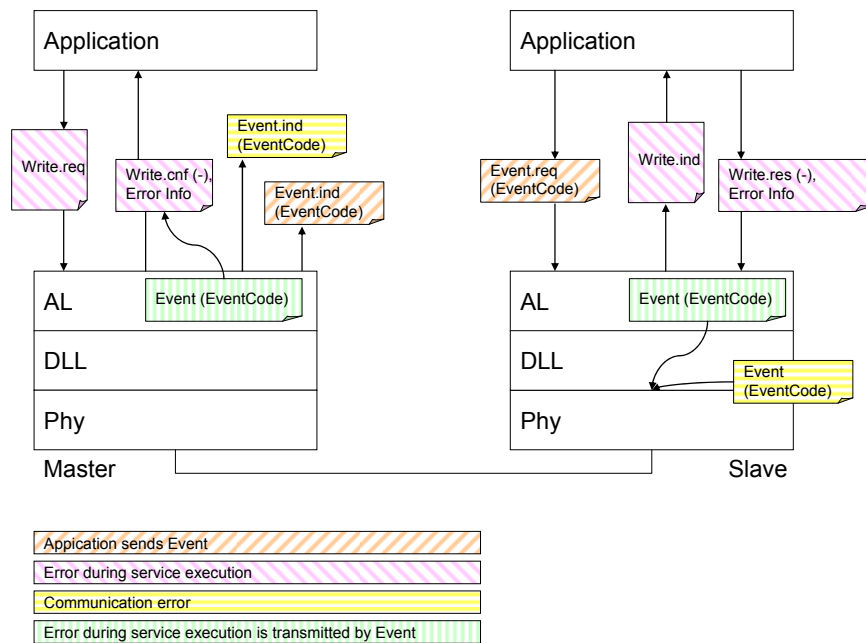


Figure 74 — Event transfer

8.2.4 ALPDU-specific structure and encoding

8.2.4.1 Structure of the Service PDU

8.2.4.1.1 General structure

The general structure of a Service PDU is illustrated in Figure 75, where the sequence of the elements in the diagram corresponds to the transmission sequence. The elements of a Service PDU can take various forms depending on the type of service.

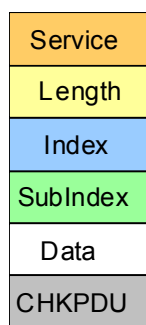


Figure 75 — Service PDU

The Service PDU allows accessing data object to be transmitted. The data objects shall be addressed by the “Index” element.

### 8.2.4.1.2 Service

The type of service is specified in the “Service” element of a Service PDU. Permissible values for “Service” are listed in Table 75. This definition differs for requests (service transmission from master to device) and responses (service transmission from device to master).

**Table 75 — Definition for Service**

Service Code (bin)	Meaning		Index format
	Master	Device	
0000	No Service	No Service	n/a
0001	Write Request	reserved	8-bit index
0010	Write Request	reserved	8-bit index and subindex
0011	Write Request	reserved	16-bit index and subindex
0100	reserved	Write Response (-)	none
0101	reserved	Write Response (+)	none
0110	reserved	reserved	
0111	reserved	reserved	
1000	reserved	reserved	
1001	Read Request	reserved	8-bit index
1010	Read Request	reserved	8-bit index and subindex
1011	Read Request	reserved	16-bit index and subindex
1100	reserved	Read Response (-)	none
1101	reserved	Read Response (+)	none
1110	reserved	reserved	
1111	reserved	reserved	

NOTE 1 The device need not to support all service codes. The device shall send the ILLEGAL\_SERVICE error message in response to service codes, which it does not support (event channel).

NOTE 2 The EVENT “ComError” shall be generated in response to a reserved service code.

### 8.2.4.1.3 Length

The number of bytes transmitted in this service, including all protocol information, is specified in the “Length” element of a Service PDU. If the total length is more than 15 bytes, the length is specified using an extended length. Permissible values for “Length” are listed in Table 76 (see F.2.17).

**Table 76 — Definition for Length**

Service	Length	Extended length	Meaning
0	0	n/a	No service, PDU length is 1.
0	1	n/a	Device busy, PDU length is 1.
0	2 ..15	n/a	reserved
1 .. 15	0	n/a	reserved
1 .. 15	1	16 to 232	Length of Service PDU in extended length
1 .. 15	1	233 to 255	reserved
1 .. 15	2 to 15	n/a	Length of Service PDU

#### 8.2.4.1.4 Index and SubIndex

The parameter address of the data object to be transmitted using the Service PDU is specified in the “Index” element. “Index” has a range of values from 0 to 65535. Index values 0 and 1 shall not be provided by the device.

The data element address of a structured parameter of the data object to be transmitted using the Service PDU is specified in the “SubIndex” element. “SubIndex” has a range of values from 0 to 255, whereby a value of 0 is used to reference the entire object.

Table 77 shows the index formats used in the Service PDU depending on the parameters transmitted to the service interface of the application layer.

**Table 77 — Use of index formats**

Index	SubIndex	Index format of Service PDU
0 .. 255	0	8-bit index
0 .. 255	1.. 255	8-bit index and subindex
256 .. 65535	0 .. 255	16-bit index and subindex

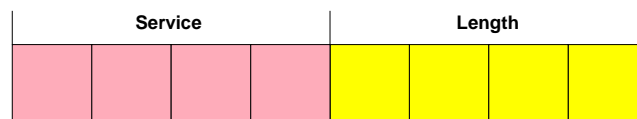
NOTE The device need not to support all Index and Subindex values. The device shall send a negative response to Index or Subindex values, which it does not support.

#### 8.2.4.1.5 Data

The “Data” element contains the user data. The data length corresponds to the entries in the “Length” element minus the protocol frame; the content is application-specific.

#### 8.2.4.2 Encoding of the Service PDU

The Service and Length parameters are transmitted together in the SERVICE byte. The structure of SERVICE is illustrated in Figure 76.



**Figure 76 — SERVICE byte**

##### Bits 0 to 3: Length

The encoding of the Length element of the Service PDU is defined in Table 76.

##### Bits 4 to 7: Service

The encoding of the Service element of the Service PDU is defined in Table 75.

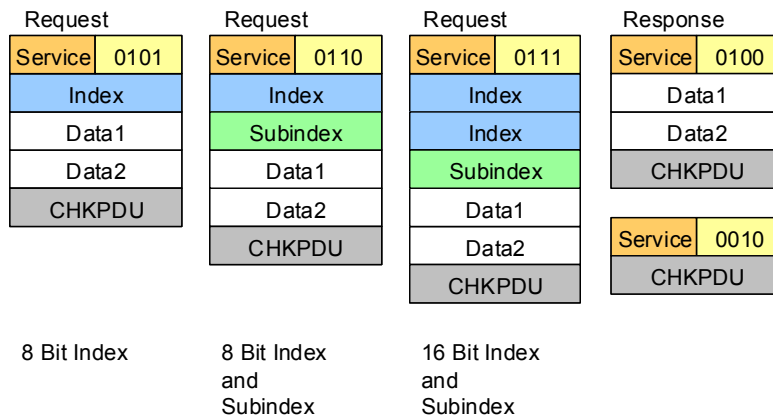
All other elements are transmitted as independent bytes.

Table 78 describes the syntax of the Service PDUs.

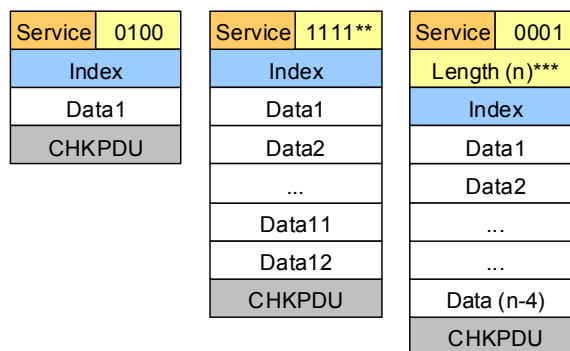
**Table 78 — Service PDU syntax**

SPDU name	SPDU structure
Write Request	{Service(0x1), LEN, Index, [Data*], CHPDU} ^ {Service(0x2), LEN, Index, Index, [Data*], CHKPDU} ^ {Service(0x3), LEN, Index, Index, SubIndex, [Data*], CHPDU}
Write Response (+)	Service(0x4), Length(0x2), CHPDU
Write Response (-)	Service(0x5), Length(0x4), ErrorInfo, CHPDU
Read Request	{Service(0x9), Length(0x4), Index, CHPDU} ^ {Service(0xA), Length(0x5), Index, Index, CHKPDU} ^ {Service(0xB), Length(0x6), Index, Index, SubIndex, CHPDU}
Read Response (+)	Service(0xC), LEN,[Data*], CHPDU
Read Response (-)	Service(0xD), Length(0x4), ErrorInfo, CHPDU
LEN	{Length(0x1), ExtendedLength} ^ {Length}

Figure 77 through Figure 80 (see F.2.18) illustrate typical examples of Service PDUs.

