IO-Link
Interface and System

Specification

V1.1.5
October 2025

Order No: 10.002

File name: **IOL-Interface-Spec_10002_V1.1.5_Oct2025.docx**

The IO-Link technology is standardized in IEC 61131-9 Edition 2. The IO-Link Community is a D-Liaison member in the corresponding IEC working group.

Any comments, proposals, requests on this document are appreciated. Please use www.io-link-projects.com for your entries and provide name and email address.
Login: **IO-Link-V113**
Password: **Report**

Publisher:

IO-Link Community
c/o PROFIBUS Nutzerorganisation e.V.
Ohiostrasse 8
76149 Karlsruhe
Germany
Phone: +49 721 / 98 61 97 0
Fax: +49 721 / 98 61 97 11
E-mail: info@io-link.com
Web site: www.io-link.com

Conventions:

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

shall:	indicates a mandatory requirement. Designers shall implement such mandatory requirements to ensure interoperability and to claim conformity with this specification.
should:	indicates flexibility of choice with a strongly preferred implementation.
can:	indicates flexibility of choice with no implied preference (possibility and capability).
may:	indicates a permission.
highly recommended:	indicates that a feature shall be implemented except for well-founded cases. Vendor shall document the deviation within the user manual and within the manufacturer declaration.

CONTENTS

INTRODUCTION

0.1 General

IEC 61131-9 is part of a series of standards on programmable controllers and the associated peripherals and should be read in conjunction with the other parts of the series.

Where a conflict exists between this and other IEC standards (except basic safety standards), the provisions of this standard should be considered to govern in the area of programmable controllers and their associated peripherals.

The increased use of micro-controllers embedded in low-cost sensors and actuators has provided opportunities for adding diagnosis and configuration data to support increasing application requirements.

The driving force for the SDCI (IO-Link™[1])) technology is the need of these low-cost sensors and actuators to exchange this diagnosis and configuration data with a controller (PC or PLC) using a low-cost, digital communication technology while maintaining backward compatibility with the current DI/DO signals.

In fieldbus concepts, the SDCI technology defines a generic interface for connecting sensors and actuators to a Master unit, which may be combined with gateway capabilities to become a fieldbus remote I/O node.

Any SDCI compliant Device can be attached to any available interface port of the Master. SDCI compliant Devices perform physical to digital conversion in the Device, and then communicate the result directly in a standard format using "coded switching" of the 24 V I/O signalling line, thus removing the need for different DI, DO, AI, AO modules and a variety of cables.

Physical topology is point-to-point from each Device to the Master using 3 wires over distances up to 20 m. The SDCI physical interface is backward compatible with the usual 24 V I/O signalling specified in IEC 61131-2. Transmission rates of 4,8 kbit/s, 38,4 kbit/s and 230,4 kbit/s are supported.

The Master of the SDCI interface detects, identifies and manages Devices plugged into its ports.

Tools allow the association of Devices with their corresponding electronic I/O Device Descriptions (IODD) and their subsequent configuration to match the application requirements.

The SDCI technology specifies three different levels of diagnostic capabilities: for immediate response by automated needs during the production phase, for medium term response by operator intervention, or for longer term commissioning and maintenance via extended diagnosis information.

The structure of this standard is described in 4.8.

Conformity with IEC 61131-9 cannot be claimed unless the requirements of Annex H are met.

Terms of general use are defined in IEC 61131-1 or in the IEC 60050 series. More specific terms are defined in each part.

0.2 Patent declaration

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning the point-to-point serial communication interface for small sensors and actuators as follows, where the [xx] notation indicates the holder of the patent right:

---

1 IO-Link™ is a trade name of the "IO-Link Community". This information is given for the convenience of users of this international Standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this standard does not require use of the registered logos for IO-Link™. Use of the registered logos for IO-Link™ requires permission of the "IO-Link Community".

DE 102 119 39 A1 US 2003/0200323 A1	[SK]	Coupling apparatus for the coupling of devices to a bus system
DE10201100203883	[SK]	Filling level sensor for determination of filling level in toroidal container, has evaluation unit determining total filling level measurement value, and total filling level output outputting total filling level measurement values
DE102016114600B3	[SK]	IO-Link capable sensor and method of communication
DE202016104342U1	[SK]	IO-Link-capable sensor

IEC takes no position concerning the evidence, validity and scope of these patent rights.

The holders of these patents' rights have assured the IEC that they are willing to negotiate licences either free of charge or under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with IEC.

Information may be obtained from:

[SK]	Sick AG Waldkirch Germany

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. IEC shall not be held responsible for identifying any or all such patent rights.

ISO (www.iso.org/patents) and IEC (http://patents.iec.ch) maintain on-line data bases of patents relevant to their standards. Users are encouraged to consult the databases for the most up to date information concerning patents.

PROGRAMMABLE CONTROLLERS —

Part 9: Single-drop digital communication interface for small sensors and actuators (SDCI)

1 Scope

This part of IEC 61131 specifies a single-drop digital communication interface technology for small sensors and actuators SDCI (commonly known as IO-Link™[2]), which extends the traditional digital input and digital output interfaces as defined in IEC 61131-2 towards a point-to-point communication link for the exchange of complex data in both directions. This technology also enables the transfer of parameters to or from Devices and the delivery of identification and diagnostic information from the Devices to the automation system.

This technology is mainly intended for use with simple sensors and actuators in factory automation, which include small and cost-effective microcontrollers.

This part specifies the SDCI communication services and protocol (physical layer, data link layer and application layer in accordance with the ISO/OSI reference model) for both SDCI Masters and Devices.

This part also includes EMC test requirements.

This part does not cover communication interfaces or systems incorporating multiple point or multiple drop linkages, or integration of SDCI into higher level systems such as fieldbuses.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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, radiofrequency, 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*

IEC 61000-4-11, *Electromagnetic compatibility (EMC) – Part 4-11: Testing and measurement techniques – Voltage dips, short interruptions and voltage variations immunity tests*

---

[2] IO-Link™ is a trade name of the "IO-Link Community". This information is given for the convenience of users of this international Standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this standard does not require use of the registered logos for IO-Link™. Use of the registered logos for IO-Link™ requires permission of the "IO-Link Community".

IEC 61000-6-2, *Electromagnetic compatibility (EMC) – Part 6-2: Generic standards – Immunity for industrial environments*

IEC 61000-6-4, *Electromagnetic compatibility (EMC) – Part 6-4: Generic standards – Emission standard for industrial environments*

IEC 61076-2-101, *Connectors for electronic equipment – Product requirements – Part 2-101: Circular connectors – Detail specification for M12 connectors with screw-locking*

IEC 61131-1, *Programmable controllers – Part 1: General information*

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

IEC/TR 62390, *Common automation device – Profile guideline*

ISO/IEC 646:1991, *Information technology – ISO 7-bit coded character set for information interchange*

ISO/IEC 2022, *Information technology – Character code structure and extension techniques*

ISO/IEC 10646, *Information technology – Universal Multiple-Octet Coded Character Set (UCS)*

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

ISO/IEC 19505 (all parts), *Information technology – Object Management Group Unified Modeling Language (OMG UML)*

ISO 1177, *Information processing – Character structure for start/stop and synchronous character-oriented transmission*

ANSI/IEEE Std 754-1985, *IEEE Standard for Floating-Point Arithmetic*

Internet Engineering Task Force (IETF): RFC 1305 – *Network Time Protocol Version 4: Specification, Implementation and Analysis*; available at < www.ietf.org >

3 Terms, definitions, symbols, abbreviated terms and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 61131-1 and IEC 61131-2, as well as the following apply.

3.1.1
address
part of the M-sequence control to reference data within data categories of a communication channel

3.1.2
application layer
AL
<SDCI> part of the protocol responsible for the transmission of Process Data objects and On-request Data objects

3.1.3
Block Parameter
consistent parameter access via multiple Indices or Subindices

3.1.4
checksum
<SDCI> complementary part of the overall data integrity measures in the data link layer in addition to the UART parity bit

3.1.5
CHKPDU
integrity protection data within an ISDU communication channel generated through XOR processing the octets of a request or response

3.1.6
coded switching
SDCI communication, based on the standard binary signal levels of IEC 61131-2

3.1.7
COM1
SDCI communication mode with transmission rate of 4,8 kbit/s

3.1.8
COM2
SDCI communication mode with transmission rate of 38,4 kbit/s

3.1.9
COM3
SDCI communication mode with transmission rate of 230,4 kbit/s

3.1.10
COMx
one out of three possible SDCI communication modes COM1, COM2, or COM3

3.1.11
communication channel
logical connection between Master and Device

Note 1 to entry: Four communication channels are defined: process channel, page and ISDU channel (for parameters), and diagnosis channel.

3.1.12
communication error
unexpected disturbance of the SDCI transmission protocol

3.1.13
cycle time
time to transmit an M-sequence between a Master and its Device including the following idle time

3.1.14
Device
single passive peer to a Master such as a sensor or actuator

Note 1 to entry: Uppercase "Device" is used for SDCI equipment, while lowercase "device" is used in a generic manner.

3.1.15
Direct Parameters
directly (page) addressed parameters transferred acyclically via the page communication channel without acknowledgment

3.1.16
dynamic parameter
part of a Device's parameter set defined by on-board user interfaces such as teach-in buttons or control panels in addition to the static parameters

3.1.17
Event
instance of a change of conditions in a Device

Note 1 to entry: Uppercase "Event" is used for SDCI Events, while lowercase "event" is used in a generic manner.

Note 2 to entry: An Event is indicated via the Event flag within the Device's status cyclic information, then acyclic transfer of Event data (typically diagnosis information) is conveyed through the diagnosis communication channel.

3.1.18
fallback
transition of a port from coded switching to switching signal mode

3.1.19
inspection level
degree of verification for the Device identity

3.1.20
interleave
segmented cyclic data exchange for Process Data with more than 2 octets through subsequent cycles

3.1.21
input
information transport in direction from Device to Master

3.1.22
ISDU
indexed service data unit used for acyclic acknowledged transmission of parameters that can be segmented in a number of M-sequences

3.1.23
legacy (Device or Master)
Device or Master designed in accordance with [8][3]

3.1.24
M-sequence
sequence of two messages comprising a Master message and its subsequent Device message

3.1.25
M-sequence control
first octet in a Master message indicating the read/write operation, the type of the communication channel, and the address, for example offset or flow control

3.1.26
M-sequence error
unexpected or wrong message content, or no response

3.1.27
M-sequence type
one particular M-sequence format out of a set of specified M-sequence formats

3.1.28
Master
active peer connected through ports to one up to n Devices and which provides an interface to the gateway to the upper level communication systems or PLCs

Note 1 to entry: Uppercase "Master" is used for SDCI equipment, while lowercase "master" is used in a generic manner.

3.1.29
message
<SDCI> sequence of UART frames transferred either from a Master to its Device or vice versa following the rules of the SDCI protocol

3.1.30
On-request Data
OD
acyclically transmitted data upon request of the Master application consisting of parameters or Event data

---

3 Numbers in square brackets refer to the Bibliography.

3.1.31
output
information transport in direction from Master to Device

3.1.32
physical layer
first layer of the ISO-OSI reference model, which provides the mechanical, electrical, functional and procedural means to activate, maintain, and de-activate physical connections for bit transmission between data-link entities

Note 1 to entry: Physical layer also provides means for wake-up and fallback procedures.

[SOURCE: ISO/IEC 7498-1, 7.7.2, modified — text extracted from subclause, note added]

3.1.33
port
communication medium interface of the Master to one Device

3.1.34
Process Data
PD
input or output (seen from Master’s view) values from or to a discrete or continuous automation process cyclically transferred with high priority and in a configured schedule automatically between Master and Device

3.1.35
Process Data cycle
complete transfer of all Process Data from or to an individual Device that may comprise several cycles in case of segmentation (interleave)

3.1.36
single parameter
independent parameter access via one single Index or Subindex

3.1.37
SIO
port operation mode in accordance with digital input and output defined in IEC 61131-2 (seen from Master’s view) that is established after power-up or fallback or unsuccessful communication attempts

3.1.38
static parameter
part of a Device's parameter set to be saved in a Master for the case of replacement without engineering tools

3.1.39
switching signal
binary signal from or to a Device when in SIO mode (as opposed to the "coded switching" SDCI communication)

3.1.40
System Management
SM
<SDCI> means to control and coordinate the internal communication layers and the exceptions within the Master and its ports, and within each Device

3.1.41
UART frame
<SDCI> bit sequence starting with a start bit, followed by eight bits carrying a data octet, followed by an even parity bit and ending with one stop bit

3.1.42
wake-up
procedure for causing a Device to change its mode from SIO to SDCI

3.1.43
wake-up request

WURQ

physical layer service used by the Master to initiate wake-up of a Device, and put it in a receive ready state

3.2 Symbols and abbreviated terms

Δf_{DTRM}	permissible deviation from data transfer rate (measured in %)
ΔVS	power supply ripple (measured in V)
AL	application layer
BEP	bit error probability
C/Q	connection for communication (C) or switching (Q) signal (SIO)
CL_{eff}	effective total cable capacity (measured in nF)
CQ	input capacity at C/Q connection (measured in nF)
DI	digital input (Master's view)
DL	data link layer
DO	digital output (Master's view)
f_{DTR}	data transfer rate (measured in bit/s)
H/L	high/low signal at receiver output
I/O	input/output
ILL	input load current at input C/Q to $V0$ (measured in A)
IODD	IO Device Description (see 10.9)
$IP24_{\mathrm{M}}$	extra DC supply current for Devices
IQ	driver current in saturated operating status ON (measured in A)
IQH	driver current on high-side driver in saturated operating status ON (measured in A)
IQL	driver current on low-side driver in saturated operating status ON (measured in A)
$IQPK$	maximum driver current in unsaturated operating status ON (measured in A)
$IQPKH$	maximum driver current on high-side driver in unsaturated operating status ON (measured in A)
$IQPKL$	maximum driver current on low-side driver in unsaturated operating status ON (measured in A)
IQQ	quiescent current at input C/Q to $V0$ with inactive output drivers (measured in A)
IQ_{WU}	amplitude of Master's wake-up request current (measured in A)
IS	supply current at $V+$ (measured in A)
$ISIR$	current pulse supply capability at $V+$ (measured in A)
LED	light emitting diode
L-	power supply (-)
L+	power supply (+)
N24	24 V extra power supply (-)
n_{WU}	wake-up retry count
On/Off	driver's ON/OFF switching signal
OD	On-request Data
OVD	signal overload detect
P24	24 V extra power supply (+)
PD	Process Data
PDCT	port and Device configuration tool
PL	physical layer
PLC	programmable logic controller