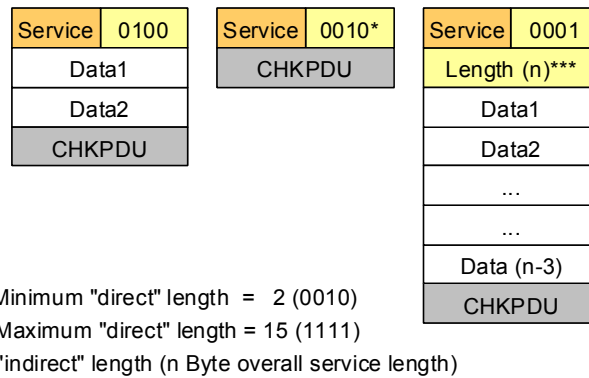
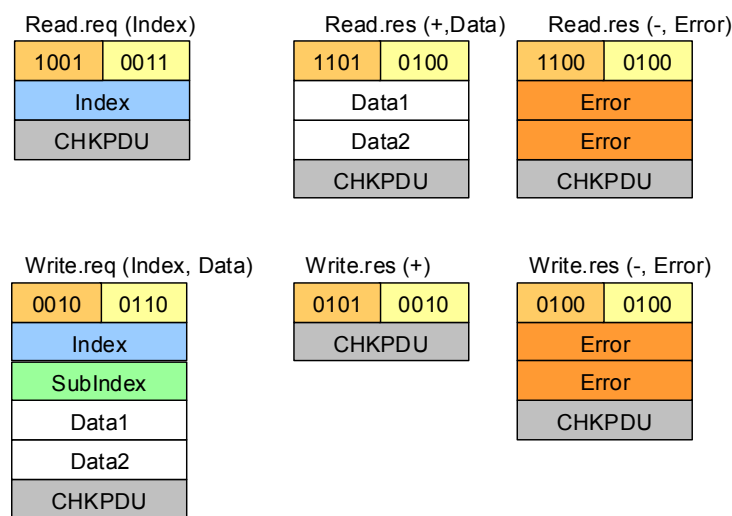


**Figure 77 — Examples of index formats in Service PDUs**



\* Minimum "direct" length = 2 (0010)  
 \*\* Maximum "direct" length = 15 (1111)  
 \*\*\* "indirect" length (n Byte overall service length)

**Figure 78 — Examples of Request Service PDUs**

**Figure 79 — Examples of Response Service PDUs****Figure 80 — Examples of Read and Write Service PDUs**

## 9 System management

### 9.1 System management services

#### 9.1.1 Services depending on station type

This chapter describes the services the system management provides to the user via its upper interface. Table 79 lists the assignment of master and device to their roles as client or server in the context of the execution of individual system management services.

**Table 79 — Assignment of services to master and device**

Service name	Master	Device
SetMode	LR	
Get Mode	LR	
SM-Operate	LR	



Key
I Initiator of service
R Receiver (Responder) of service
(R) Receiver (Responder) of service (but no interaction with application)
LR Local service triggered by the application
LI Local service triggered by the application layer

### 9.1.2 SetMode

The SetMode service is used to set the mode for local communication. On the basis of this setting, system management shall switch the state machines on the communication layer. The parameters of the service primitives are listed in Table 80.

**Table 80 — SetMode**

Parameter name	.req	.cnf
Argument	M	
PortFeature	M	
Result (+)		S(=)
RealMode		M
Baudrate		U
Result (-)		S(=)
ErrorInfo		M(=)

#### Argument

The service-specific parameters of the service request are transmitted in the argument.

#### PortFeature

This parameter specifies the features for communication on an IO-Link master port.

Parameter type: Record

Record Elements:

Port Number

Cycle Time (0: FreeRunning, >0: time in us)

Target Mode (PHY1, PHY2, *(configuration of the port at this point not complete Automode, etc.)* (see F.2.19)

Deactivated, DigitalINPUT, DigitalOUTPUT, IOLink

Set baud rate: (Auto, Com1, Com2, Com3)

The meaning of the values for Target Mode is described in Table 81.

**Table 81 — Target modes**

Target Mode	Result	Remark
ComPhy1	(+): Device found	Starts IO-Link communication without WakeUp pulse at a baud rate of 230 K for 2-wire topology
	(-): No device connected, search mode activated	Starts IO-Link communication without WakeUp pulse at a baud rate of 230 K for 2-wire topology

Target Mode	Result	Remark
ComPhy2	(+): Device found, startup parameters in confirmation	Starts IO-Link communication with WakeUp pulse
	(-): No device connected, search mode activated	Starts IO-Link communication with WakeUp pulse
SIO	(+)	IO-Link communication is stopped

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**RealMode** (see F.2.19)

This parameter specifies the actual resulting mode on the data link layer.

Parameter type: Record

Record Elements:

Port Number

OperationMode (Inactive, ComPHY1, ComPHY2, SIO, DigitalINPUT, DigitalOUTPUT)

Actual baud rate (Com1, Com2, Com3)

The meaning of the values for Target Mode is described in Table 81 .

Parameter type: Enumeration

Permitted values: IO\_LINK, OK

**Baudrate**

This parameter specifies the actual data rate on the data link layer.

Parameter type: Enumeration

Permitted values: 4K8, 38K4, 230K4

**Result (-):**

This selection parameter indicates that the service request has failed.

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values:

NOT\_CONNECTED = No communication

DEVICE\_CYCLE\_TIME\_CONFLICT = The required CycleTime cannot be fulfilled on the device.

MASTER\_CYCLE\_CONFLICT = The required CycleTime is in conflict with ports that are already active.

STATE\_CONFLICT = Service not allowed in this state.

**9.1.3 GetMode**

The GetMode service is used to read out the local communication mode. The parameters of the service primitives are listed in Table 82.

**Table 82 — GetMode**

Parameter name	.req	.cnf
Argument	M	
Result (+)		S(=)
PortFeature		M
RealMode		M
Baudrate		U
Result (-)		S(=)
ErrorInfo		M

**Argument**

The service-specific parameters of the service request are transmitted in the argument.

**PortFeature**

See SetMode

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**RealMode**

This parameter specifies the actual resulting mode on the data link layer.

Parameter type: Enumeration

Permitted values: IO\_LINK, OK

**Baudrate**

This parameter specifies the actual data rate on the data link layer.

Parameter type: Enumeration

Permitted values: 4K8, 38K4, 230K4

**Result (-):**

This selection parameter indicates that the service request has failed.

**ErrorInfo**

This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values:

NOT\_CONNECTED = No communication

STATE\_CONFLICT = Service not allowed in this state

**9.1.4 SM-Operate**

The SM-Operate service prompts system management to calculate the cycle times of the ports acknowledged positively with SM-SetMode.cnf(+). The parameters of the service primitives are listed in Table 83.

**Table 83 — SM-Operate**

Parameter name	.req	.cnf
Argument	M	

Parameter name	.req	.cnf
Result (+)		S
Result (-)		S
ErrorInfo		M

**Result (+):**

This selection parameter indicates that the service request has been executed successfully.

**Result (-):**

This selection parameter indicates that the service request has failed.

**ErrorInfo**

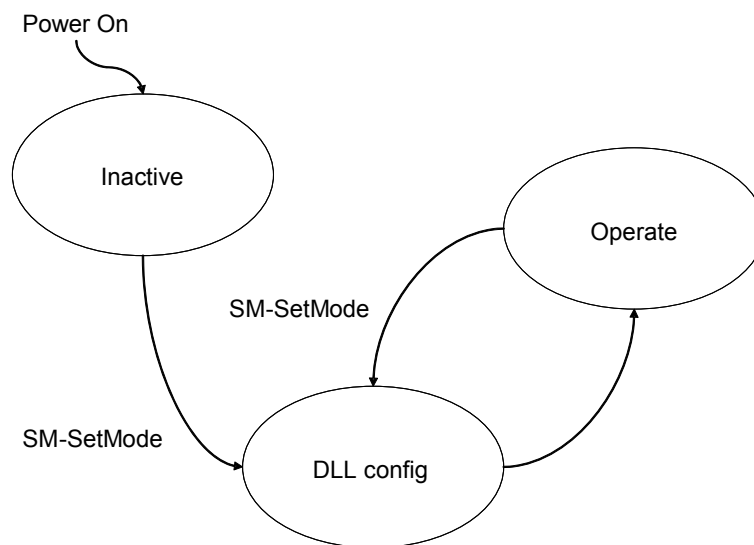
This parameter contains error information to supplement the Result parameter.

Parameter type: ERROR\_TYPE

Permitted values: STATE\_CONFLICT

**9.2 System management protocol**

Figure 81 shows the system management state machine.



**Figure 81 — System management states**

DLL config sends DL-SetMode.

Operate sends DL-SetMode, thereby initializing the device and setting it to DL Operate.

The device may be reconfigured using DL-SetMode-Fallback.

**9.2.1 Startup**

The application sends SM-SetMode.req for all ports to prompt system management to set one port to the selected target mode. By acknowledging this request positively, system management indicates that it has been able to set the port accordingly. A negative acknowledgment from system management indicates that the port is not in use.

Note The DL shall continue to attempt to set the port cyclically. If it finds a device, the DL shall send a DL-Mode.ind to system management and the latter shall generate an Event.ind.

Following receipt of all SM-SetMode.cnf telegrams, the application shall send SM-Operate.req to prompt system management to calculate and activate the cycle time of all ports acknowledged positively (Figure 82, Figure 83, Figure 84).

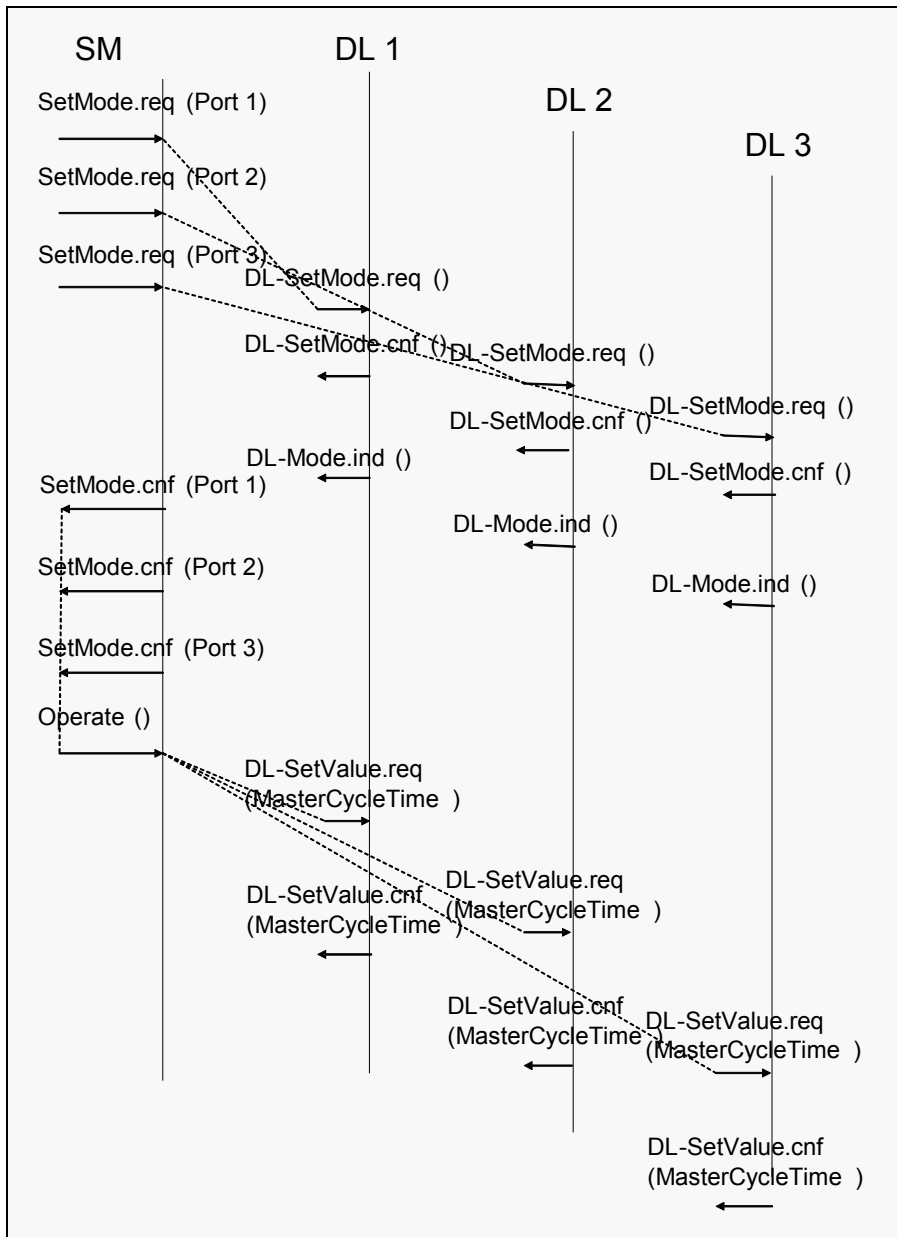


Figure 82 — System management start-up

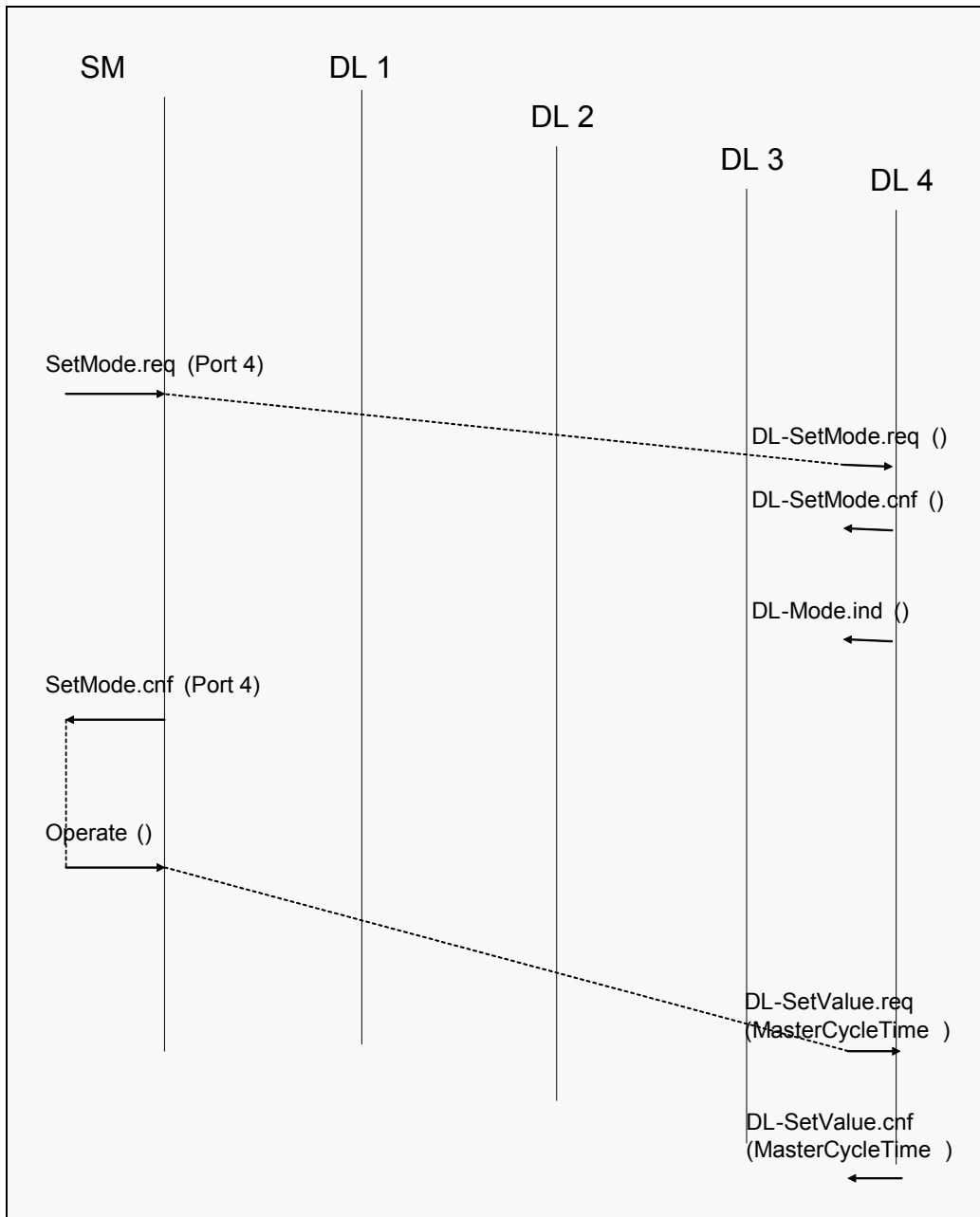
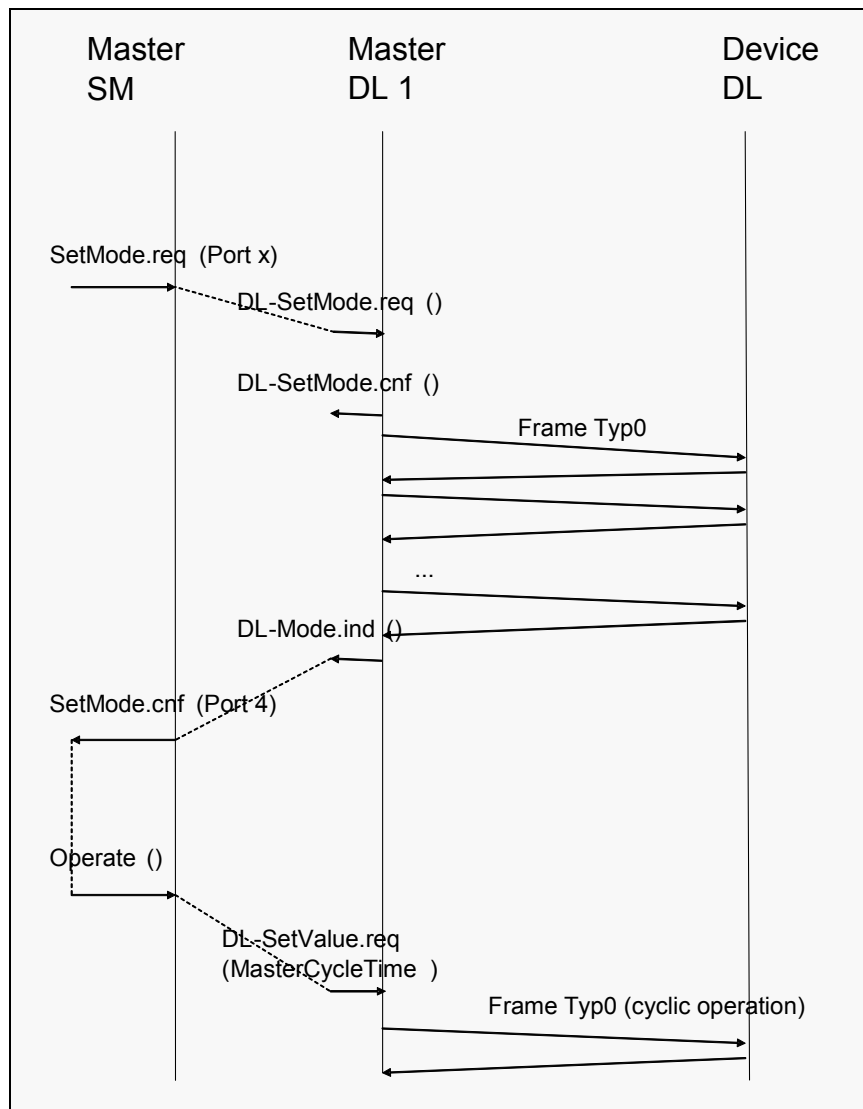


Figure 83 — System management start-up for port 4



**Figure 84 — System management start-up master and device**

## 9.3 Cycle time

### 9.3.1 Operation with preset cycle time

The application uses SM-SetMode to preset the required cycle time for a device. All ports with the same cycle time are synchronous by definition, i.e., the master shall start the frames of these ports in a fixed schedule equidistantly with a tolerance determined by the implementation.