PS	power supply (measured in V)	
QIS_{D}	power-up charge consumption	
r	time to reach a stable level with reference to the beginning of the start bit (measured in T_{BIT})	
RL_{eff}	loop resistance of cable (measured in Ω)	
s	time to exit a stable level with reference to the beginning of the start bit (measured in T_{BIT})	
SDCI	single-drop digital communication interface	
SIO	standard input output (digital switching mode, Master's view)	[IEC 61131-2]
SM	system management	
SMI	standardized Master interface	
t_1	UART frame transfer delay on Master (measured in T_{BIT})	
t_2	UART frame transfer delay on Device (measured in T_{BIT})	
t_{A}	response delay on Device (measured in T_{BIT})	
T_{BIT}	bit time (measured in s)	
t_{CYC}	cycle time on M-sequence level (measured in s)	
t_{DF}	fall time (measured in s)	
T_{DMT}	delay time while establishing Master port communication (measured in T_{BIT})	
T_{DR}	rise time (measured in s)	
T_{DSIO}	delay time on Device for transition to SIO mode following wake-up request (measured in s)	
T_{DWU}	wake-up retry delay (measured in s)	
$t_{\mathrm{M\text{-}sequence}}$	M-sequence duration (measured in T_{BIT})	
t_{idle}	idle time between two M-sequences (measured in s)	
t_{H}	detection time for high level (measured in s)	
t_{L}	detection time for low level (measured in s)	
t_{ND}	noise suppression time (measured in s)	
T_{RDL}	wake-up readiness following power ON (measured in s)	
T_{REN}	receive enable (measured in s)	
T_{SD}	device detect time (measured in s)	
T_{WU}	pulse duration of wake-up request (measured in s)	
UART	universal asynchronous receiver transmitter	
UML	Unified Modelling Language	[ISO/IEC 19505]
$V+$	voltage at L+	
$V0$	voltage at L-	
$VD+_{\mathrm{L}}$	voltage drop on the line between the L+ connections on Master and Device (measured in V)	
$VD0_{\mathrm{L}}$	voltage drop on the line between the L- connections on Master and Device (measured in V)	
VDQ_{L}	voltage drop on the line between the C/Q connections on Master and Device (measured in V)	
$VHYS$	hysteresis of receiver threshold voltage (measured in V)	
VI	input voltage at connection C/Q with reference to $V0$ (measured in V)	
VIH	input voltage range at connection C/Q for high signal (measured in V)	
VIL	input voltage range at connection C/Q for low signal (measured in V)	
$VP24_{\mathrm{M}}$	extra DC supply voltage for Devices	
VRQ	residual voltage on driver in saturated operating status ON (measured in V)	
$VRQH$	residual voltage on high-side driver in operating status ON (measured in V)	
$VRQL$	residual voltage on low-side driver in saturated operating status ON (measured in V)	

VTH	threshold voltage of receiver with reference to *V0* (measured in V)
VTHH	threshold voltage of receiver for safe detection of a high signal (measured in V)
VTHL	threshold voltage of receiver for safe detection of a low signal (measured in V)
WURQ	wake-up request pulse

3.3 Conventions

3.3.1 General

The service model, service primitives, and the diagrams shown in this standard are entirely abstract descriptions. The implementation of the services may reflect individual issues and can be different.

3.3.2 Service parameters

Service primitives are used to represent service provider/consumer interactions (ISO/IEC 10731). They convey parameters which indicate the information available in the provider/consumer interaction. In any particular interface, not each and every parameter needs to be explicitly stated.

The service specification in this standard 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:

1) parameter name;
2) request primitive (.req);
3) indication primitive (.ind);
4) response primitive (.rsp); and
5) confirmation primitive (.cnf).

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 can or cannot 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.

– 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 related to the parameter and its use.

3.3.3 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 a communication layer service provider and a communication layer service consumer 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.

3.3.4 Service attributes

The nature of the different (Master and Device) services is characterized by attributes. All services are defined from the view of the affected layer towards the layer above.

I Initiator of a service (towards the layer above)

R Receiver (responder) of a service (from the layer above)

3.3.5 Figures

For figures that show the structure and services of protocol layers, the following conventions are used:

- an arrow with just a service name represents both a request and the corresponding confirmation, with the request being in the direction of the arrow;
- a request without confirmation, as well as all indications and responses are labelled as such (i.e. service.req, service.ind, service.rsp).

Figure 1 shows the example of a confirmed service.

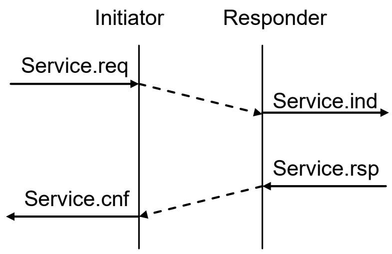


Figure 1 – Example of a confirmed service

3.3.6 Transmission octet order

Figure 2 shows how WORD based data types are transferred from memory to transmission medium and vice versa (i.e. most significant octet transmitted first, see 7.3.3.2 and 7.3.6.1).

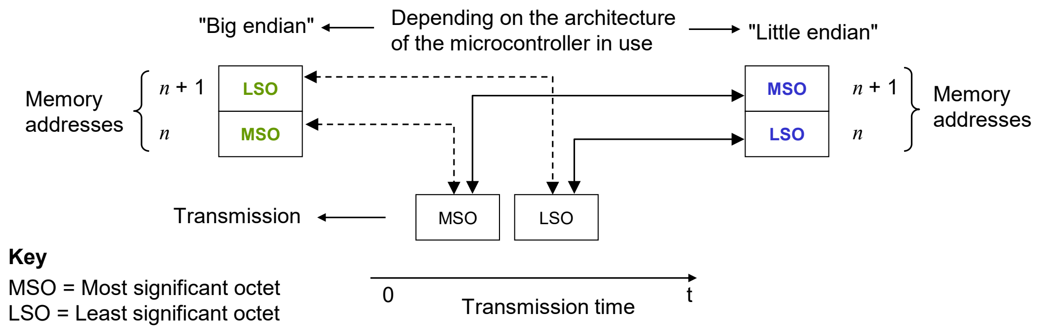


Figure 2 – Memory storage and transmission order for WORD based data types

3.3.7 Behavioral descriptions

For the behavioral descriptions, the notations of UML 2 (ISO/IEC 19505) are used (e.g. state, sequence, activity, timing diagrams, guard conditions).

State diagrams are the primary source for implementations whereas sequence charts illustrate certain use cases.

Characteristics of state diagrams are

- triggers/events coming from external requests ("calls") or internal changes such as timeouts;
- [guard(s)] as Boolean expressions for exits of states;
- numbered transitions describing actions in addition to the triggers within separate state-transition tables.

The layout of these tables is following IEC/TR 62390.

In this document, the concept of "nested states" with superstates and substates is used as shown in the example of Figure 3.

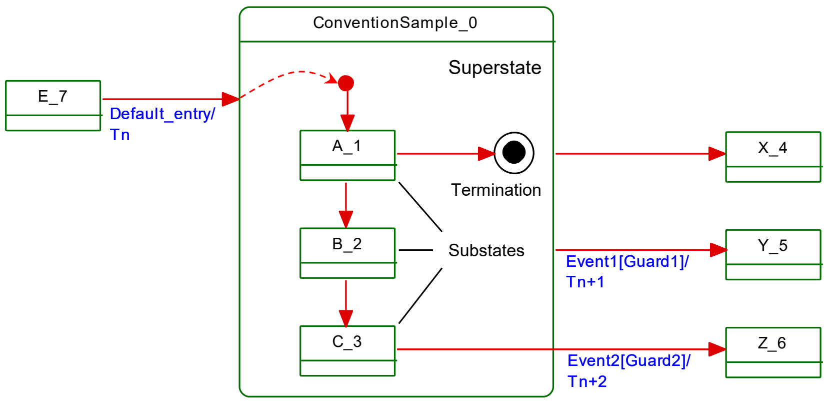


Figure 3 – Example of a nested state

UML 2 allows hierarchies of states with superstates and substates. The highest superstate represents the entire state machine.

This concept allows for simplified modelling since the content of superstates can be moved to a separate drawing. An eyeglasses icon usually represents this content.

Compared to "flat" state machines, a particular set of rules shall be observed for "nested states":

a) A transition to the edge of a superstate (e.g. Default_entry) implies transition to the initial substate (e.g. A_1).

b) Transition to a termination state inside a superstate implies a transition without event and guard to a state outside (e.g.X_4). The superstate will become inactive.

c) A transition from any of the substates (e.g. A_1, B_2, or C_3) to a state outside (Y_5) can take place whenever Event1 occurs and Guard1 is true. This is helpful in case of common errors within the substates. The superstate will become inactive.

d) A transition from a particular substate (e.g. C_3) to a state outside (Z_6) can take place whenever Event2 occurs and Guard2 is true. The superstate will become inactive.

Due to UML design tool restrictions the following exceptions apply.

For state diagrams, a service parameter (in capital letters) is attached to the service name via an underscore character, such as for example in DL_SetMode_INACTIVE.

For sequence diagrams, the service primitive is attached via an underscore character instead of a dot, and the service parameter is added in parenthesis, such as for example in DL_Event_ind (OPERATE).

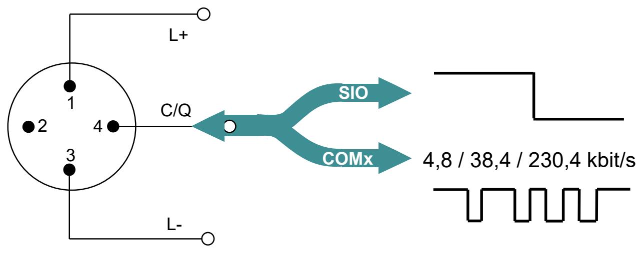
Timing constraints are labelled "tm(time in ms)".

Asynchronously received service calls are not modelled in detail within state diagrams.

4 Overview of SDCI (IO-Link™[4])

4.1 Purpose of technology

Figure 4 shows the basic concept of SDCI.



Pin	Signal	Definition	Standard
1	L+	24 V	IEC 61131-2
2	I/Q	Not connected, DI, or DO	IEC 61131-2
3	L-	0 V	IEC 61131-2
4	Q	"Switching signal" (SIO)	IEC 61131-2
	C	"Coded switching" (COM1, COM2, COM3)	IEC 61131-9

Figure 4 – SDCI compatibility with IEC 61131-2

The single-drop digital communication interface technology for small sensors and actuators SDCI (commonly known as IO-Link™) defines a migration path from the existing digital input and digital output interfaces for switching 24 V Devices as defined in IEC 61131-2 towards a point-to-point communication link. Thus, for example, digital I/O modules in existing fieldbus peripherals can be replaced by SDCI Master modules providing both classic DI/DO interfaces and SDCI. Analog transmission technology can be replaced by SDCI combining its robustness, parameterization, and diagnostic features with the saving of digital/analog and analog/digital conversion efforts.

[4] IO-Link™ is a trade name of the "IO-Link Community". This information is given for the convenience of users of this international Standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this standard does not require use of the registered logos for IO-Link™. Use of the registered logos for IO-Link™ requires permission of the "IO-Link Community".

4.2 Positioning within the automation hierarchy

Figure 5 shows the domain of the SDCI technology within the automation hierarchy.

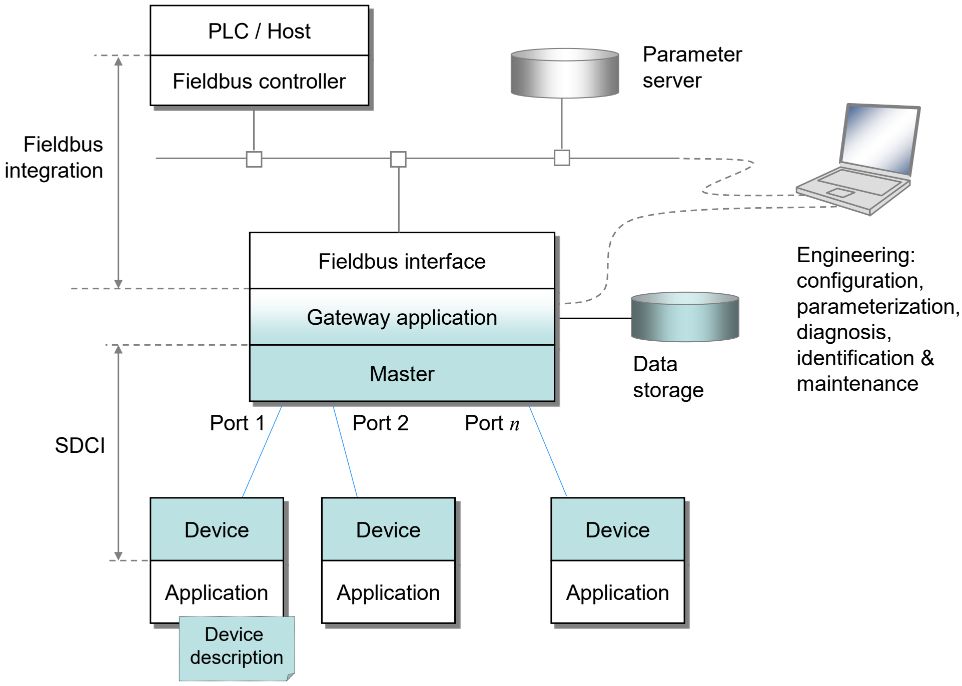


Figure 5 – Domain of the SDCI technology within the automation hierarchy

The SDCI technology defines a generic interface for connecting sensors and actuators to a Master unit, which may be combined with gateway capabilities to become a fieldbus remote I/O node.

Starting point for the design of SDCI is the classic 24 V digital input (DI) defined in IEC 61131-2 and output interface (DO) specified in Table 6. Thus, SDCI offers connectivity of classic 24 V sensors ("switching signals") as a default operational mode. Additional connectivity is provided for actuators when a port has been configured into "single-drop communication mode".

Many sensors and actuators nowadays are already equipped with microcontrollers offering a UART interface that can be extended by addition of a few hardware components and protocol software to support SDCI communication. This second operational mode uses "coded switching" of the 24 V I/O signalling line. Once activated, the SDCI mode supports parameterization, cyclic data exchange, diagnosis reporting, identification & maintenance information, and external parameter storage for Device backup and fast reload of replacement devices. Sensors and actuators with SDCI capability are referred to as "Devices" in this standard. To improve start-up performance these Devices usually provide non-volatile storage for parameters.

NOTE Configuration and parameterization of Devices is supported through an XML-based device description (see [6]), which is not part of this standard.

4.3 Wiring, connectors and power

The default connection (port class A) comprises 4 pins (see Figure 4). The default wiring for port class A complies with IEC 60947-5-2 and uses only three wires for 24 V, 0 V, and a signal line. The fourth wire may be used as an additional signal line complying with IEC 61131-2.