System management sends DL-SetMode.req to prompt the data link layer to set the selected target mode.

The data link layer reads the device's readable communication parameters and sends these immediately to system management with DL-SetMode.cnf or, if the attempt to establish communication succeeds subsequently, with DL-Mode.ind.

If a device's minCycleTime is longer than that preset for this port with SetMode, system management shall send a negative acknowledgment in the form of SM-SetMode.cnf (-) .

If a device's minCycleTime is shorter than the preset value, system management shall check whether the minCycleTime on another device might be causing the value preset for this port with SM-SetMode.req not to be reached, in which case a negative acknowledgment shall be generated in the form of SM-SetMode.cnf (-).

Note Mutual interference is common in the event of differing baud rates or multiplex operation.

Following receipt of all SM-SetMode.cnf, the application shall send SM-Operate.req to prompt system management to activate all ports acknowledged positively with the cycle time preset for this port with SM-SetMode.

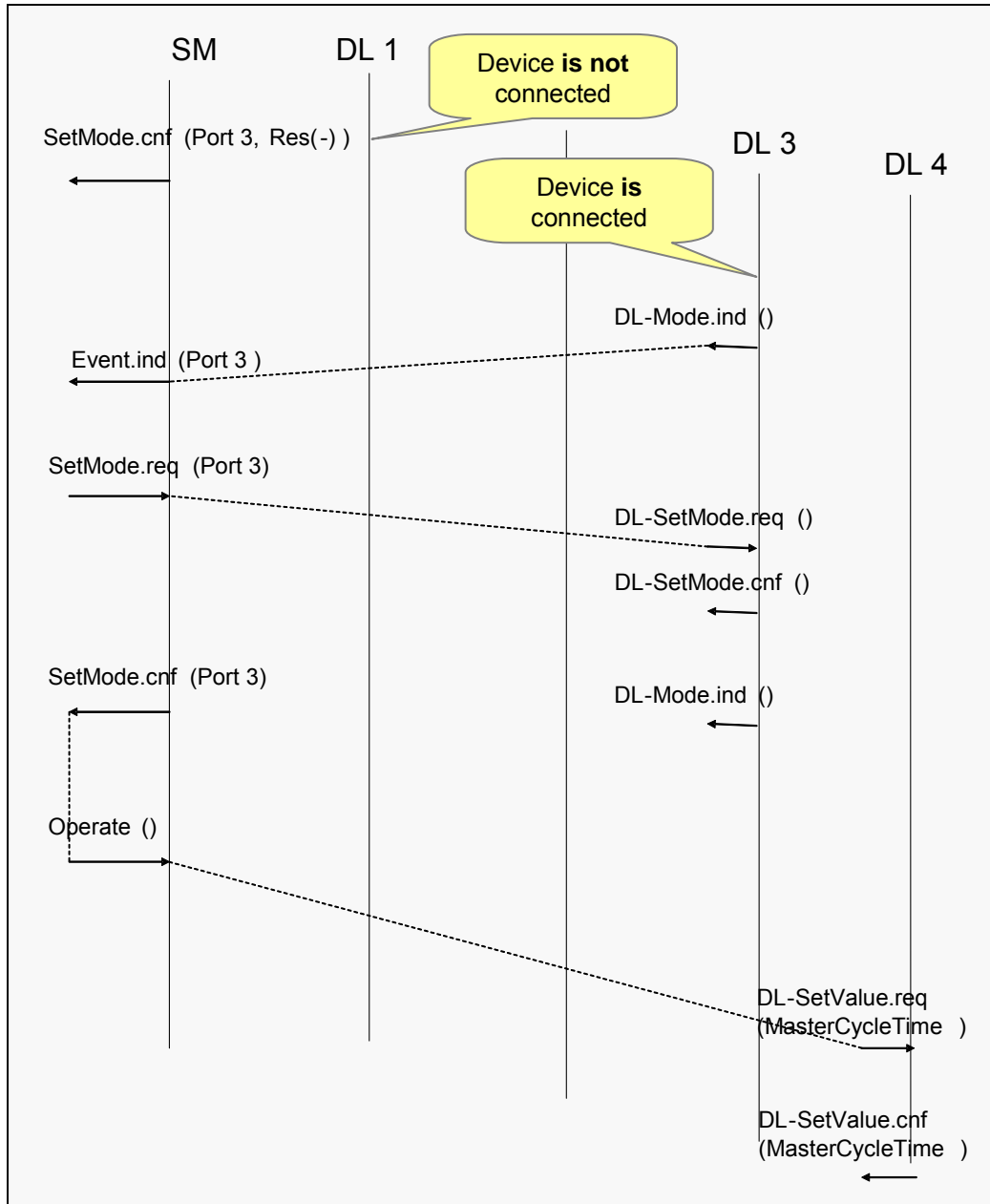
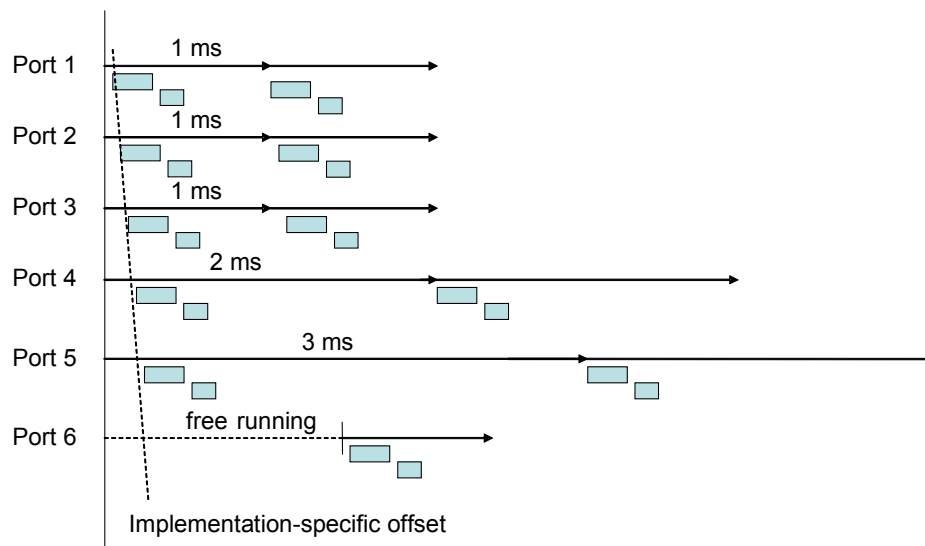


Figure 85 — Cycle time management

Every time an SM-SetMode.req request is sent to activate another device following SM-Operate.req, system management shall check whether the required cycle time can be achieved without affecting ports that are already active. If it cannot, the service shall be confirmed negatively.

Note Activation is subject to a SetOperate.req being received following a positive acknowledgment.





**Figure 86 — Example of port cycle operations**

### 9.3.2 Free running

A cycle time is not preset in SM-SetMode. The actual cycle time of each port is determined by processing on the master. Once the minCycleTime of a free-running device has elapsed, the master should send no more than one frame on each other free-running port before a frame sends to this port. Free-running devices shall not impact upon devices with preset cycle time, i.e., corrupt their cycle time.

### 9.3.3 Frame type

When the transition is made to the OPERATE state, a switch is also made to the frame type relevant for cyclic operation. The frame type to be used in OPERATE is determined by the width of the input and output data. Table 84 lists the rules for the master data link layer in respect of determining the frame type. The input and output data formats shall be read out from the connected device in advance. All defined frame types shall be implemented on a master as it has to be able to support a number of different sensors and actuators simultaneously.

**Table 84 — Use of frame types**

Number of input data bytes	Number of output data bytes	Frame type
0	0	Type 0 Type 1 (NOTE)
1	0	Type 2.1
2	0	Type 2.2
0	1	Type 2.3
0	2	Type 2.4
1	1	Type 2.5
Sum of input data bytes and output data bytes > 2		Type 1
NOTE Type 1 is only used if supported accordingly by the "Frame Capability" parameter.		

## **10 Requirement for certification tests**

In order to prove compatibility with the IO-Link specification, new devices have to undergo interoperability tests. The test scope is defined in a test specification and elaborated in a subgroup of the IO-Link Physics Working Group. This test specification is updated with the findings of regular interoperability meetings.

In the long term, should suitable test tools become available, vendor declarations in respect of compatibility ought to be possible. In order for this to happen, the test tools that would be used need to be acceptance-tested. Work is currently underway on a vendor declaration format.

(Outdated, see F.2.20)

## Annex A Error types

The Error\_Type is used in the parameter ErrorInfo for negative service confirmations at the application layer service interface of the master, when Service PDU transport is used. It indicates the reason of a negative confirmation of a read or write service. The origin of the error may be in the master (local) or in the device (remote).

The Error\_Type consists of the following elements:

- Error Code (1 byte, high byte) and
- Additional Code (1 byte, low byte).

The error code contains information about the error category, the origin and the instance. The permissible error codes are listed in Table A.1, all other Error Code values shall be reserved.

**Table A.1 — Error Codes**

Type	Origin	Name	Category	Mode	Inst.	Value hex.	Remark
PDU buffer overflow	remote	S_PDU_BUFFER	ERROR	SINGLE SHOT	DL	52	Device buffer is too small for storing the complete PDU
PDU checksum error (master)	local	M_PDU_CHECK	ERROR	SINGLE SHOT	DL	56	Calculated PDU checksum in master does not match actual received SPDU
PDU checksum error (device)	remote	S_PDU_CHECK	ERROR	SINGLE SHOT	DL	56	Calculated PDU checksum in device does not match actual received SPDU
PDU flow control error	remote	S_PDU_FLOW	ERROR	SINGLE SHOT	DL	56	Violation of flow control rule during transfer of SPDU between master and device
Illegal PDU service primitive (master)	local	M_PDU_ILLEGAL	ERROR	SINGLE SHOT	AL	57	Unknown service primitive or wrong response e.g. Read Reponse on Write Request
Illegal PDU service primitive (device)	remote	S_PDU_ILLEGAL	ERROR	SINGLE SHOT	AL	58	Unknown service primitive e.g. different protocol revision
Communication error	local / remote	COM_ERR		SINGLE SHOT	unknown	10	Negative service response initiated by a communication error, eg. IO-Link connection interrupted
Device application error	remote	S_APP_DEV	ERROR	SINGLE SHOT	APP	80	Service PDU transferred, but not processed due to device error. See error details in Additional Code

The Additional Code may be used to indicated a detailed description of the error. The Additional Code value 0 shall be used, if no additional details have to be indicated.

The permissible additional codes for error type S\_APP\_DEV are listed in Table A.2, all other Additional Code values are reserved.

**Table A.2 — Additional Codes for Error Code S\_APP\_DEV**

Type	Value hex.	Remark
No details	00	
Index not available	11	
Subindex not available	12	
Service temporarily not available	20	
Service temporarily not available, local control	21	local operation at device inhibits access
Service temporarily not available, device control	22	device state does not allow access. e.g. during teach or calibration
Access denied	23	e.g. write on read-only parameter
Parameter value out of range	30	
Parameter value above limit	31	
Parameter value below limit	32	
Interfering parameter	40	collision with other parameter values
Application failure	81	e.g. separate application and communication processors
Application not ready	82	e.g. separate application and communication processors

## Annex B Event codes

The event codes shall be used for event indications, when detailed event information is available according to the signaled status code. As specified in clause 7.2.4.4.2.2 the event entry contains an event code in addition to the event qualifier. The event code contains the ID for an actual event. Permissible values for EventCode are listed in Table B.1, all other EventCode values shall be reserved.

**Table B.1 — Event Codes**

Code hex.	Meaning
0000	no malfunction
1000	general malfunction
1800 .. 1FFF	general malfunction (manufacturer specific)
4000	temperature
4100	ambient temperature
4110	excess ambient temperature
4120	too low ambient temperature
4200	temperature device
4210	excess temperature device
4220	too low temperature device
4300	temperature periphery
4310	excess temperature periphery
4320	too low temperature periphery
5000	device hardware
5010	component malfunction
5100	supply
5110	supply low voltage
5111	U1 = supply +/- 15 V
5112	U2 = supply + 24 V
5113	U3 = supply + 5 V
5114 .. 5119	U4 .. U9 = manufacturer specific
5151	short circuit
5160	supply periphery
5200	control
5300	operating unit
5400	power section
5410	output stages
5450	fuses
5451	S1 = L1
5452	S2 = L2
5453	S3 = L3
5454 .. 5459	S4 .. S9 = manufacturer specific
5500	additional communication
5510	communication interface no. 1
5520	communication interface no. 2
6000	device software

Code hex.	Meaning
6010	software reset (watchdog)
6100	internal software
6300	data record
6310	loss of parameter
6320	parameter error
6330	parameter not initialized
6340	parameter non specific
6350	parameter changed
8000	monitoring
8100	communication
8110	process data monitoring
8C00	application
8C10	excess process variable range
8C20	excess measurement range
8C30	too low process variable range
8C40	advance warning
8CA0 .. 8CFF	manufacturer specific
8D00 .. 8DFF	user defined (customer specific mapping of events)
9000	external malfunction
FF00 .. FFFF	IO-Link specific event codes (see Table B.2)

Table B.2 gives an overview about encoding of typical events in the context of process data, application and others. This overview does not claim to be complete.

**Table B.2 — Exemplary event data**

Type	Origin	Instance	Type	Mode	Event code	Value	Action	Remark
Device PD invalid flag	remote	APP	ERROR	appear	PD_INVALID	0xFF0	PD stop Event.ind	PD invalid event from device
Device PD invalid flag	remote	APP	ERROR	disappear	PD_INVALID	0xFF0	PD run Event.ind	PD invalid event disappears from device
<b>No Details</b>								
Device Message	remote	APP	MESSAGE	single shot	DEVICE	0xFF80	Event.ind	
Device Warning	remote	APP	WARNING	single shot	DEVICE	0xFF80	Event.ind	
Device Error	remote	APP	ERROR	single shot	DEVICE	0xFF80	Event.ind PD stop	
Parameter Error	remote	APP	ERROR	single shot	PARAMETER	0x6320	Event.ind	from application specific listing
Communication Error	unknown	unknown	ERROR	single shot	COMM	0xFF10	Event.ind PD stop	
<b>With Details</b>								
<b>Process Data</b>								

Type	Origin	Instance	Type	Mode	Event code	Value	Action	Remark
Address increment error	remote	DL	ERROR	single shot	PD_INCR	0xFFE1	PD stop	missing PD address
Access outside PD length	remote	DL	ERROR	single shot	PD_LEN	0xFFE2	PD stop	written or read PD length too long
Incomplete PD length	remote	DL	ERROR	single shot	PD_SHORT	0xFFE3	PD stop	written or read PD length too short
Read at input length 0	remote	DL	ERROR	single shot	NO_PDIN	0xFFE4	PD stop	
Write at input length 0	remote	DL	ERROR	single shot	NO_PDOU T	0xFFE5	PD stop	
<b>System Management</b>								
Mode Indication	local	DL	MESSAGE	single shot	NEW_SLAVE	0xFF21	PD stop	
<b>Unspecified</b>								
Incorrect event signaling	local	DL	ERROR	single shot	EVENT	0xFF31	Event.ind	
<b>Application</b>								
Store Parameter	remote	APP	MESSAGE	single shot	STORE_PARAMETER	0xFF91	Event.ind	
Restore Parameter	remote	APP	MESSAGE	single shot	RESTORE_PARAMETER	0xFF92	Event.ind	
Com Startup request	remote	APP	MESSAGE	single shot	RESTART_COM	0xFF98	Event.ind	detailed functionality TBD
Application specific	remote	APP	MESSAGE WARNING ERROR	single shot appear disappear		0xFFFF	specific	see Table B.1 for application specific listing

## Annex C Data object addressing by Service PDU Index

Service PDU transfer may be used to exchange application data objects. The index of the Service PDU is used to address the data objects. Data objects, which are accessible at the direct parameter page may be optionally provided via Service PDU additionally. Figure C.1 illustrates the general mapping of data objects for direct parameter access and Service PDU access.

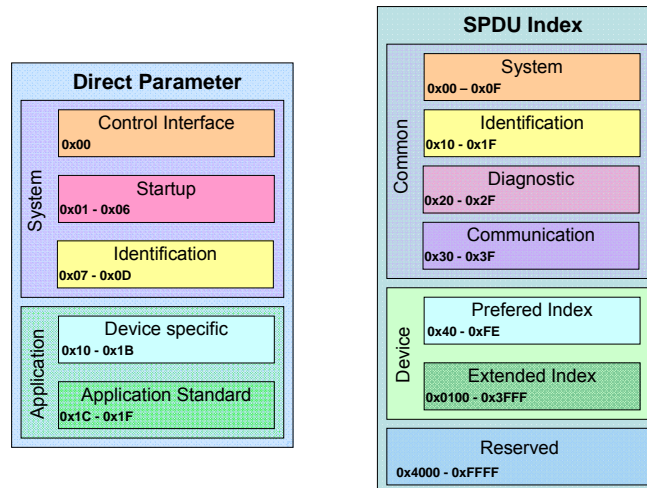


Figure C.1 — General mapping of data objects

Table C.1 defines the assignment of data objects to the index range of Service PDUs.