Five pins connections (port class B) are specified for Devices requiring additional power from an independant 24 V power supply.

NOTE A port class A Device using the fourth wire is not compatible with a port class B Master.

Maximum length of cables is 20 m, shielding is not required.

4.4 Communication features of SDCI

The generic Device model is shown in Figure 6 and explained in the following paragraphs.

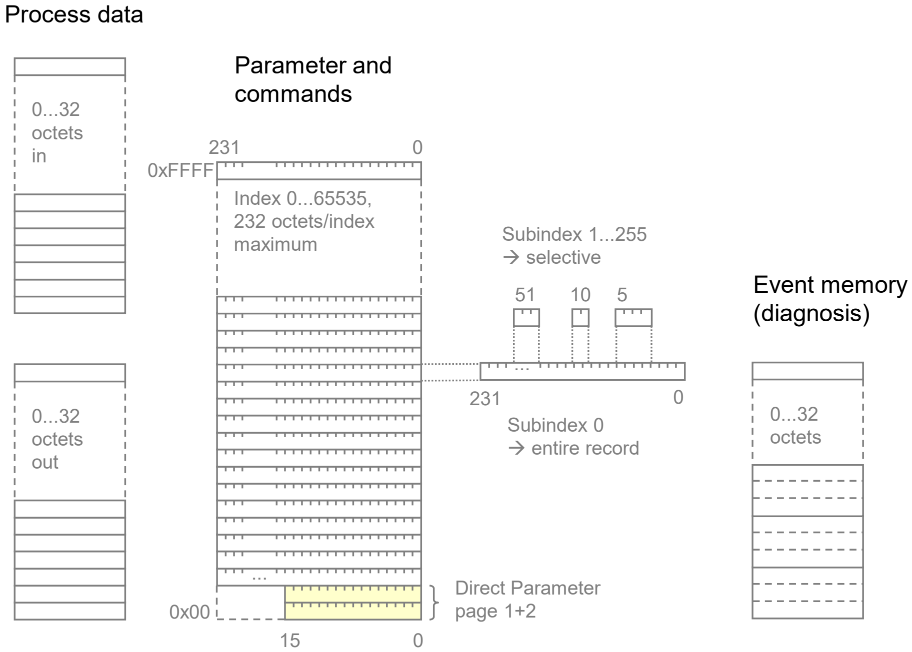


Figure 6 – Generic Device model for SDCI (Master's view)

A Device may receive Process Data (out) to control a discrete or continuous automation process or send Process Data (in) representing its current state or measurement values. The Device usually provides parameters enabling the user to configure its functions to satisfy particular needs. To support this case a large parameter space is defined with access via an Index (0 to 65535; with a predefined organization) and a Subindex (0 to 255).

The first two index entries 0 and 1 are reserved for the Direct Parameter page 1 and 2 with a maximum of 16 octets each. Parameter page 1 is mainly dedicated to Master commands such as Device startup and fallback, retrieval of Device specific operational and identification information. Parameter page 2 allows for a maximum of 16 octets of Device specific parameters.

The other indices (2 to 65535) each allow access to one record having a maximum size of 232 octets. Subindex 0 specifies transmission of the complete record addressed by the Index, other subindices specify transfer of selected data items within the record.

Within a record, individual data items may start on any bit offset, and their length may range from 1 bit to 232 octets, but the total number of data items in the record cannot exceed 255. The organization of data items within a record is specified in the IO Device Description (IODD).

All changes of Device condition that require reporting or intervention are stored within an Event memory before transmission. An Event flag is then set in the cyclic data exchange to indicate the existence of an Event.

Communication between a Master and a Device is point-to-point and is based on the principle of a Master first sending a request message and then a Device sending a response message (see Figure 38). Both messages together are called an M-sequence. Several M-sequence types are defined to support user requirements for data transmission (see Figure 39).

Data of various categories are transmitted through separate communication channels within the data link layer, as shown in Figure 7.

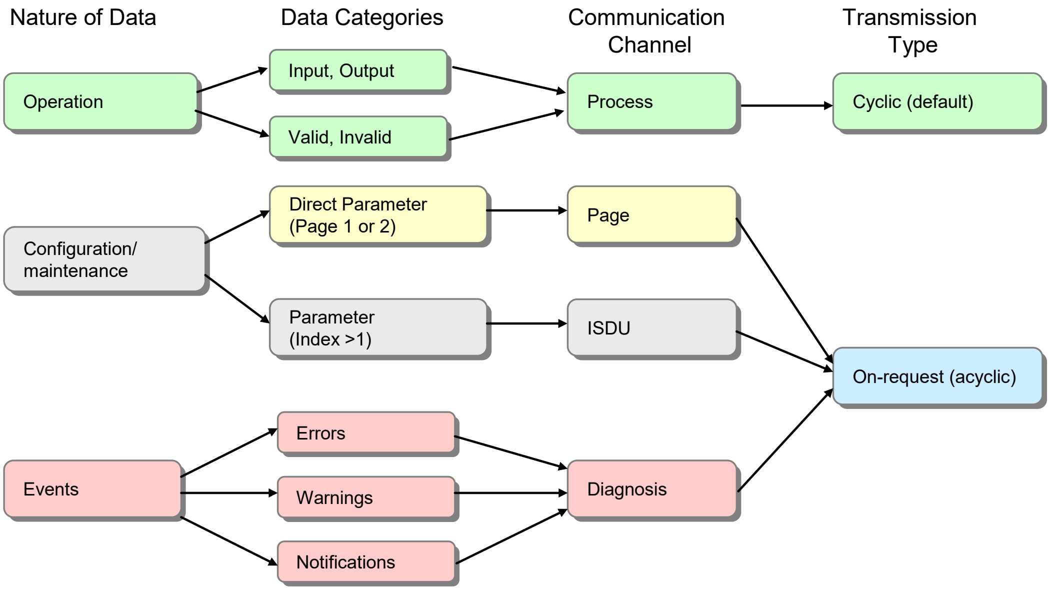


Figure 7 – Relationship between nature of data and transmission types

- Operational data such as Device inputs and outputs is transmitted through a process channel using cyclic transfer. Operational data may also be associated with qualifiers such as valid/invalid.
- Configuration and maintenance parameters are transmitted using acyclic transfers. A page channel is provided for direct access to parameter pages 1 and 2, and an ISDU channel is used for accessing additional parameters and commands.
- Device events are transmitted using acyclic transfers through a diagnostic channel. Device events are reported using 3 severity levels, error, warning, and notification.

The first octet of a Master message controls the data transfer direction (read/write) and the type of communication channel.

Figure 8 shows each port of a Master has its own data link layer which interfaces to a common master application layer. Within the application layer, the services of the data link layer are translated into actions on Process Data objects (input/output), On-request Data objects (read/write), and events. Master applications include a Configuration Manager (CM), Data Storage mechanism (DS), Diagnosis Unit (DU), On-request Data Exchange (ODE), and a Process Data Exchange (PDE).

System Management checks identification of the connected Devices and adjusts ports and Devices to match the chosen configuration and the properties of the connected Devices. It controls the state machines in the application (AL) and data link layers (DL), for example at start-up.

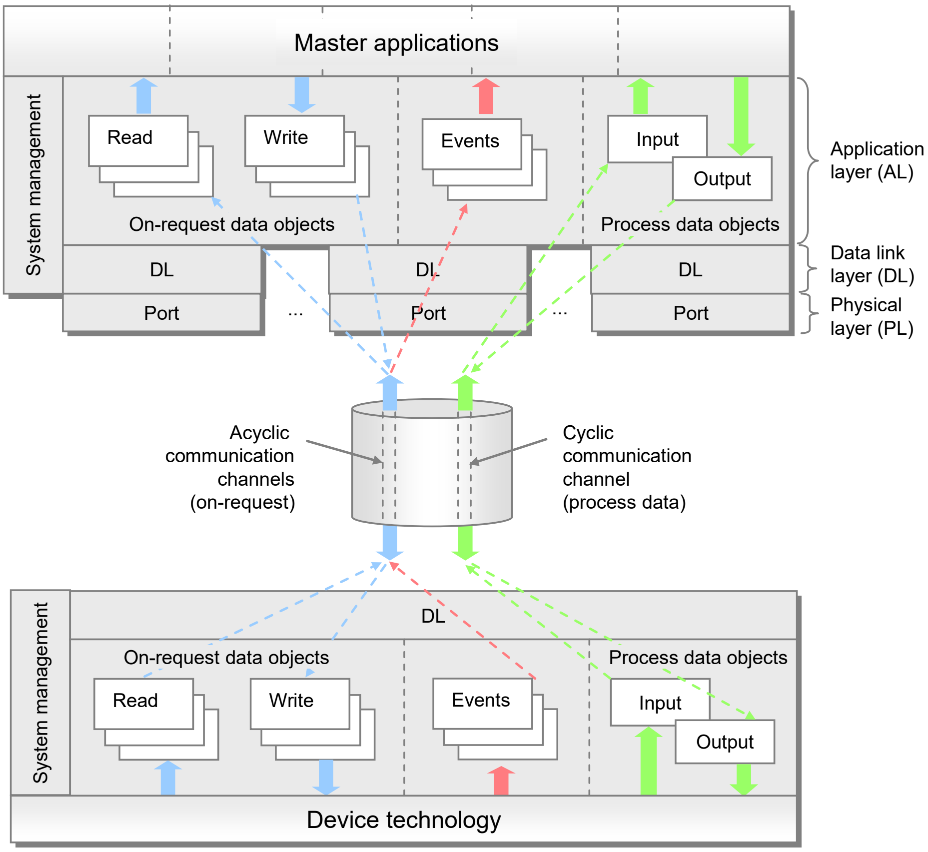


Figure 8 – Object transfer at the application layer level (AL)

4.5 Role of a Master

A Master accommodates 1 to n ports and their associated data link layers. During start-up it changes the ports to the user-selected port modes, which can be DEACTIVATED, IOL_MANUAL, IOL_AUTOSTART, DI_C/Q, or DO_C/Q. If communication is requested, the Master uses a special wake-up current pulse to initiate communication with the Device. The Master then auto-adjusts the transmission rate to COM1, COM2, or COM3 (see Table 9) and checks the "personality" of the connected Device, i.e. its VendorID, DeviceID, and communication properties.

If there is a mismatch between the Device parameters and the stored parameter set within the Master, the parameters in the Device are overwritten (see 11.4) or the stored parameters within the master are updated depending on the configuration.

The Master is responsible for the assembling and disassembling of all data from or to the Devices (see Clause 11).

The Master provides a Data Storage area of at least 2 048 octets per Device for backup of Device data (see 11.4). The Master may combine this Device data together with all other relevant data for its own operation, and make this data available for higher level applications for Master backup purpose or recipe control (see 13.4.2).

4.6 SDCI configuration

Engineering support for a Master is usually provided by a Port and Device Configuration Tool (PDCT). The PDCT configures both port properties and Device properties (see parameters shown in Figure 6). It combines both an interpreter of the I/O Device Description (IODD) and a configurator (see 13). The IODD provides all the necessary properties to establish communication and the necessary parameters and their boundaries to establish the desired function of a sensor or actuator. The PDCT also supports the compilation of the Process Data for propagation on the fieldbus and vice versa.

4.7 Mapping to fieldbuses and/or other upper level systems

Specifications for integration of Masters into upper level systems such as a fieldbus system, i.e. the definition of gateway functions for exchanging data with upper level entities, is out of scope of this standard. However, all functions of this standard are mandatory to be made available to the users by a particular integration according to the capability level of the upper level system technology except for those functions that are declared explicitly as optional.

EXAMPLE These functions include mapping of the Process Data exchange, realization of program-controlled parameterization or a remote parameter server, or the propagation of diagnosis information.

The integration of a PDCT into engineering tools of a particular fieldbus or other upper level system is out of scope of this standard.

4.8 Standard structure

Figure 9 shows the logical structure of the Master and Device. Clause 5 specifies the Physical Layer (PL) of SDCI, Clause 6 specifies details of the SIO mode. Clause 7 specifies Data Link Layer (DL) services, protocol, wake-up, M-sequences, and the DL layer handlers. Clause 0 specifies the services and the protocol of the Application Layer (AL) and clause 9 the System Management responsibilities (SM).

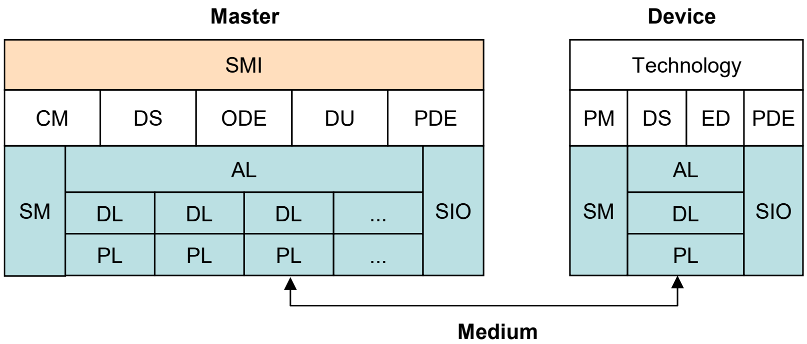


Figure 9 – Logical structure of Master and Device

Clause 10 specifies Device applications and features. These include Process Data Exchange (PDE), Parameter Management (PM), Data Storage (DS), and Event Dispatcher (ED). Technology specific Device applications are not part of this standard. They may be specified in profiles for particular Device families.

Clause 11 specifies Master applications and features. These include Process Data Exchange (PDE), On-request Data Exchange (ODE), Configuration Management (CM), Data Storage (DS) and Diagnosis Unit (DU). A Standardized Master Interface (SMI) ensures uniform behavior via specified services and allows for usage of one PDCT (Master tool) for different Master brands.

Clause 12 provides a holistic best practice view on Data Storage behavior of both Master and Device.

Clause 13 outlines integration aspects of IO-Link into various automation and IT realms.

Several normative and informative annexes are included. Annex A defines the available M-sequence types. Annex B describes the parameters of the Direct Parameter page and the fixed Device parameters. Annex C lists the error types in case of acyclic transmissions and Annex D the EventCodes (diagnosis information of Devices). Annex E specifies the coding of argument blocks for the SMI services. Annex F specifies the available basic and composite data types. Annex G defines the structure of Data Storage objects. Annex H deals with conformity and electromagnetic compatibility test requirements and Annex I provides graphs of residual error probabilities, demonstrating the level of SDCI's data integrity. The informative Annex J provides an example of the sequence of acyclic data transmissions. The informative Annex K explains two recommended methods for detecting parameter changes in the context of Data Storage.