Table C.1 — Index definition for Service PDU

Index	Object name	Access	Length	Data type	M/O	Remark
0000	Direct Parameter Page Direct Parameter 0 - 15	R	16 byte	Record	M	direct parameter address 0 - 15 Master: mandatory Device: not implemented
0001	Direct Parameter Page Direct Parameter 16 - 31	R/W	16 byte	Record	M	direct parameter address 16 – 31 Master: mandatory Device: not implemented
0002	System Command	W	1 byte	Octet	O	Command Code Definition Public: 00h - 9Fh Vendor specific : A0h - FFh (see Table C.2)
0003-000F	reserved					
0010	Vendor Name	R	max. 64 byte	Visible-String	M	
0011	Vendor Text	R	max. 64 byte	Visible-String	O	
0012	Product Name	R	max. 64 byte	Visible-String	M	
0013	Product ID	R	max. 64 byte	Visible-String	O	e.g. order number
0014	Product Text	R	max. 64 byte	Visible-String	O	e.g. device function
0015	Serial number	R	max. 16 byte	Visible-String	O	Vendor specific serial number
0016	Hardware Revision	R	max. 64 byte	Visible-String	O	Vendor specific format



Index	Object name	Access	Length	Data type	M/O	Remark
0017	Firmware Revision	R	max. 64 byte	Visible-String	O	Vendor specific format
0018	Application Specific Name	R/W	max. 64 byte	Visible-String	O	e.g. location name defined by customer
0019	Parameter ID	R	4 byte	Octet string	O	Unique ID for actual parameter set
001A	Storage Parameter	R	variable	Record	O	Set of data objects for storage
001B-001F	reserved					
0020	Error count	R	2 Byte	Unsigned16	O	errors since power-up or reset
0021	Last Event	R	3 Byte	Record	O	copy from diagnostic page Event Mode <11 = inactive, 11=active
0022	Diagnostic history short	R	4 Byte	Record	O	events with continuous event number
0023	Diagnostic history long	R	variable	Record	O	event information with e.g. time stamp
0024-0027	reserved					
0028	Process Data Input	R	PD length	Unsigned	O	read last valid process data from PDin channel
0029	Process Data Output	R/W	PD length	Unsigned	O	write process data to Pdout channel read last valid process data from Pdout channel
002A-003F	reserved					
0040-00FE	device specific				O	Preferred index range 40h .. FE addressed via 8 bit index
00FF	reserved					
0100-3FFF	device specific				O	Extended index range 100h .. 7FFF addressed via 16 bit index
4000-FFFF	reserved					

Table C.2 lists the permissible values of the data object system Command.

**Table C.2 — Values of System Command**

Command	Object name	M/O	Remark
00 .. 59	reserved		
5A	Fallback	M	only for devices with support of SIO mode
5B .. 7F	reserved		
80	Device reset	O	
81	Application reset	O	
82	Restore factory setting	O	
83 .. 8C	reserved		
8D	Parameter storage done	O	required for data storage model (NOTE)
8E	Parametrization end	O	required for data storage model (NOTE)
8F	Parametrization start	O	required for data storage model (NOTE)

---

Command	Object name	M/O	Remark
90 .. 96	reserved		
97	ComStop	M	return to communication startup
98	PDOOut valid	O	mandatory for actuator
99	Device Operate	M	goto cyclic PD transfer
9A .. 9F	reserved		
A0 .. FF	vendor specific		
NOTE reserved for future IO-Link versions.			

## Annex D Example Service PDU transmission

Figure D.1 illustrates an example for the transmission of a Service PDU using a Read service with a 16-bit index and subindex for 19 bytes of user data with mapping to a DL frame type 2 for sensors and with interruption in the event of an event transmission.

Cycle	Action (related to oi-requestdata)	Master					Device					Action (related to oi-requestdata)	
		CMD			CHK/TYPE		On-request	PROCESS	CHK/STAT				
		R/W	Ch.	Flow	Typ	CHK			STAT	CHK			
				CTRL			E	r.					
		(bin)	(hex)	(bin)									
1	sends	0	11	10	10	XX	1011 0101		xx xx	0	0	XX	receives
2	sends	0	11	01	10	XX	Index(hi)		xx xx	0	0	XX	receives
3	sends	0	11	02	10	XX	Index(lo)		xx xx	0	0	XX	receives
4	sends	0	11	03	10	XX	Subindex		xx xx	0	0	XX	receives
5	sends	0	11	04	10	XX	CHKPDU		xx xx	0	0	XX	receives
6	receives	1	11	10	10	XX		0000 0001	xx xx	0	0	XX	Slave „busy“
7	receives	1	11	10	10	XX		0000 0001	xx xx	0	0	XX	Slave „busy“
8	receives	1	11	10	10	XX		0000 0001	xx xx	0	0	XX	Slave „busy“
9	receives	1	11	10	10	XX		0000 0001	xx xx	0	0	XX	Slave „busy“
10	receives	1	11	10	10	XX		0000 0001	xx xx	0	0	XX	Slave „busy“
11	receives	1	11	10	10	XX		1101 0001	xx xx	0	0	XX	sends
12	receives	1	11	01	10	XX		0001 0110	xx xx	0	0	XX	sends
13	receives	1	11	02	10	XX		Data 1	xx xx	0	0	XX	sends
14	receives	1	11	03	10	XX		Data 2	xx xx	0	0	XX	sends
15	receives	1	11	04	10	XX		Data 3	xx xx	0	0	XX	sends
16	receives	1	11	05	10	XX		Data 4	xx xx	0	0	XX	sends
17	receives	1	11	06	10	XX		Data 5	xx xx	0	0	XX	sends
18	receives	1	11	07	10	XX		Data 6	xx xx	0	0	XX	sends
19	receives	1	11	08	10	XX		Data 7	xx xx	0	0	XX	sends
20	receives	1	11	09	10	XX		Data 8	xx xx	0	0	XX	sends
21	other channel	X	XX		X	XX	?? ??	?? ??	xx xx	1	0	XX	other channel
22	other channel	X	XX		X	XX	?? ??	?? ??	xx xx	1	0	XX	other channel
23	other channel	X	XX		X	XX	?? ??	?? ??	xx xx	1	0	XX	other channel
24	other channel	X	XX		X	XX	?? ??	?? ??	xx xx	0	0	XX	other channel
25	other channel	X	XX		X	XX	?? ??	?? ??	xx xx	0	0	XX	other channel
26	other channel	X	XX		X	XX	?? ??	?? ??	xx xx	0	0	XX	other channel
27	receives	1	11	0A	10	XX		Data 9	xx xx	0	0	XX	sends
28	receives	1	11	0B	10	XX		Data 10	xx xx	0	0	XX	sends
29	receives	1	11	0C	10	XX		Data 11	xx xx	0	0	XX	sends
30	receives	1	11	0D	10	XX		Data 12	xx xx	0	0	XX	sends
31	receives	1	11	0E	10	XX		Data 13	xx xx	0	0	XX	sends
32	receives	1	11	0F	10	XX		Data 14	xx xx	0	0	XX	sends
33	receives	1	11	00	10	XX		Data 15	xx xx	0	0	XX	sends
34	receives	1	11	01	10	XX		Data 16	xx xx	0	0	XX	sends
45	receives	1	11	02	10	XX		Data 17	xx xx	0	0	XX	sends
36	receives	1	11	03	10	XX		Data 18	xx xx	0	0	XX	sends
37	receives	1	11	04	10	XX		Data 19	xx xx	0	0	XX	sends
38	receives	1	11	05	10	XX		CHKPDU	xx xx	0	0	XX	sends
39	processes	1	11	11	10	XX	?? ??	0000 0000	xx xx	0	0	XX	Slave „idle“
40	other channel	X	XX	X	10	XX	?? ??	?? ??	xx xx	0	0	XX	other channel

Figure D.1 — Example for Service PDU transmission

### Annex E Residual fault probability of the telegram checksum

Block signals are used. This results in a residual fault probability (RFP) as illustrated in Figure E.1 (see F.2.21).

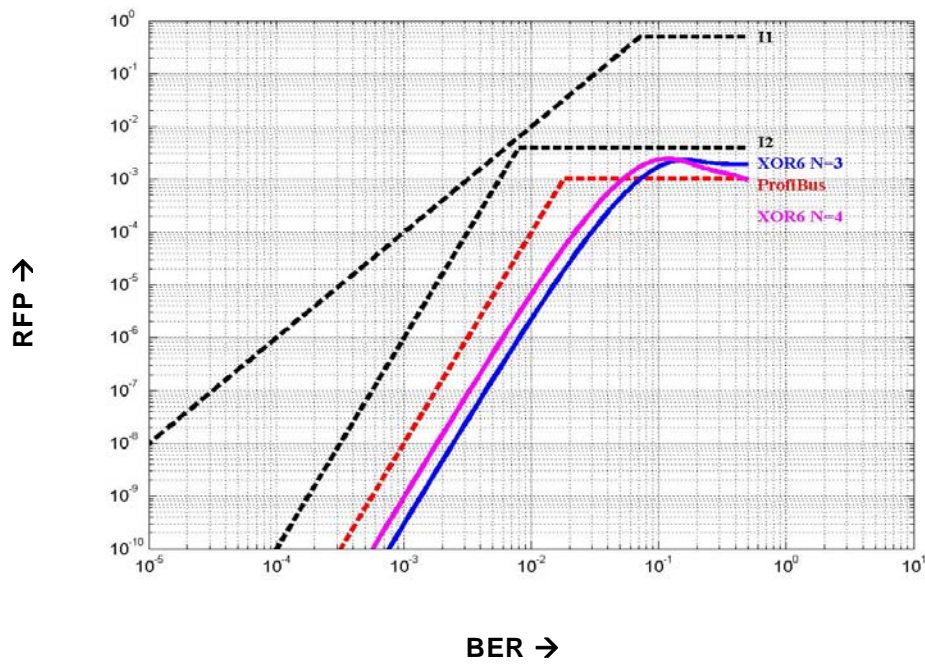


Figure E.1 — Residual fault probability

## Annex F Supplements and errata

### F.1 IO-Link device

#### F.1.1 Requirements for the application

- The devices shall always behave the same independent of the sequence of parameterization.
- The PDU length shall be such that records (over subindex 0) can be read out completely.
- 15Bit Index
- Inconsistencies during reparameterization shall be avoided

An application that carries out a device parameterization starts and ends this procedure with a SystemCommand “Parameterization started (0x8f)” and “Parameterization ended (0x8e)”. Afterwards, the device may carry out a testing and eventually send a negative response with respective error codes.

#### F.1.1.1 Parameter objects

##### Variable

A variable is a data object which is addressed by an index and is of a certain data type. Variables may be structured and may be described as array or record.

Indices of standardized variables which are optional may only be applied as is specified.

##### Array

An array, which is addressed by the index, is a data structure with elements of a certain data type. Certain elements are addressable by the subindex. Subindex =0 addresses the whole data object.

##### Record

A record, which is addressed by the index, is a data structure with elements of different data types. Certain elements are addressable by the subindex. Subindex =0 addresses the whole data object.

If the attribute ‘bitOffset’ is used:

- It shall be applied to all ‘RecordItems’
- The bitOffsets must increase their values monotone

The order of the RecordItems in the XML-description equals the order of the RecordItems in the record data representation.

There are no additional restrictions for subindex values.

#### Examples:

Parameter: Setpoint

Object type: Variable

Index	Subindex	Data	Data Type	Name
84	0	2323	UIntegerT, bitLength = 32	Setpoint

Parameter: Setpoint array

Object type: Array

Index	Subindex	Data	Data Type	Name
66	1	23234455	UIntegerT, bitLength = 32	Setpoint 1
	2	23235543	UIntegerT, bitLength = 32	Setpoint 2

Parameter: Limit value

Object type: Record

Index	Subindex	Data	Data Type	Name
46	1	23 45	UIntegerT, bitLength = 16	Status
	2	h a l l o x0 x0	StringT, fixedLength = 7	Text
	3	56 12 22 34	UIntegerT, bitLength = 32	Value

Variables may be described as records. With access to the subindex, individual Recorditems may be addressed on certain bit positions and described. Nearby bits are not changed during the description process.

Example:

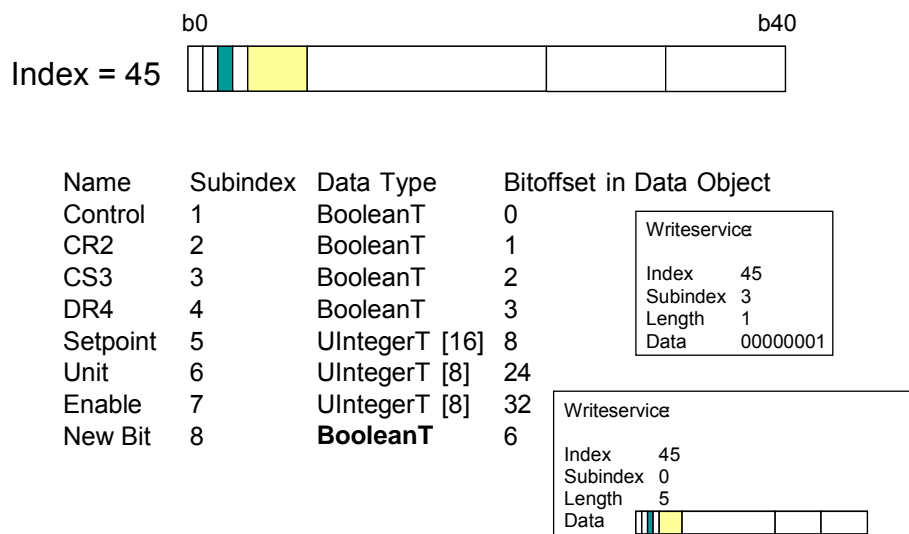


Figure F.1 — Example data structure

F.1.1.2 Data Types

F.1.1.2.1 BooleanT

True



False



Coding within a byte:

True

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

False

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

#### F.1.1.2.2 UIntegerT (bitLength = 2...64)

Coding:

UIntegerT bitLength = 4, Value 13

1	1	0	1
---	---	---	---

Mapped on 1 byte

0	0	0	0	1	1	0	1
---	---	---	---	---	---	---	---

A UIntegerT with bitLength 4 will be right-aligned, and extended to 8 bit

- UIntegerT (bitLength = 2...8) is mapped on 1 byte
- UIntegerT (bitLength = 9...16) is mapped on 2 bytes
- UIntegerT (bitLength = 17...32) is mapped on 4 bytes
- UIntegerT (bitLength = 33...64) is mapped on 8 bytes

#### F.1.1.2.3 IntegerT (bitLength = 2...64)

Coding:

IntegerT bitLength = 4, Value -4

1	1	0	0
---	---	---	---

Mapped on 1 byte

1	1	1	1	1	1	0	0
---	---	---	---	---	---	---	---

An IntegerT with bitLength 4 will be right-aligned and extended correctly signed to 8 bit

- IntegerT (bitLength = 2...8) is mapped on 1 byte
- IntegerT (bitLength = 9...16) is mapped on 2 bytes
- IntegerT (bitLength = 17...32) is mapped on 4 bytes
- IntegerT (bitLength = 33...64) is mapped on 8 bytes

**F.1.1.2.4 StringT (fixedLength max. = 232) UTF 8**

Coding:

StringT, fixedLength = 7

h	a	l	l	o	x0	x0
---	---	---	---	---	----	----

Normally, strings shall be transmitted in their fixedLength. If the string is shorter, it shall be zero-terminated.

The string transmission may be optimized when:

- The string is the only data object transmitted by a Service PDU
- The string variable does not have the attribute 'fixedLengthRestriction'

In case of optimization, the string will be transmitted in its actual length and without zero-termination. The string length is deduced from the SPDU-length.

Example:

```
Type rec1 {
    UInteger8    Var1; // length is given by datatype
    String [5]   Var2; // length is given in IODD by attribute 'fixedLength'
    UInteger32   Var3; // length is given by datatype
}
```

Access via Subindex 0:

Var1	Var2					Var3			
0xAA	I	O	L	0x00	0x00	0x01	0x02	0x03	0x04

Access via Subindex 2:

Var2		
I	O	L

**F.1.1.2.5 OctetStringT (fixedLength max. = 232)**

Coding:

OctetStringT, fixedLength = 7

1f	0a	23	AA	BB	A1	D0
----	----	----	----	----	----	----

**F.1.1.2.6 Float32T**

Coding (refer to ANSI/IEEE 754):



SN: sign 0 = positive, 1 = negative

Bits	7	6	5	4	3	2	1	0
Octets 1	Exponent (E)							
	SN	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$
Octets 2	(E)	Fraction (F)						
	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	$2^{-5}$	$2^{-6}$	$2^{-7}$
Octets 3	Fraction (F)							
	$2^{-8}$	$2^{-9}$	$2^{-10}$	$2^{-11}$	$2^{-12}$	$2^{-13}$	$2^{-14}$	$2^{-15}$
Octets 4	Fraction (F)							
	$2^{-16}$	$2^{-17}$	$2^{-18}$	$2^{-19}$	$2^{-20}$	$2^{-21}$	$2^{-22}$	$2^{-23}$

**F.1.1.2.7 TimeT IEC 61158-6**

Coding:

bits octets	7	6	5	4	3	2	1	0		
1	$2^{31}$	$2^{30}$	$2^{29}$	$2^{28}$	$2^{27}$	$2^{26}$	$2^{25}$	$2^{24}$	Seconds since 1.1.1900 0.00,00 or since 7.2. 2036 6.28,16 when time value less than 0x9dff4400.00000000	
2	$2^{23}$	$2^{22}$	$2^{21}$	$2^{20}$	$2^{19}$	$2^{18}$	$2^{17}$	$2^{16}$		
3	$2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$		
4	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$		
5	$2^{31}$	$2^{30}$	$2^{29}$	$2^{28}$	$2^{27}$	$2^{26}$	$2^{25}$	$2^{24}$	Fraction Part of seconds one unit is $1/(2^{32})$ s	
6	$2^{23}$	$2^{22}$	$2^{21}$	$2^{20}$	$2^{19}$	$2^{18}$	$2^{17}$	$2^{16}$		
7	$2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$		
8	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$		
	msb							lsb		

**F.1.1.2.8 TimeSpanT IEC 61158-6**

Coding:

bits octets	7	6	5	4	3	2	1	0		
1	$2^{31}$	$2^{30}$	$2^{29}$	$2^{28}$	$2^{27}$	$2^{26}$	$2^{25}$	$2^{24}$	Seconds as Integer32	
2	$2^{23}$	$2^{22}$	$2^{21}$	$2^{20}$	$2^{19}$	$2^{18}$	$2^{17}$	$2^{16}$		
3	$2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$		
4	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$		
5	$2^{31}$	$2^{30}$	$2^{29}$	$2^{28}$	$2^{27}$	$2^{26}$	$2^{25}$	$2^{24}$	Fraction Part of seconds one unit is $1/(2^{32})$ s	
6	$2^{23}$	$2^{22}$	$2^{21}$	$2^{20}$	$2^{19}$	$2^{18}$	$2^{17}$	$2^{16}$		
7	$2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$		
8	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$		
	msb							lsb		

## F.2 Errata for Version 1.0 dated November 2007

### F.2.1 Clause 6.2.2.2 Table 4

Transmission parameters: Parameters 'r' and 's' are unitless.

### F.2.2 Clause 6.2.4.1 Pin2/5

Port Class B is now defined as a 5 Pin M12 connector. Its Pin 2 and Pin 5 are used for actuator power supply (Figure F.2).