5 Physical Layer (PL)

5.1 General

5.1.1 Basics

The 3-wire connection system of SDCI is based on the specifications in IEC 60947-5-2. The three lines are used as follows: (L+) for the 24 V power supply, (L-) for the ground line, and (C/Q) for the switching signal (Q) or SDCI communication (C), as shown in Figure 10.

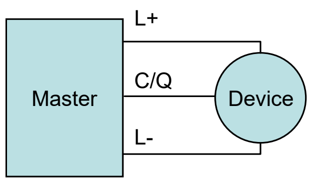


Figure 10 – Three wire connection system

NOTE Binary sensors compliant with IEC 60947-5-2 are compatible with the SDCI 3-wire connection system (including from a power consumption point of view).

Support of the SDCI 3-wire connection system is mandatory for Master. Ports with this characteristic are called port class A.

Port class A uses a four-pin connector. The fourth wire may be used as an additional signal line complying with IEC 61131-2. Its support is optional in both Masters and Devices.

Five wire connections (port class B) are specified for Devices requiring additional power from an independant 24 V power supply (see 5.5.1).

NOTE A port class A Device using the fourth wire is not compatible with a port class B Master.

5.1.2 Topology

The SDCI system topology uses point-to-point links between a Master and its Devices as shown in Figure 11. The Master may have multiple ports for the connection of Devices. Only one Device shall be connected to each port.

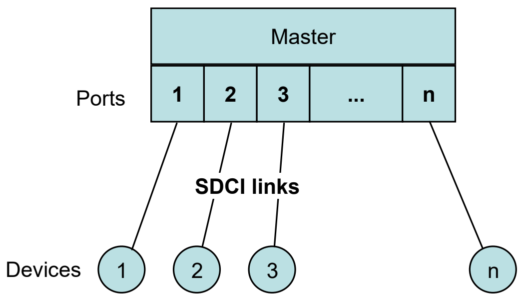


Figure 11 – Topology of SDCI

5.2 Physical layer services

5.2.1 Overview

Figure 12 shows an overview of the Master's physical layer and its service primitives.

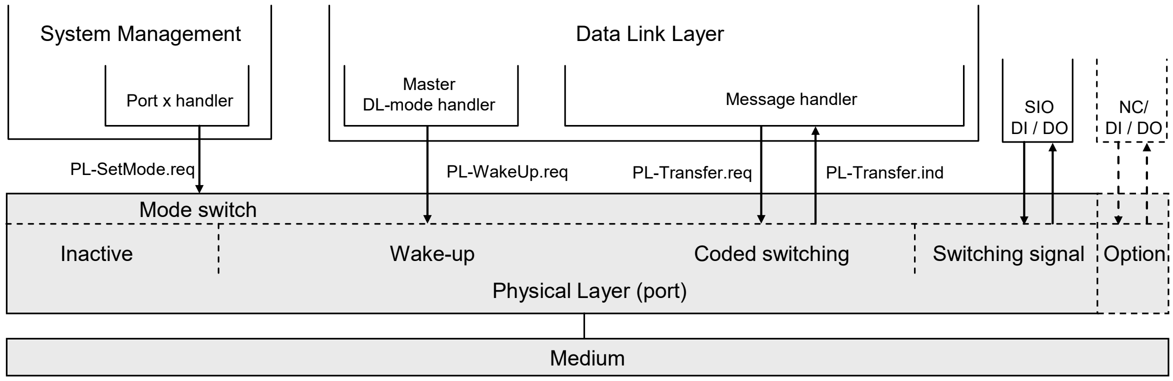


Figure 12 – Physical layer (Master)

The physical layer specifies the operation of the C/Q line in Figure 4 and the associated line driver (transmitter) and receiver of a particular port. The Master operates this line in three main modes (see Figure 12): inactive, "Switching signal" (DI/DO), or "Coded switching" (COMx). The service PL-SetMode.req is responsible for switching into one of these modes.

If the port is in inactive mode, the C/Q line shall be high impedance (floating). In SIO mode, the port can be used as a standard input or output interface according to the definitions of IEC 61131-2 or in Table 6 respectively. The communication layers of SDCI are bypassed as shown in Figure 12; the signals are directly processed within the Master application. In SDCI mode, the service PL_WakeUp.req creates a special signal pattern (current pulse) that can be detected by an SDCI enabled Device connected to this port (see 5.3.3.3).

Figure 13 shows an overview of the Device's physical layer and its service primitives.

The physical layer of a Device according to Figure 13 follows the same principle, except that there is no inactive state. By default, at power on or cable reconnection, the Device shall operate in the SIO mode, as a digital input (from a Master's point of view). The Device shall always be able to detect a wake up except during a permanent inactive state. The service PL_WakeUp.ind reports successful detection of the wake-up request (usually a microcontroller interrupt), which is required for the Device to switch to the SDCI mode.

A special MasterCommand (fallback) sent via SDCI causes the Device to switch back to SIO mode.

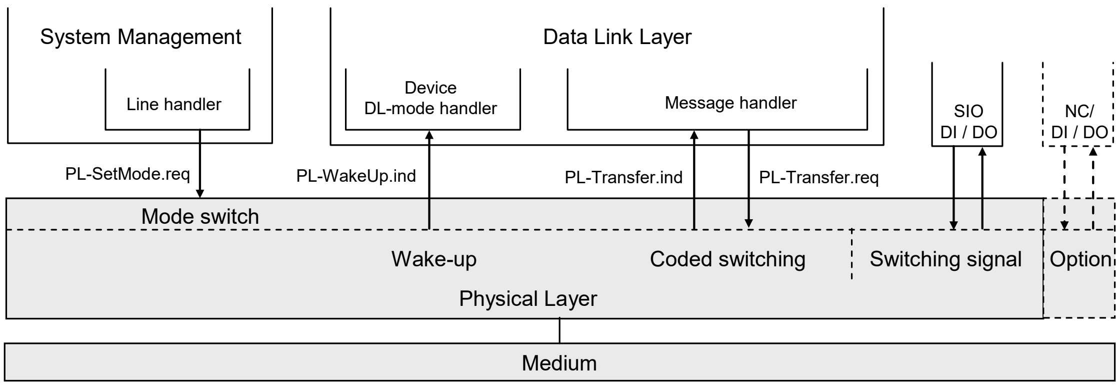


Figure 13 – Physical layer (Device)

Subsequently, the services are specified that are provided by the PL to System Management and to the Data Link Layer (see Figure 85 and Figure 96 for a complete overview of all the services). Table 1 lists the assignments of Master and Device to their roles as initiator or receiver for the individual PL services.

Table 1 – Service assignments of Master and Device

Service name	Master	Device
PL-SetMode	R	R
PL-WakeUp	R	I
PL-Transfer	I / R	R / I
Key (see 3.3.4) I Initiator of service R Receiver (Responder) of service		

5.2.2 PL services

5.2.2.1 PL_SetMode

The PL-SetMode service is used to setup the electrical characteristics and configurations of the Physical Layer. The parameters of the service primitives are listed in Table 2.

Table 2 – PL_SetMode

Parameter name	.req
Argument	M
TargetMode	M

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

TargetMode
This parameter indicates the requested operation mode

Permitted values:
INACTIVE (C/Q line in high impedance),
DI (C/Q line in digital input mode),
DO (C/Q line in digital output mode),
COM1 (C/Q line in COM1 mode),
COM2 (C/Q line in COM2 mode),
COM3 (C/Q line in COM3 mode)

5.2.2.2 PL_WakeUp

The PL-WakeUp service initiates or indicates a specific sequence which prepares the Physical Layer to send and receive communication requests (see 5.3.3.3). This unconfirmed service has no parameters. Its success can only be verified by a Master by attempting to communicate with the Device. The service primitives are listed in Table 3.

Table 3 – PL_WakeUp

Parameter name	.req	.ind
<none>		

5.2.2.3 PL_Transfer

The PL-Transfer service is used to exchange the SDCI data between Data Link Layer and Physical Layer. The parameters of the service primitives are listed in Table 4.

Table 4 – PL_Transfer

Parameter name	.req	ind.
Argument Data	 M	 M
Result (+)		S
Result (-) Status		S M

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

Data
This parameter contains the data value which is transferred over the SDCI interface.

Permitted values: 0…255

Result (+):
This selection parameter indicates that the service request has been executed successfully.

Result (-):
This selection parameter indicates that the service request failed.

Status
This parameter contains supplementary information on the transfer status.

Permitted values:
PARITY_ERROR (UART detected a parity error),
FRAMING_ERROR (invalid UART stop bit detected),
OVERRUN (octet collision within the UART)

5.3 Transmitter/Receiver

5.3.1 Description method

The physical layer is specified by means of electrical and timing requirements. Electrical requirements specify signal levels and currents separately for Master and Device in the form of reference schematics. Timing requirements specify the signal transmission process (specifically the receiver) and a special signal detection function.

5.3.2 Electrical requirements

5.3.2.1 General

The line driver is specified by a reference schematic corresponding to Figure 14. On the Master side, a transmitter comprises a combination of two line drivers and one current sink. On the Device side, in its simplest form, the transmitter takes the form of a p-switching driver. As an option there can be an additional n-switching or non-switching driver (this also allows the option of push-pull output operation).

In operating status ON the descriptive variables are the residual voltage VRQ, the standard driver current IQ, and the peak current $IQPK$. The source is controlled by the On/Off signal. An overload current event is indicated at the “Overload” output (OVD). This feature can be used for the current pulse detection (wake-up).

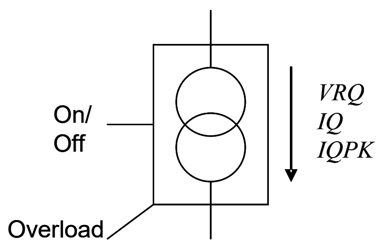


Figure 14 – Line driver reference schematics

The receiver is specified by a reference schematic according to Figure 15. It performs the function of a comparator and is specified 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.

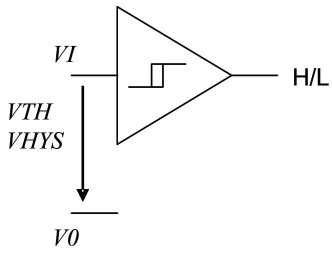
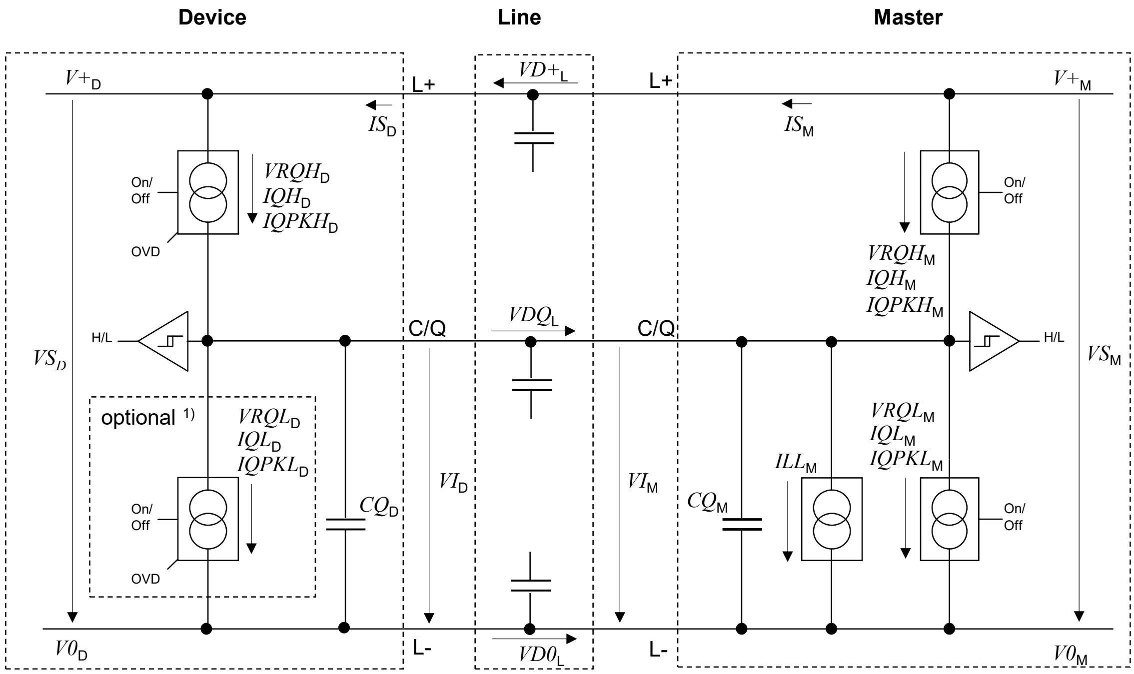


Figure 15 – Receiver reference schematics

Figure 16 shows the reference schematics for the interconnection of Master and Device for the SDCI 3-wire connection system.



1) Optional: low-side driver (push-pull only)

Figure 16 – Reference schematics for SDCI 3-wire connection system

The subsequent illustrations and parameter tables refer to the voltage level definitions in Figure 17. The parameter indices refer to the Master (M), Device (D) or line (L). The voltage drops on the line $VD+_L$, VDQ_L and $VD0_L$ are implicitely specified in 5.5 through cable parameters.

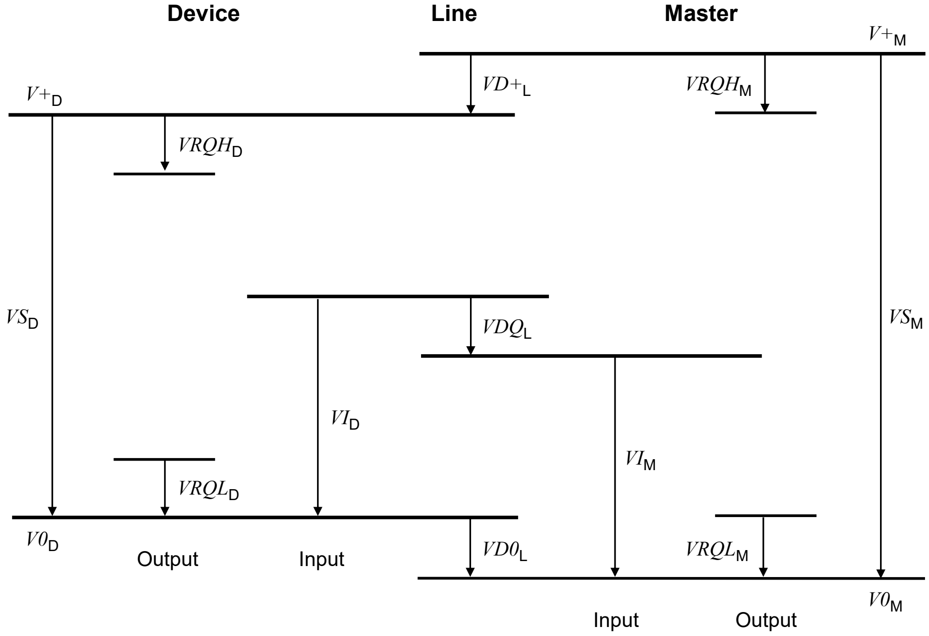


Figure 17 – Voltage level definitions

5.3.2.2 Receiver

The voltage range and switching threshold definitions are the same for Master and Device. The definitions in Table 5 apply (see also 5.4.1).