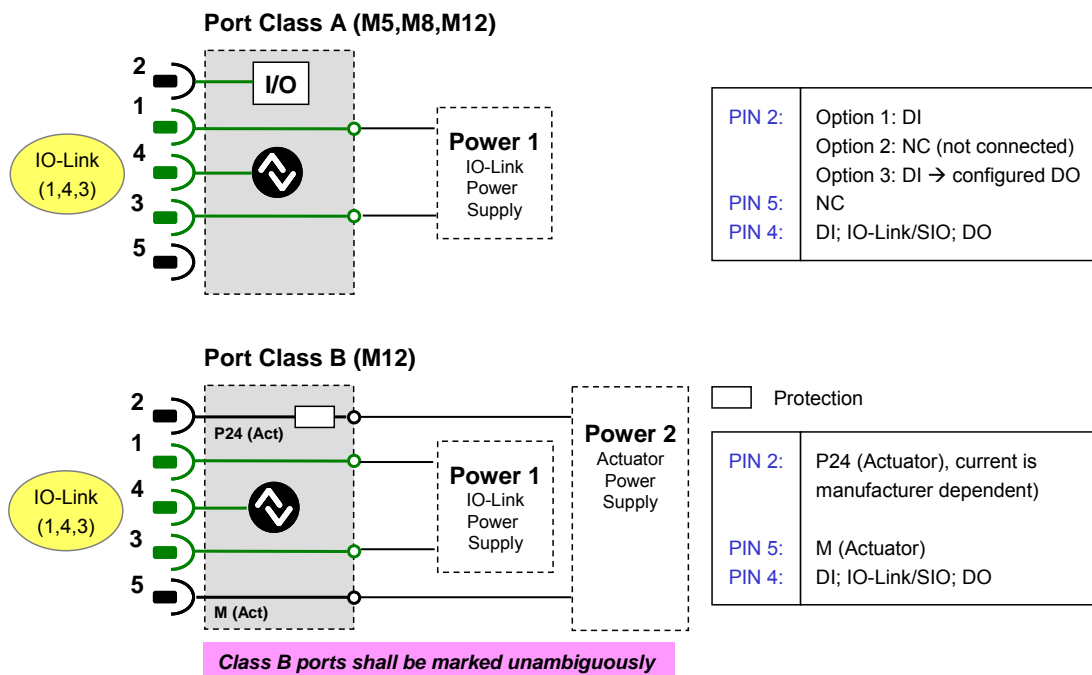


Figure F.2 — Class B port definitions

### F.2.3 Clause 7.1.10 DL-SetMode

Extend description for RealMode:

#### RealMode

Values see 7.1.11 DL-Mode

### F.2.4 Clause 7.1.11 DL-Mode

Change parameter type:

#### RealMode

Parameter type: Enum

### F.2.5 Clause 7.2.4.1 Figure 41 (formerly 31)

'EventFlag' should be 'EventFlag.ind' as in text.

### F.2.6 Clause 7.2.4.3 Figures 42-44 (formerly 32-34)

See Figure F.3 for changes in Figure 42.

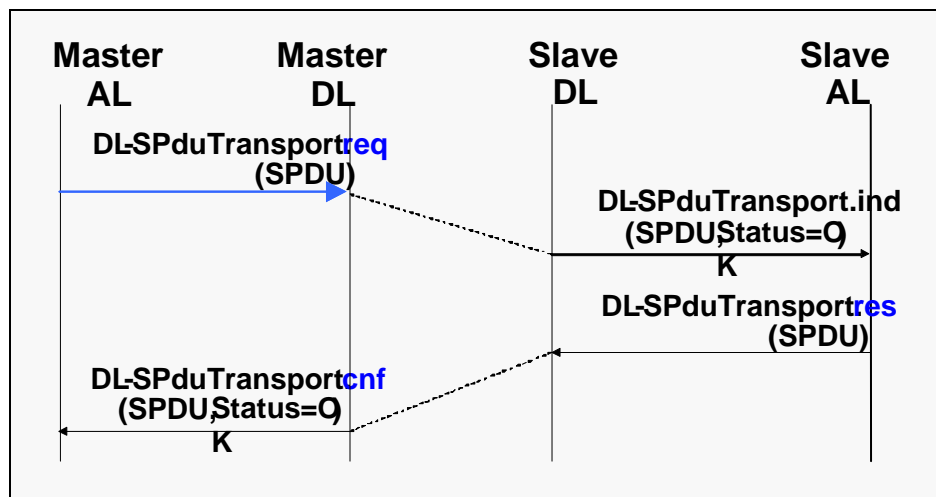


Figure F.3 — Changes in figure 42

Figures 43 and 44: same changes as in Figure 42

### F.2.7 Clause 7.2.4.3.1 Table 45 (formerly 43)

PDU\_RESPONSE: transition to PDU\_COMPLETE is missing

Add text: If the master data link layer detects a valid SPDU transmission, it shall switch to PDU\_Complete

Change to: NOTE 1 ... is a default telegram if no on-request data is transmitted.'

### F.2.8 Clause 7.2.4.3.3

Note should be 'Annex C'

### F.2.9 Clause 7.2.4.3.4 Figure 47 (formerly 37)

'\*\*\*' (stars) not explained. They will be deleted in future releases.

### F.2.10 Clause 7.2.4.4.2.5 Table 52 (formerly 50)

Name should be 'Event channel addresses'?

### F.2.11 Clause 7.2.5.3 Figure 54 (formerly 44)

Rename Master/Device frame → Master/Device telegram

### F.2.12 Clause 7.2.5.5.2

Clause 0 → Clause 6.2.2.2

### F.2.13 Clause 7.2.5.5.6

"TFRAME" → "tFRAME"

### F.2.14 Clause 7.2.5.5.9

"TOFS" → "tOFS"

**F.2.15 Clause 8.2.2 Figure 70 (formerly 60)**

'Zykluszeit' → 'Cycle time'

**F.2.16 Clause 8.2.3.2 Figure 74 (formerly 63)**

'write.res' → 'write.rsp'

**F.2.17 Clause 8.2.4.1.3 Table 76 (formerly 74)**

Replace table with the following Table F.1:

**Table F.1 — Definition for Length**

Service	Length	Extended length	Meaning
0	0	n/a	No service, PDU length is 1.
0	1	n/a	Device busy, PDU length is 1.
0	2 ..15	n/a	reserved
1 .. 15	0	n/a	reserved
1 .. 15	1	17 to 232	Length of Service PDU in extended length
1 .. 15	1	233 to 255	reserved
1 .. 15	2 to 15	n/a	Length of Service PDU

**F.2.18 Clause 8.2.4.2 Figure 80 (formerly 69)**

'\*.res' → '\*.rsp'

**F.2.19 Clause 9.1.2 Target Mode, Real Mode**

'DigitalINPUT', 'DigitalOUT(T)PUT' are equal to 'DigInput' and 'DigOutput'

**F.2.20 Clause 10**

Ever since the IO-Link Consortium published the V1.0 of this specification in November 2007 it emphasized the quality issues of IO-Link systems. The EMC requirements in 6.3 have been elaborated and test specifications for IO-Link devices and masters are available for download from the IO-Link portal [www.io-link.com](http://www.io-link.com). Normalized test equipment for IO-Link devices can be acquired; those for IO-Link masters are in preparation.

Thus the IO-Link Consortium requires the manufacturers to test their products according to the test specifications and to provide a manufacturer's declaration of conformity according to the layout proposal presented in Annex G together with the product and its associated mandatory IODD (IO-Link device description) file.

**F.2.21 Annex E Figure E.1**

Add key:

I1 and I2 are data (safety?) integrity levels (see IEC 61508)


IO-Link uses 'XOR6 N-4'

BER = Bit Error Rate

RFP = Residual Fault Probability (formerly "RFP")

## Annex G Layout for a manufacturer's declaration of conformity

Figure G.1 shows a layout proposal for a manufacturer's conformity declaration.

	<b>MANUFACTURER'S DECLARATION OF CONFORMITY</b>	<i>(Company logo)</i>
<b>WE:</b>	<i>Company's name and address</i>	
	<b>declare under our own responsibility that the product(s):</b>	
	<i>Trademark, IO-Link product types (annotate "IO-Link Master" or "IO-Link Device")</i>	
	<b>to which this declaration refers conform to:</b>	
	<ul style="list-style-type: none"> <li>• IO-Link Communication Specification V1.0, January 2009</li> <li>• IO-Link Device Description V1.0, January 2009</li> <li>• IEC 61000-6-2 (IEC 61131-2) (NOTE 1)</li> <li>• IEC 60947-5-2 (NOTE 2)</li> </ul>	
	<b>The conformity tests are documented in the test report:</b>	
	<i>Testreport identification</i>	
	<b>Issued at</b> <i>location, date</i>	<b>Authorized signatory</b>
	<b>Name:</b>	<i>First, last name</i>
	<b>Title:</b>	<i>Job title</i>
	<b>Signature:</b>	<i>Signature</i>
Reproduction and all distribution without written authorization prohibited		

NOTE 1 Relevant EMC standards in case of an IO-Link master

NOTE 2 Relevant EMC standard in case of an IO-Link device

**Figure G.1 — Layout proposal for a manufacturer's declaration**

© Copyright by:

IO-Link Consortium  
Haid-und-Neu-Str. 7  
76131 Karlsruhe  
Germany

Phone: +49 (0) 721 / 96 58 590

Fax: +49 (0) 721 / 96 58 589

e-mail: [info@io-link.com](mailto:info@io-link.com)

<http://www.io-link.com/>