Table 5 – Electrical characteristics of a receiver

Property	Designation	Minimum	Typical	Maximum	Unit	Remark
$VTHH_{D,M}$	Input threshold 'H'	10,5	n/a	13	V	See NOTE 1
$VTHL_{D,M}$	Input threshold 'L'	8	n/a	11,5	V	See NOTE 1
$VHYS_{D,M}$	Hysteresis between input thresholds 'H' and 'L'	0	n/a	n/a	V	Shall not be negative See NOTE 2
VIL_D	Permissible voltage range 'L'	$V0_D$ -1,0	n/a	n/a	V	With reference to relevant negative supply voltage See NOTE 3
VIL_M	Permissible voltage range 'L'	$V0_M$	n/a	n/a	V	
VIH_D	Permissible voltage range 'H'	n/a	n/a	$V+_D$ + 1,0	V	With reference to relevant positive supply voltage. See NOTE 3
VIH_M	Permissible voltage range 'H'	n/a	n/a	$V+_M$	V	

NOTE 1 Thresholds are compatible with the definitions of type 1 digital inputs in IEC 61131-2.

NOTE 2 Hysteresis voltage $VHYS = VTHH – VTHL$

NOTE 3 Due to 5.4.1 the Master receiver signals VI_M are always within permitted supply ranges.

Figure 18 demonstrates the switching thresholds for the detection of Low and High signals.

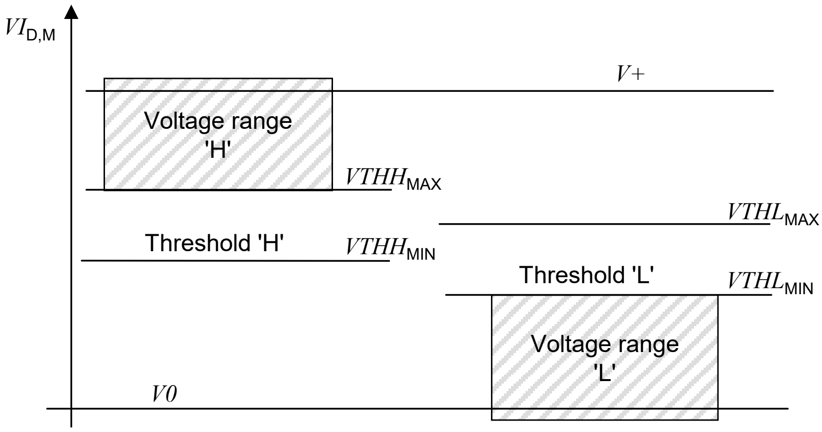


Figure 18 – Switching thresholds

5.3.2.3 Master port

The definitions in Table 6 are valid for the electrical characteristics of a Master port.

Table 6 – Electrical characteristics of a Master port

Property	Designation	Minimum	Typical	Maximum	Unit	Remark
VS_M	Supply voltage for Devices	20	24	30	V	See Figure 17
IS_M	Supply current for Devices	200	n/a	n/a	mA	See 5.4.1
$ISIR_M$	Current pulse capability for Devices	400	n/a	n/a	mA	See Figure 19
ILL_M	Load or discharge current for 0 V < VI_M < 5 V 5 V < VI_M < 15 V 15 V< VI_M < 30 V	 0 5/2 5	 n/a n/a n/a	 15 15 15	 mA mA mA	See NOTE 1
$VRQH_M$	Residual voltage 'H'	n/a	n/a	3	V	Voltage drop relating to $V+_M$ at maximum driver current IQH_M
$VRQL_M$	Residual voltage 'L'	n/a	n/a	3	V	Voltage drop relating to $V0_M$ at maximum driver current IQL_M
IQH_M	DC driver current 'H'	100	n/a	n/a	mA	
$IQPKH_M$	Output peak current 'H'	500	n/a	n/a	mA	Absolute value See NOTE 2
IQL_M	DC driver current 'L'	100	n/a	n/a	mA	
$IQPKL_M$	Output peak current 'L'	500	n/a	n/a	mA	Absolute value See NOTE 2
CQ_M	Input capacitance	n/a	n/a	1,0	nF	f=0 MHz to 4 MHz

NOTE 1 A minimum current of 2 mA for DI mode is compatible with the definition of type 1 digital inputs in IEC 61131-2. In communication mode, for the range 5 V < VI_M < 15 V, the minimum current is 5 mA instead of 2 mA in order to achieve short enough slew rates for pure p-switching Devices.

NOTE 2 Wake-up request current (5.3.3.3).

The Master shall provide a charge of 400 mA × 50 ms = 20 mAs within the first 50 ms after power-on without any overload-shutdown. After 50 ms, the specific current limitation of the Master or system applies.

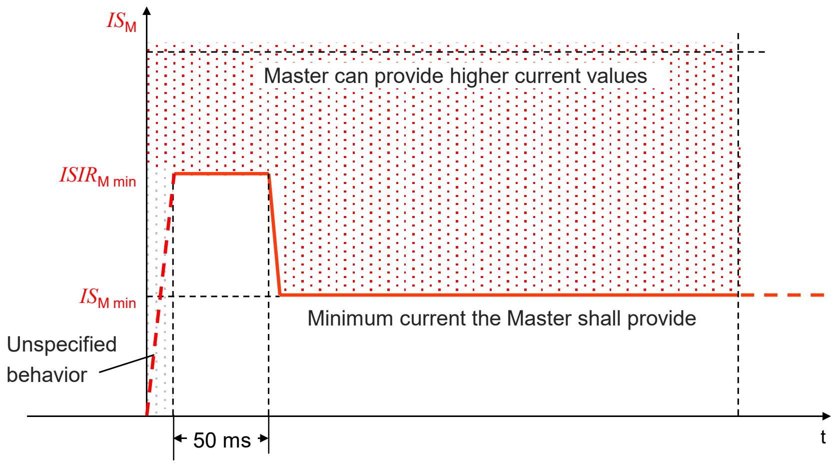


Figure 19 – Inrush current and charge (example)

5.3.2.4 Device

The definitions in Table 7 are valid for the electrical characteristics of a Device.

Table 7 – Electrical characteristics of a Device

Property	Designation	Minimum	Typical	Maximum	Unit	Remark
VS_D	Supply voltage	18	24	30	V	See Figure 17
QIS_D	Power-up charge consumption	n/a	n/a	70	mAs	See equation (1) and Table 8
ΔVS_D	Ripple	n/a	n/a	1,3	V_{pp}	Peak-to-peak absolute value limits shall not be exceeded. f_{ripple} = DC to 100 kHz
$VRQH_D$	Residual voltage 'H'	n/a	n/a	3	V	Voltage drop compared with $V+_D$ (IEC 60947-5-2)
$VRQL_D$	Residual voltage 'L'	n/a	n/a	3	V	Voltage drop compared with $V0_D$
IQH_D	DC driver current P-switching output ("On" state)	50	n/a	minimum ($IQPKL_M$)	mA	Minimum value due to fallback to digital input in accordance with IEC 61131-2, type 2
IQL_D	DC driver current N-switching output ("On" state)	0	n/a	minimum ($IQPKH_M$)	mA	Only for push-pull output stages
IQQ_D	Quiescent current to $V0_D$ ("Off" state)	0	n/a	15	mA	Pull-down or residual current with deactivated output driver stages
CQ_D	Input capacitance	0	n/a	1,0	nF	Effective capacitance between C/Q and L+ or L- of Device in receive state

The Device shall be able to reach a stable operational state (ready for Wake-up) consuming the maximum charge according to equation (1).

$$QIS_D = ISIR_M \times 50\ ms + (T_{RDL} - 50\ ms) \times IS_M \qquad (1)$$

Figure 20 shows how the power-on behavior of a Device is defined by the ramp-up time of the Power 1 supply and by the Device internal time to get ready for the wake-up operation.

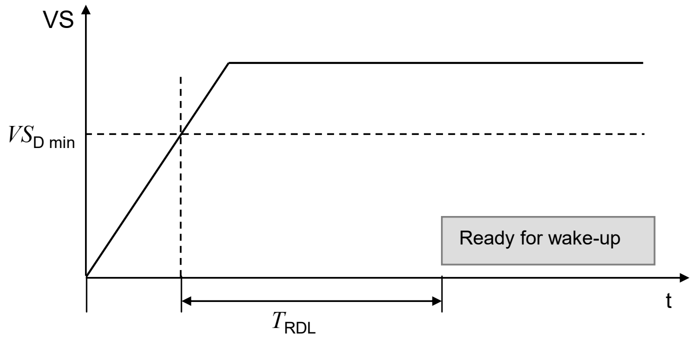


Figure 20 – Power-on timing for Power 1

Upon power-on it is mandatory for a Device to reach the wake-up ready state within the time limits specified in Table 8.

Table 8 – Power-on timing

Property	Designation	Minimum	Typical	Maximum	Unit	Remark
T_{RDL}	Wake-up readiness following power-on	n/a	n/a	300	ms	Device ramp-up time until it is ready for wake-up signal detection (See NOTE)
NOTE Equivalent to the time delay before availability in IEC 60947-5-2.						

The value of 1 nF for input capacitance CQ_D is applicable for a transmission rate of 230,4 kbit/s. It can be relaxed to a maximum of 10 nF in case of push-pull stage design when operating at lower transmission rates, provided that all dynamic parameter requirements in 5.3.3.2 are met.

5.3.3 Timing requirements

5.3.3.1 Transmission method

The “Non Return to Zero” (NRZ) modulation is used for the bit-by-bit coding. A logic value “1” corresponds to a voltage difference of 0 V between the C/Q line and L- line. A logic value "0" corresponds to a voltage difference of +24 V between the C/Q line and L- line.

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

A UART frame is used for the "data octet"-by-"data octet" coding. The format of the SDCI UART frame is a bit string structured as shown in Figure 21.

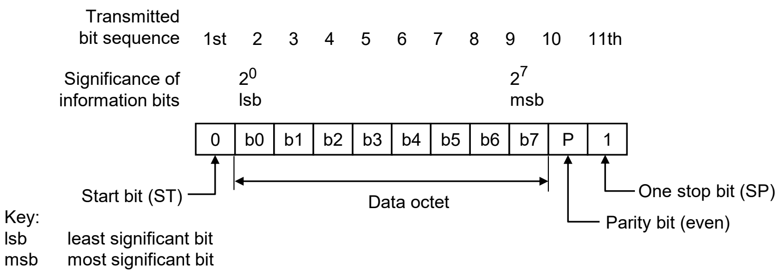


Figure 21 – Format of an SDCI UART frame

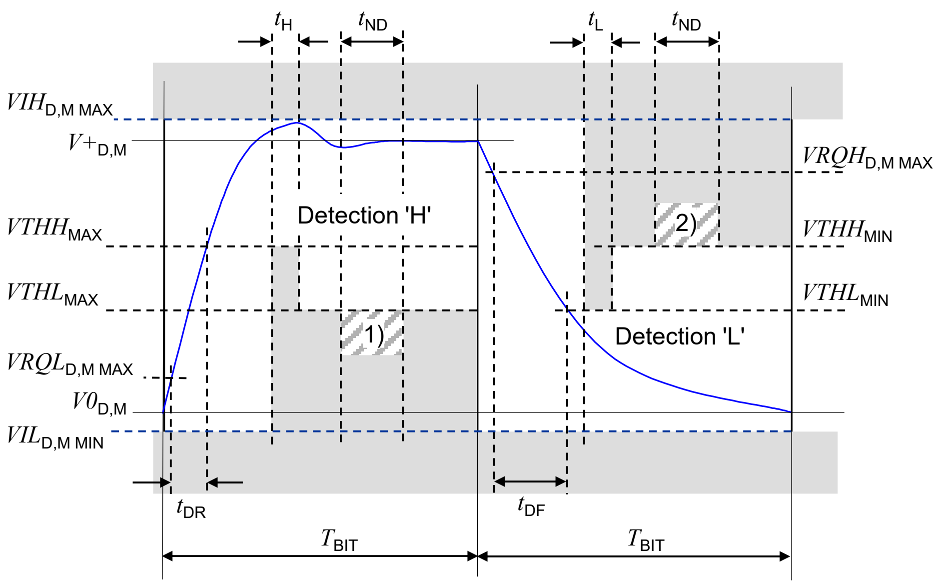
The definition of the UART frame format is based on ISO 1177 and ISO/IEC 2022.

5.3.3.2 Transmission characteristics

The timing characteristics of transmission are demonstrated in the form of an eye diagram with the permissible signal ranges (see Figure 22). These ranges are applicable for receiver in both the Master and the 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 bits as a valid signal shape within the permissible range of the eye diagram on the C/Q connection. Signal shapes in the “no detection” areas (below $VTHL_{\mathrm{MAX}}$ or above $VTHH_{\mathrm{MIN}}$ and within t_{ND}) shall not lead to invalid bits.



NOTE In the figure, 1) = no detection 'L'; and 2) = no detection 'H'

Figure 22 – Eye diagram for the 'H' and 'L' detection

In order for a UART frame to be detected correctly, a signal characteristic as demonstrated in Figure 23 is required on the receiver side. The signal delay time between the C/Q signal and the UART input shall be considered. Time T_{BIT} always indicates the receiver's bit rate.

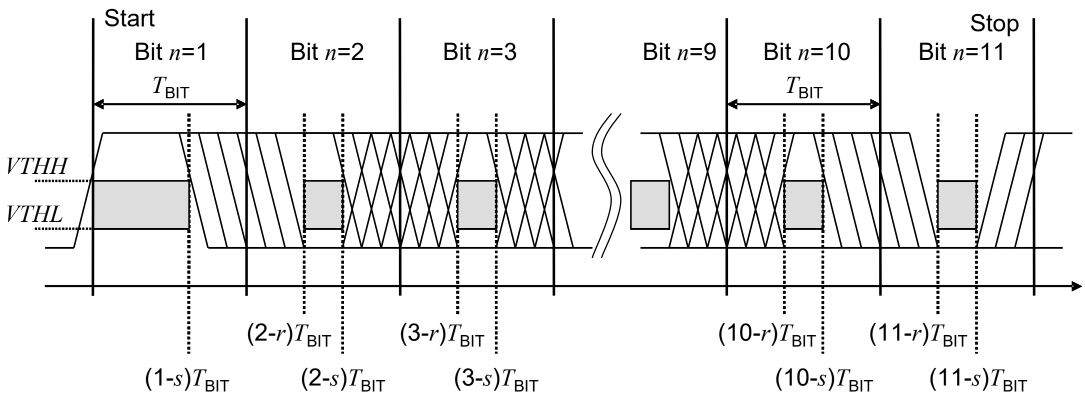


Figure 23 – Eye diagram for the correct detection of a UART frame

For every bit n in the bit sequence (n =1…11) of a UART frame, the time $(n\text{-}r)T_{\mathrm{BIT}}$ (see Table 9 for values of r) designates the time at the end of which a correct level shall be reached in the 'H' or 'L' ranges as demonstrated in the eye diagram in Figure 22. The time $(n\text{-}s)\ T_{\mathrm{BIT}}$ (see Table

9 for values of s) describes the time, which shall elapse before the level changes. Reference shall always be made to the eye diagram in Figure 22, where signal characteristics within a bit time are concerned.

This representation permits a variable weighting of the influence parameters "transmission rate accuracy", "bit-width distortion", and "slew rate" of the receiver.

Table 9 specifies the dynamic characteristics of the transmission.

Table 9 – Dynamic characteristics of the transmission

Property	Designation	Minimum	Typical	Maximum	Unit	Remark
f_{DTR}	transmission rate	n/a	4,8 38,4 230,4	n/a	kbit/s	COM1 COM2 COM3
T_{BIT}	Bit time at 4,8 kbit/s at 38,4 kbit/s at 230,4 kbit/s		 208,33 26,04 4,34		 µs µs µs	
Δf_{DTRM}	Master transmis-sion rate accuracy at 4,8 kbit/s at 38,4 kbit/s at 230,4 kbit/s	 -0,1 -0,1 -0,1	 n/a n/a n/a	 +0,1 +0,1 +0,1	 % % %	Tolerance of the transmission rate of the Master $\Delta T_{BIT}/T_{BIT}$
r	Start of detection time within a bit with reference to the raising edge of the start bit	0,65	n/a	n/a	-	Calculated in each case from the end of a bit at a UART sampling rate of 8
s	End of detection time within a bit with reference to the raising edge of the start bit	n/a	n/a	0,22	-	Calculated in each case from the end of a bit at a UART sampling rate of 8
T_{DR}	Rise time at 4,8 kbit/s at 38,4 kbit/s at 230,4 kbit/s	0 0 0 0	n/a n/a n/a n/a	0,20 41,7 5,2 869	T_{BIT} µs µs ns	With reference to the bit time unit. The minimum values could be critical to meet the requirements in H.1.5
t_{DF}	Fall time at 4,8 kbit/s at 38,4 kbit/s at 230,4 kbit/s	0 0 0 0	n/a n/a n/a n/a	0,20 41,7 5,2 869	T_{BIT} µs µs ns	With reference to the bit time unit. The minimum values could be critical to meet the requirements in H.1.5
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

The parameters 'r' and 's' apply to the respective Master or Device receiver side. This definition allows for a more flexible definition of oscillator accuracy, bit distortion and slewrate on the Device side. The overall bit-width distortion on the last bit of the UART frame shall provide a correct level in the range of Figure 23.

5.3.3.3 Wake-up current pulse

The wake-up feature is used to request that a Device goes to the COMx mode.

A service call (PL_WakeUp.req) from the DL initiates the wake-up process (see 5.2.2.2).

The wake-up request (WURQ) starts with a current pulse induced by the Master (port) for a time T_{WU}. The wake-up request comprises the following phases (see Figure 24):

a) Injection of a current IQ_{WU} by the Master depending on the level of 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" this is a current sink.

b) Delay time of the Device until it is ready to receive.

The wake-up request pulse can be detected by the Device through a voltage change on the C/Q line or evaluation of the current of the respective driver element within the time T_{WU}. Figure 24 shows examples for Devices with low output power.

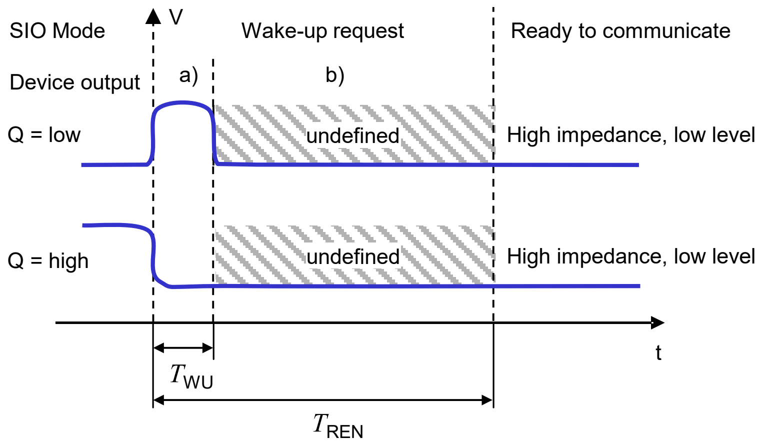


Figure 24 – Wake-up request

Table 10 specifies the current and timing properties associated with the wake-up request. See Table 6 for values of $IQPKL_M$ and $IQPKH_M$.

Table 10 – Wake-up request characteristics

Property	Designation	Minimum	Typical	Maximum	Unit	Remark
IQ_{WU}	Amplitude of Master's wake-up current pulse	$IQPKL_M$ or $IQPKH_M$	n/a	n/a	mA	Current pulse followed by switching status of Device
T_{WU}	Duration of Master's wake-up current pulse	75	n/a	85	µs	Master property
T_{REN}	Receive enable delay	n/a	n/a	500	µs	Device property

5.4 Power supply

5.4.1 Power supply options

The SDCI connection system provides dedicated power lines in addition to the signal line. The communication section of a Device shall always be powered by the Master using the power lines defined in the 3-wire connection system (Power 1).

Manufacturers/vendors shall emphasize this requirement within the user manual of the Master. Any additional measure for further increased robustness is within the responsibility of the designer/manufacturer of the Master.

The minimum supply current available from a Master port is specified in Table 6.

The application section of the Device may be powered in one of three ways:

- via the power lines of the SDCI 3-wire connection system (class A ports), using Power 1
- via the extra power lines of the SDCI 5-wire connection system (class B ports), using an extra power supply at the Master (Power 2) that shall be nonreactive, that means no impact on voltages and currents of Power 1 and on SDCI communications
- via a local power supply at the Device (design specific) that shall be nonreactive to Power 1, thus guaranteeing correct communication even in case of failing local power supply

It is recommended for Devices not to consume more than the minimum current a Master shall support (see Table 6). This ensures easiest handling of Master/Device systems without inquiries, checking, and calculations. Whenever a Device requires more than the minimum current the capabilities of the respective Master port and of its cabling shall be checked.

5.4.2 Port Class B

Figure 25 shows the layout of the two port classes A and B. Class B ports shall be marked to distinguish from Class A ports due to risks deriving from incompatibilities on pin 2 and pin 5.

Power 2 on port class B shall meet the following requirements

- electrical isolation of Power 2 from Power 1;
- degree of isolation according to IEC 60664 (clearance and creepage distances);
- electrical safety (SELV) according to IEC 61010-2-201:2017;
- direct current with P24 (+) and N24 (-);
- Device shall continue communicating correctly even in case of failing Power 2.

NOTE: EMC tests should consider maximum ripple and load switching

A Device designer shall ensure that Power 1 and Power 2 are always electrically isolated even in particular deployments/applications at the customer's site. Violation of this rule at one port can have impact on all other ports.

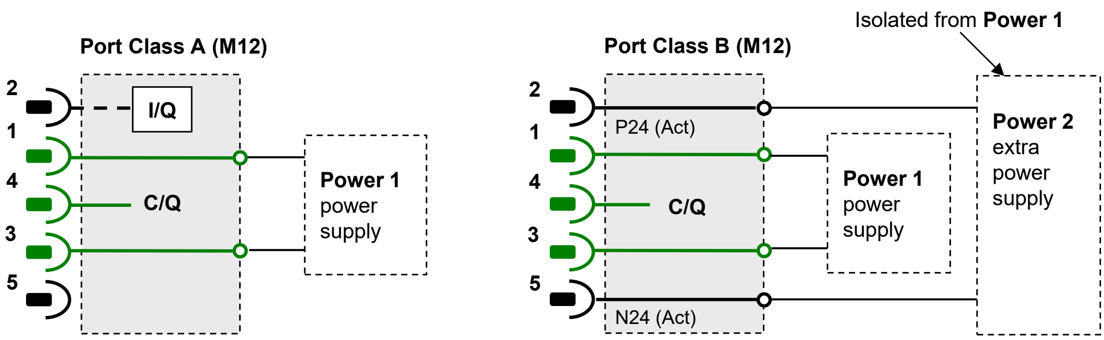


Figure 25 – Class A and B port definitions

Table 11 shows the electrical characteristics of a Master port class B (M12).

Table 11 – Electrical characteristic of a Master port class B

Property	Designation	Minimum	Typical	Maximum	Unit	Remark
$VP24_M$	Extra DC supply voltage for Devices	20[a)]	24	30	V	

Property	Designation	Minimum	Typical	Maximum	Unit	Remark
$IP24_M$	Extra DC supply current for Devices	1,6[b)]	n/a	3,5[c)]	A	

a) A minimum voltage shall be guaranteed for testing at maximum recommended supply current. At the Device side 18 V shall be available in this case.

b) Minimum current in order to guarantee a high degree of interoperability.

c) The recommended maximum current for a wire gauge of 0,34 mm^2 and standard M12 connector is 3,5 A. Maximum current depends on the type of connector, the wire gauge, maximum temperature, and simultaneity factor of the ports (check user manual of a Master).

In general, the requirements of Devices shall be checked whether they meet the available capabilities of the Master. In case a simultaneity factor for Master ports exists, it shall be documented in the user manual and be observed by the user of the Master.

5.4.3 Power-on requirements

The power-on requirements are specified in 5.3.2.3 and 5.3.2.4.

5.5 Medium

5.5.1 Connectors

The Master and Device pin assignment is based on the specifications in IEC 60947-5-2, with extensions specified in the paragraphs below.

Ports class A use M5, M8, and M12 connectors, with a maximum of five pins.

Ports class B only use M12 connectors with 5 pins.

M12 connectors are mechanically A-coded according to IEC 61076-2-101.

NOTE For legacy or compatibility reasons, direct wiring or different types of connectors can be used instead, provided that they do not violate the electrical characteristics and use signal naming specified in this standard.

Female connectors are assigned to the Master. Table 12 lists the pin assignments and

Figure **26** shows the layout and mechanical coding for M12, M8, and M5 connections.

Table 12 – Master pin assignments

Pin	Signal	Designation	Remark
1	L+	Power supply (+)	See Table 6
2	I/Q	NC/DI(OSSDe)/DO (port class A)	Option 1: NC (not connected) Option 2: DI Option 3: DI, then configured DO Option 4: OSSDe (see [10])
	P24	P24 (port class B)	Extra power supply for power Devices (port class B)
3	L-	Power supply (-)	See Table 6
4	C/Q	SIO(OSSDe)/SDCI	Standard I/O mode (DI/DO) or SDCI (see Table 6 for electrical characteristics of DO). See [10] for OSSDe definitions.
5	NC	NC (port class A)	Shall not be connected on the Master side (port class A).
	N24	N24 (port class B)	Reference potential to the extra power supply (port class B)
NOTE M12 is always a 5-pin version on the Master side (female).			

Figure **26** shows the layout of the two port classes A and B. Class B ports shall be marked to distinguish them from Class A ports, because of risks deriving from incompatibilities.

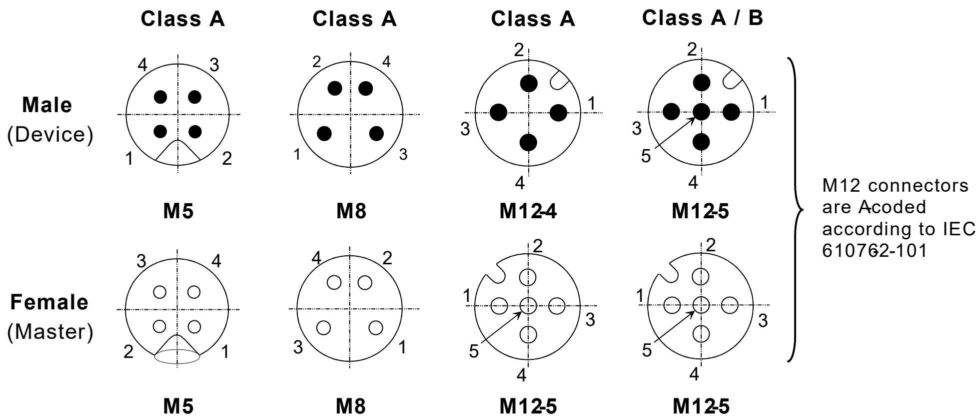


Figure 26 – Pin layout front view

Male connectors are assigned to the Device. Table 13 lists the pin assignments.

Table 13 – Device pin assignments

Pin	Signal	Designation	Remark
1	L+	Power supply (+)	See Table 7
2	I/Q [a]	NC/DI(OSSDe)/DO/ AI/AO (port class A)	Option 1: NC (not connected) Option 2: DI (Master's view) Option 3: DO (Master's view) Option 4: Analog signal (I / U) [d] Option 5: OSSDe (see [10])
	P24 [b]	P24 (port class B)	Extra power supply for power Devices (port class B)
3	L-	Power supply (-)	See Table 7
4	C/Q [c]	SIO(OSSDe)/SDCI	Standard I/O mode (DI/DO) or SDCI (see Table 6 for electrical characteristics of DO). See [10] for OSSDe definitions.
5	Q	ANY (port class A)	ANY (any functionality) [e]
	N24 [b]	N24 (port class B)	Reference to the extra power supply (port class B)

a) Device signals shall not interfere with the I/Q functionality of a Master. Devices shall withstand permanent DC (see Table 6) or P24 (see 5.4.2) on the Master side.

b) Devices relying on Port class A shall use 3-wire connection in this case in order to avoid bypassing electrical isolation

c) A Master shall always be able to establish and maintain SDCI communication without interferences

d) Typical for U is 0-10V, 1-5V, and for I is 0-20mA, 4-20mA

e) Device signals shall not interfere with the communication on the C/Q input of a Master. Devices shall withstand permanent N24 (see 5.4.2) on the Master side. Device output shall not impact the integrity of any Master.

5.5.2 Cable

The transmission medium for SDCI communication is a multi-wired cable with 3 or more wires. The definitions in the following paragraphs implicitly cover the static voltage definitions in Table 5 and Figure 17. To ensure functional reliability, the cable properties shall comply with Table 14.

Table 14 – Cable characteristics

Property	Minimum	Typical	Maximum	Unit
Length L	0	n/a	20	m
Overall loop resistance RL_{eff} [a]	n/a	n/a	6,0 (for a current of 200 mA) 1,2 (for a current of 1000 mA)	Ω

Property	Minimum	Typical	Maximum	Unit
Effective line capacitance CL_{eff}	n/a	n/a	3,0	nF (<1 MHz)
a) The overall loop resistence shall be rated such that minimum Device supply voltages are guaranteed at maximum supply current (see Table 7).				

The loop resistance RL_{eff} and the effective line capacitance CL_{eff} may be measured as demonstrated in Figure 27.

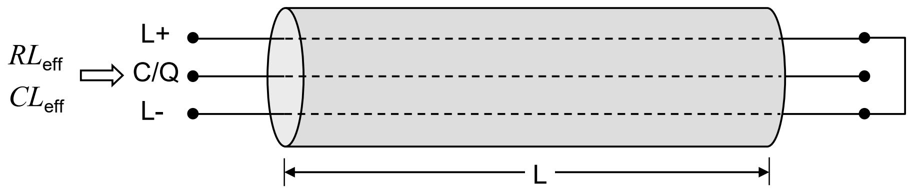


Figure 27 – Reference schematic for effective line capacitance and loop resistance

Table 15 shows the cable conductors and their assigned color codes.

Table 15 – Cable conductor assignments

Signal	Designation	Color	Remark
L-	Power supply (-)	Blue[a)]	SDCI 3-wire connection system
C/Q	Communication signal	Black[a)]	SDCI 3-wire connection system
L+	Power supply (+)	Brown[a)]	SDCI 3-wire connection system
I/Q	DI or DO	White[a)]	Optional
P24	Extra power supply (+)	Any other	Optional
N24	Extra power supply (-)	Any other	Optional
[a)] Corresponding to IEC 60947-5-2			

6 Standard Input and Output (SIO)

Figure 85 and Figure 96 demonstrate how the SIO mode allows a Device to bypass the SDCI communication layers and to map the DI or DO signal directly into the data exchange message of the upper level fieldbus or system. Changing between the SDCI and SIO mode is defined by the user configuration or implicitly by the services of the Master applications. The System Management takes care of the corresponding initialization or deactivation of the SDCI communication layers and the physical layer (mode switch). The characteristics of the interfaces for the DI and DO signals are derived from the caracteristics specified in IEC 61131-2 for type 1.

7 Data link layer (DL)

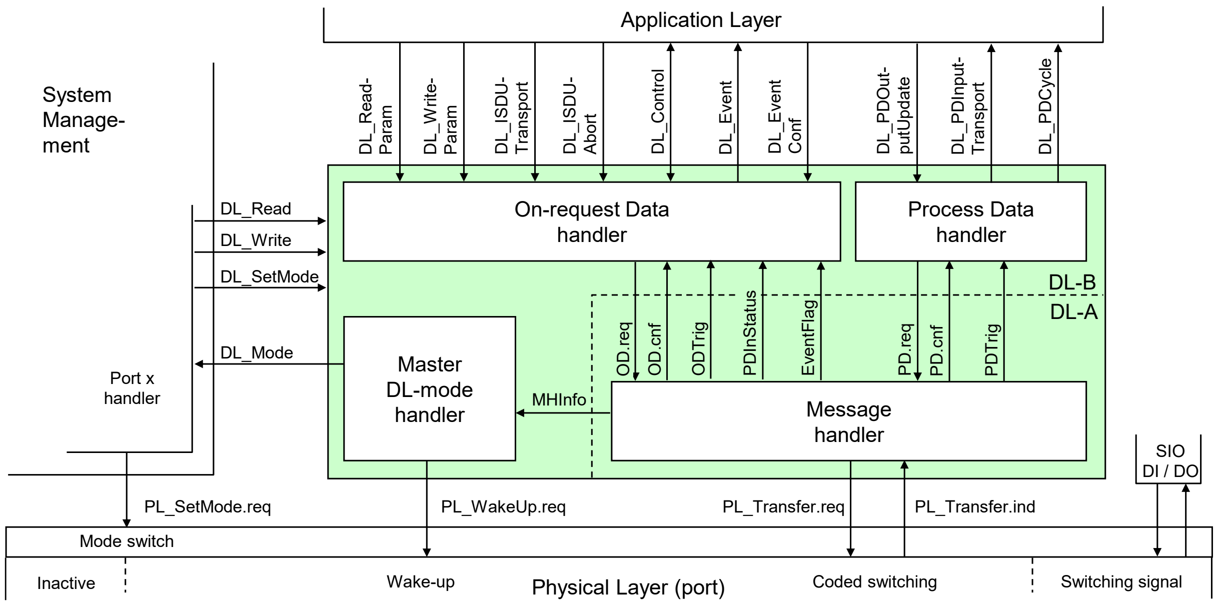
7.1 General

The data link layers of SDCI are concerned with the delivery of messages between a Master and a Device across the physical link. It uses several M-sequence ("message sequence") types for different data categories.

A set of DL-services is available to the application layer (AL) for the exchange of Process Data (PD) and On-request Data (OD). Another set of DL-services is available to System Management (SM) for the retrieval of Device communication and identification parameters and the setting of state machines within the DL. The DL uses PL-Services for controlling the physical layer (PL) and for exchanging UART frames. The DL takes care of the error detection of messages (whether internal or reported from the PL) and the appropriate remedial measures (e.g. retry).

The data link layers are structured due to the nature of the data categories into Process Data handlers and On-request Data handlers which are in turn using a message handler to deal with the requested transmission of messages. The special modes of Master ports such as wake-up, COMx, and SIO (disable communication) require a dedicated DL-mode handler within the Master DL. The special wake-up signal modulation requires signal detection on the Device side and thus a DL-mode handler within the Device DL. Each handler comprises its own state machine.

The data link layer is subdivided in a DL-A section with its own internal services and a DL-B section with the external services. The DL uses additional internal administrative calls between the handlers which are defined in the "internal items" section of the associated state-transition tables. Figure 28 shows an overview of the structure and the services of the Master's data link layer.



NOTE This figure uses the conventions in 3.3.5.

Figure 28 – Structure and services of the data link layer (Master)

Figure 29 shows an overview of the structure and the services of the Device's data link layer.

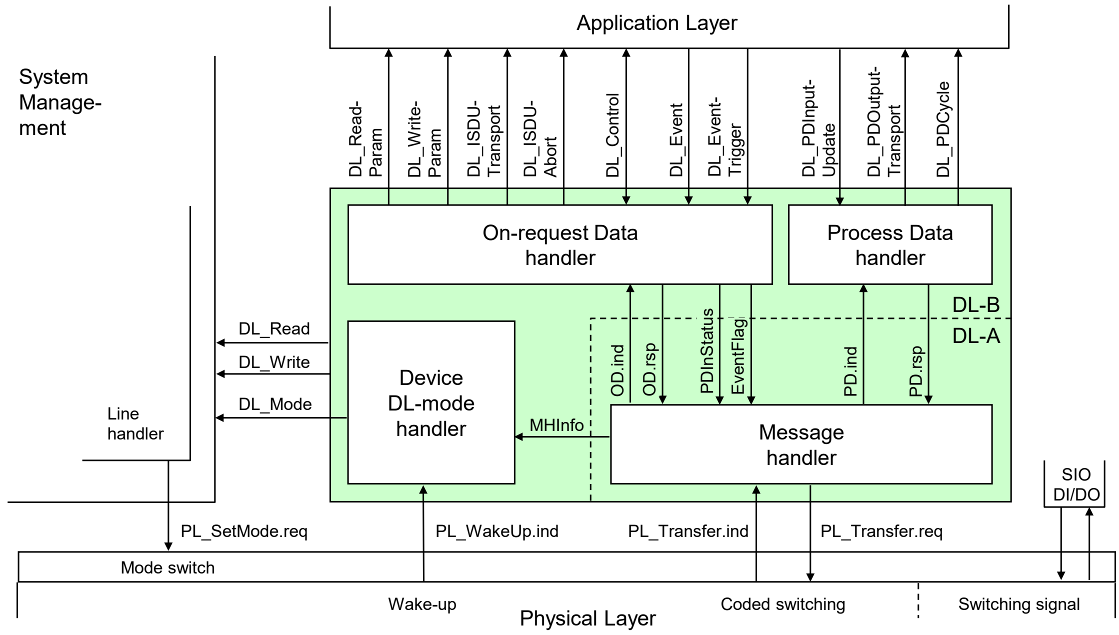


Figure 29 – Structure and services of the data link layer (Device)

7.2 Data link layer services

7.2.1 DL-B services

7.2.1.1 Overview of services within Master and Device

This clause defines the services of the data link layer to be provided to the application layer and System Management via its external interfaces. Table 16 lists the assignments of Master and Device to their roles as initiator or receiver for the individual DL services. Empty fields indicate no availability of this service on Master or Device.

Table 16 – Service assignments within Master and Device

Service name	Master	Device
DL_ReadParam	R	I
DL_WriteParam	R	I
DL_ISDUTransport	R	I
DL_ISDUAbort	R	I
DL_PDOutputUpdate	R	
DL_PDOutputTransport		I
DL_PDInputUpdate		R
DL_PDInputTransport	I	
DL_PDCycle	I	I
DL_SetMode	R	
DL_Mode	I	I
DL_Event	I	R
DL_EventConf	R	
DL_EventTrigger		R
DL_Control	I / R	R / I
DL_Read	R	I
DL_Write	R	I

Service name	Master	Device
Key (see 3.3.4) I Initiator of service R Receiver (responder) of service		

See 3.3 for conventions and how to read the service descriptions in 7.2, 8.2, 9.2.2, and 9.3.2.

7.2.1.2 DL_ReadParam

The DL_ReadParam service is used by the AL to read a parameter value from the Device via the page communication channel. The parameters of the service primitives are listed in Table 17.

Table 17 – DL_ReadParam

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

Argument
The service-specific parameters are transmitted in the argument.

Address
This parameter contains the address of the requested Device parameter, i.e. the Device parameter addresses within the page communication channel (see Table B.1).

Permitted values: 0 to 31

Result (+):
This selection parameter indicates that the service has been executed successfully.

Value
This parameter contains read Device parameter values.

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
NO_COMM (no communication available),
STATE_CONFLICT (service unavailable within current state)

7.2.1.3 DL_WriteParam

The DL_WriteParam service is used by the AL to write a parameter value to the Device via the page communication channel. The parameters of the service primitives are listed in Table 18.

Table 18 – DL_WriteParam

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

Argument
The service-specific parameters are transmitted in the argument.

Address
This parameter contains the address of the requested Device parameter, i.e. the Device parameter addresses within the page communication channel.

Permitted values: 16 to 31, in accordance with Device parameter access rights

Value
This parameter contains the Device parameter value to be written.

Result (+):
This selection parameter indicates that the service has been executed successfully.

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
NO_COMM (no communication available),
STATE_CONFLICT (service unavailable within current state)

7.2.1.4 DL_Read

The DL_Read service is used by System Management to read a Device parameter value via the page communication channel. The parameters of the service primitives are listed in Table 19.

Table 19 – DL_Read

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

Argument
The service-specific parameters are transmitted in the argument.

Address
This parameter contains the address of the requested Device parameter, i.e. the Device parameter addresses within the page communication channel (see Table B.1).

Permitted values: 0 to 15, in accordance with Device parameter access rights

Result (+):
This selection parameter indicates that the service has been executed successfully.

Value
This parameter contains read Device parameter values.

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
NO_COMM (no communication available),
STATE_CONFLICT (service unavailable within current state)

7.2.1.5 DL_Write

The DL_Write service is used by System Management to write a Device parameter value to the Device via the page communication channel. The parameters of the service primitives are listed in Table 20.

Table 20 – DL_Write

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

Argument
The service-specific parameters are transmitted in the argument.

Address
This parameter contains the address of the requested Device parameter, i.e. the Device parameter addresses within the page communication channel.

Permitted values: 0 to 15, in accordance with parameter access rights

Value
This parameter contains the Device parameter value to be written.

Result (+):
This selection parameter indicates that the service has been executed successfully.

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
NO_COMM (no communication available),
STATE_CONFLICT (service unavailable within current state)

7.2.1.6 DL_ISDUTransport

The DL_ISDUTransport service is used to transport an ISDU. This service is used by the Master to send a service request from the Master application layer to the Device. It is used by the Device to send a service response to the Master from the Device application layer. The parameters of the service primitives are listed in Table 21.

Table 21 – DL_ISDUTransport

Parameter name	.req	.ind	.cnf	.rsp
Argument	M	M		
ValueList	M	M		
Result (+)			S	S
Data			C	C
Qualifier			M	M
Result (-)			S	S
ISDUTransportErrorInfo			M	M

Argument
The service-specific parameters are transmitted in the argument.

ValueList
This parameter contains the relevant operating parameters

Parameter type: Record

Index
Permitted values: 2 to 65535 (See B.2.1 for constraints)

Subindex
Permitted values: 0 to 255

Data
Parameter type: Octet string

Direction
Permitted values:
READ (Read operation),
WRITE (Write operation)

Result (+):
This selection parameter indicates that the service has been executed successfully.

Data
Parameter type: Octet string

Qualifier
Permitted values: an I-Service Device response according to Table A.12

Result (-):
This selection parameter indicates that the service failed.

ISDUTransportErrorInfo
This parameter contains error information.

Permitted values:
NO_COMM (no communication available),
STATE_CONFLICT (service unavailable within current state),
ISDU_TIMEOUT (ISDU acknowledgment time elapsed, see Table 102),
ISDU_NOT_SUPPORTED (ISDU not implemented),
VALUE_OUT_OF_RANGE (Service parameter value violates range definitions)

7.2.1.7 DL_ISDUAbort

The DL_ISDUAbort service aborts the current ISDU transmission. This service has no parameters. The service primitives are listed in Table 22.

Table 22 – DL_ISDUAbort

Parameter name	.req	.cnf
<none>		

The service returns with the confirmation after abortion of the ISDU transmission.

7.2.1.8 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 23.

Table 23 – DL_PDOutputUpdate

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

Argument
The service-specific parameters are transmitted in the argument.

OutputData
This parameter contains the Process Data provided by the application layer.

Parameter type: Octet string

Result (+):
This selection parameter indicates that the service 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).

Permitted values:
YES (data transmission permitted),
NO (data transmission not permitted),

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
NO_COMM (no communication available),
STATE_CONFLICT (service unavailable within current state)

7.2.1.9 DL_PDOutputTransport

The data link layer on the Device uses the DL_PDOutputTransport service to transfer the content of output Process Data to the application layer (from Master to Device). The parameters of the service primitives are listed in Table 24.

Table 24 – DL_PDOutputTransport

Parameter name	.ind
Argument	M
OutputData	M

Argument
The service-specific parameters are transmitted in the argument.

OutputData
This parameter contains the Process Data to be transmitted to the application layer.

Parameter type: Octet string

7.2.1.10 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 25.

Table 25 – DL_PDInputUpdate

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

Argument
The service-specific parameters are transmitted in the argument.

InputData
This parameter contains the Process Data provided by the application layer.

Result (+):
This selection parameter indicates that the service 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).

Permitted values:
YES (data transmission permitted),
NO (data transmission not permitted),

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
NO_COMM (no communication available),
STATE_CONFLICT (service unavailable within current state)

7.2.1.11 DL_PDInputTransport

The data link layer on the Master uses the DL_PDInputTransport service to transfer the content of input data (Process Data from Device to Master) to the application layer. The parameters of the service primitives are listed in Table 26.

Table 26 – DL_PDInputTransport

Parameter name	.ind
Argument	M
InputData	M

Argument
The service-specific parameters are transmitted in the argument.

InputData
This parameter contains the Process Data to be transmitted to the application layer.

Parameter type: Octet string

7.2.1.12 DL_PDCycle

The data link layer uses the DL_PDCycle service to indicate the end of a Process Data cycle to the application layer. This service has no parameters. The service primitives are listed in Table 27.

Table 27 – DL_PDCycle

Parameter name	.ind
<none>	

7.2.1.13 DL_SetMode

The DL_SetMode service is used by System Management to set up the data link layer's state machines and to send the characteristic values required for operation to the data link layer. The parameters of the service primitives are listed in Table 28.

Table 28 – DL_SetMode

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

Argument
The service-specific parameters are transmitted in the argument.

Mode
This parameter indicates the requested mode of the Master's DL on an individual port.

Permitted values:
INACTIVE (handler shall change to the INACTIVE state),
STARTUP (handler shall change to STARTUP state),
PREOPERATE (handler shall change to PREOPERATE state),
OPERATE (handler shall change to OPERATE state)

ValueList
This parameter contains the relevant operating parameters.

Data structure: record

M-sequenceTime: (to be propagated to message handler)

M-sequenceType: (to be propagated to message handler)
Permitted values:
TYPE_0,
TYPE_1_1, TYPE_1_2, TYPE_1_V,
TYPE_2_1, TYPE_2_2, TYPE_2_3, TYPE_2_4, TYPE_2_5, TYPE_2_V
(TYPE_1_1 forces interleave mode of Process and On-request Data transmission, see 7.3.4.2)

PDInputLength: (to be propagated to message handler)

PDOutputLength: (to be propagated to message handler)

OnReqDataLengthPerMessage: (to be propagated to message handler)

Result (+):
This selection parameter indicates that the service has been executed successfully.

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
STATE_CONFLICT (service unavailable within current state),
PARAMETER_CONFLICT (consistency of parameter set violated)

7.2.1.14 DL_Mode

The DL uses the DL_Mode service to report to System Management that a certain operating status has been reached. The parameters of the service primitives are listed in Table 29.

Table 29 – DL_Mode

Parameter name	.ind
Argument	M
RealMode	M

Argument
The service-specific parameters are transmitted in the argument.

RealMode
This parameter indicates the status of the DL-mode handler.

Permitted values:
INACTIVE (Handler changed to the INACTIVE state)
COM1 (COM1 mode established)
COM2 (COM2 mode established)
COM3 (COM3 mode established)
COMLOST (Lost communication)
ESTABCOM (Handler changed to the EstablishCom state)
STARTUP (Handler changed to the STARTUP state)
PREOPERATE (Handler changed to the PREOPERATE state)
OPERATE (Handler changed to the OPERATE state)

7.2.1.15 DL_Event

The service DL_Event indicates a pending status or error information. The cause for an Event is located in a Device and the Device application triggers the Event transfer. The parameters of the service primitives are listed in Table 30.

Table 30 – DL_Event

Parameter name	.req	.ind
Argument	M	M
Instance	M	M
Type	M	M
Mode	M	M
EventCode	M	M
EventsLeft		M

Argument
The service-specific parameters are transmitted in the argument.

Instance
This parameter indicates the Event source.

Permitted values: Application (see Table A.17)

Type
This parameter indicates the Event category.

Permitted values: ERROR, WARNING, NOTIFICATION (see Table A.19)

Mode
This parameter indicates the Event mode.

Permitted values: SINGLESHOT, APPEARS, DISAPPEARS (see Table A.20)

EventCode
This parameter contains a code identifying a certain Event (see Table D.1).

Parameter type: 16-bit unsigned integer

EventsLeft
This parameter indicates the number of unprocessed Events.

7.2.1.16 DL_EventConf

The DL_EventConf service confirms the transmitted Events via the Event handler. This service has no parameters. The service primitives are listed in Table 31.

Table 31 – DL_EventConf

Parameter name	.req	.cnf
<none>		

7.2.1.17 DL_EventTrigger

The DL_EventTrigger request starts the Event signaling (see Event flag in Figure A.3) and freezes the Event memory within the DL. The confirmation is returned after the activated Events have been processed. Additional DL_EventTrigger requests are ignored until the previous one has been confirmed (see 7.3.8, 8.3.3 and Figure 66). This service has no parameters. The service primitives are listed in Table 32.

Table 32 – DL_EventTrigger

Parameter name	.req	.cnf
<none>		

7.2.1.18 DL_Control

The Master uses the DL_Control service to convey control information via the MasterCommand mechanism to the corresponding Device application and to get control information via the PD status flag mechanism (see A.1.5) and the PDInStatus service (see 7.2.2.5). The parameters of the service primitives are listed in Table 33.

Table 33 – 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 indicates the qualifier status of the Process Data (PD)

Permitted values:
VALID (Input Process Data valid; see 7.2.2.5, 8.2.2.12)
INVALID (Input Process Data invalid)
PDOUTVALID (Output Process Data valid; see 7.3.7.1)
PDOUTINVALID (Output Process Data invalid or missing)

7.2.2 DL-A services

7.2.2.1 Overview

According to 7.1 the data link layer is split into the upper layer DL-B and the lower layer DL-A. The layer DL-A comprises the message handler as shown in Figure 28 and Figure 29.

The Master message handler encodes commands and data into messages and sends these to the connected Device via the physical layer. It receives messages from the Device via the physical layer and forwards their content to the corresponding handlers in the form of a confirmation. When the "Event flag" is set in a Device message (see A.1.5), the Master message handler invokes an EventFlag service to prompt the Event handler.

The Master message handler shall employ a retry strategy following a corrupted message, i.e. upon receiving an incorrect checksum from a Device, or no checksum at all. In these cases, the Master shall repeat the Master message two times (see Table 102). If the retries are not successful, a negative confirmation shall be provided, and the Master shall re-initiate the communication via the Port-x handler beginning with a wake-up.

After a start-up phase the message handler performs cyclic operation with the M-sequence type and cycle time provided by the DL_SetMode service.

Table 34 lists the assignment of Master and Device to their roles as initiator (I) or receiver (R) in the context of the execution of their individual DL-A services.

Table 34 – DL-A services within Master and Device

Service name	Master	Device
OD	R	I
PD	R	I
EventFlag	I	R
PDInStatus	I	R
MHInfo	I	I
ODTrig	I	
PDTrig	I	

7.2.2.2 OD

The OD service is used to set up the On-request Data for the next message to be sent. In turn, the confirmation of the service contains the data from the receiver. The parameters of the service primitives are listed in Table 35.

Table 35 – OD

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M		
RWDirection	M	M		
ComChannel	M	M		
AddressCtrl	M	M		
Length	M	M		
Data	C	C		
Result (+)			S	S
Data			C	C(=)
Length			M	M
Result (-)			S	S
ErrorInfo			M	M(=)

Argument
The service-specific parameters are transmitted in the argument.

RWDirection
This parameter indicates the read or writes direction.

Permitted values:
READ (Read operation),
WRITE (Write operation)

ComChannel
This parameter indicates the selected communication channel for the transmission.

Permitted values: DIAGNOSIS, PAGE, ISDU (see Table A.1)

AddressCtrl
This parameter contains the address or flow control value (see A.1.2).

Permitted values: 0 to 31

Length
This parameter contains the length of data to transmit.

Permitted values: 0 to 32

Data
This parameter contains the data to transmit.

Data type: Octet string

Result (+):
This selection parameter indicates that the service has been executed successfully.

Data
This parameter contains the read data values.

Length
This parameter contains the length of the received data package.

Permitted values: 0 to 32

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
NO_COMM (no communication available),
STATE_CONFLICT (service unavailable within current state)

7.2.2.3 PD

The PD service is used to setup the Process Data to be sent through the process communication channel. The confirmation of the service contains the data from the receiver. The parameters of the service primitives are listed in Table 36.

Table 36 – PD

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M		
PDInAddress	C	C(=)		
PDInLength	C	C(=)		
PDOut	C	C(=)		
PDOutAddress	C	C(=)		
PDOutLength	C	C(=)		
Result (+)			S	S
PDIn			C	C(=)
Result (-)			S	S
ErrorInfo			M	M(=)

Argument
The service-specific parameters are transmitted in the argument.

PDInAddress
This parameter contains the address of the requested input Process Data (see 7.3.4.2).

PDInLength
This parameter contains the length of the requested input Process Data.

Permitted values: 0 to 32

PDOut
This parameter contains the Process Data to be transferred from Master to Device.

Data type: Octet string

PDOutAddress
This parameter contains the address of the transmitted output Process Data (see 7.3.4.2).

PDOutLength
This parameter contains the length of the transmitted output Process Data.

Permitted values: 0 to 32

Result (+)
This selection parameter indicates that the service has been executed successfully.

PDIn
This parameter contains the Process Data to be transferred from Device to Master.

Data type: Octet string

Result (-)
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
NO_COMM (no communication available),
STATE_CONFLICT (service unavailable within current state)

7.2.2.4 EventFlag

The EventFlag service sets or signals the status of the "Event flag" (see A.1.5) during cyclic communication. The parameters of the service primitives are listed in Table 37.

Table 37 – EventFlag

Parameter name	.ind	.req
Argument		
Flag	M	M

Argument
The service-specific parameters are transmitted in the argument.

Flag
This parameter contains the value of the "Event flag".

Permitted values:
TRUE ("Event flag" = 1)
FALSE ("Event flag" = 0)

7.2.2.5 PDInStatus

The service PDInStatus sets and signals the validity qualifier of the input Process Data. The parameters of the service primitives are listed in Table 38.

Table 38 – PDInStatus

Parameter name	.req	.ind
Argument Status	 M	 M

Argument
The service-specific parameters are transmitted in the argument.

Status
This parameter contains the validity indication of the transmitted input Process Data.

Permitted values:
VALID (Input Process Data valid based on PD status flag (see A.1.5); see 7.2.1.18)
INVALID (Input Process Data invalid)

7.2.2.6 MHInfo

The service MHInfo signals an exceptional operation within the message handler. The parameters of the service are listed in Table 39.

Table 39 – MHInfo

Parameter name	.ind
Argument MHInfo	 M

Argument
The service-specific parameters are transmitted in the argument.

MHInfo
This parameter contains the exception indication of the message handler.

Permitted values:
COMLOST (lost communication),
ILLEGAL_MESSAGETYPE (unexpected M-sequence type detected)
CHECKSUM_MISMATCH (Checksum error detected)

7.2.2.7 ODTrig

The service ODTrig is only available on the Master. The service triggers the On-request Data handler and the ISDU, Command, or Event handler currently in charge to provide the On-request Data (via the OD service) for the next Master message. The parameters of the service are listed in Table 40.

Table 40 – ODTrig

Parameter name	.ind
Argument DataLength	 M

Argument
The service-specific parameters are transmitted in the argument.

DataLength
This parameter contains the available space for On-request Data (OD) per message.

7.2.2.8 PDTrig

The service PDTrig is only available on the Master. The service triggers the Process Data handler to provide the Process Data (PD) for the next Master message.

The parameters of the service are listed in Table 41.

Table 41 – PDTrig

Parameter name	.ind
Argument DataLength	 M

Argument
The service-specific parameters are transmitted in the argument.

DataLength
This parameter contains the available space for Process Data (PD) per message.

7.3 Data link layer protocol

7.3.1 Overview

Figure 28 and Figure 29 are showing the structure of the data link layer and its components; a DL-mode handler, a message handler, a Process Data handler, and an On-request Data handler to provide the specified services. Subclauses 7.3.2 to 7.3.8 define the behaviour (dynamics) of these handlers by means of UML state machines and transition tables.

The On-request Data handler supports three independent types of data: ISDU, command and Event. Therefore, three additional state machines are working together with the On-request Data handler state machine as shown in Figure 30.

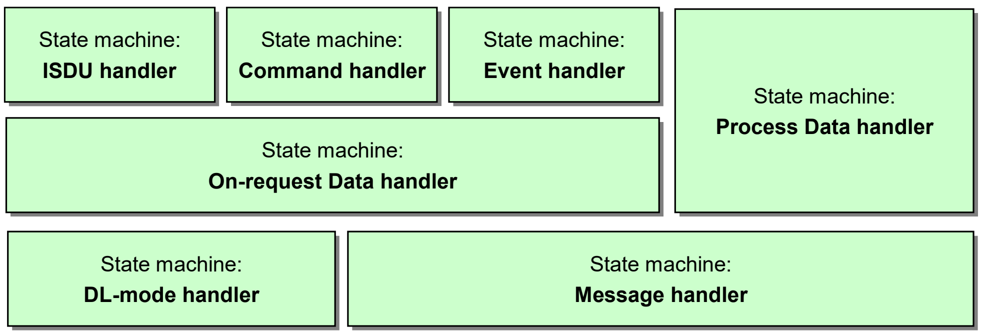


Figure 30 – State machines of the data link layer

Supplementary sequence or activity diagrams are demonstrating certain use cases. See IEC/TR 62390 and ISO/IEC 19505.

The elements each handler is dealing with, such as messages, wake-up procedures, interleave mode, ISDU (Indexed Service Data Units), and Events are defined within the context of the respective handler.

7.3.2 DL-mode handler

7.3.2.1 General

The Master DL-mode handler shown in Figure 28 is responsible to setup the SDCI communication using services of the Physical Layer (PL) and internal administrative calls to control and monitor the message handler as well as the states of other handlers.

The Device DL-mode handler shown in Figure 29 is responsible to detect a wake-up request and to establish communication. It receives MasterCommands to synchronize with the Master DL-mode handler states STARTUP, PREOPERATE, and OPERATE and manages the activation and de-activation of handlers as appropriate.

7.3.2.2 Wake-up procedures and Device conformity rules

System Management triggers the following actions on the data link layer with the help of the DL_SetMode service (requested mode = STARTUP).

The Master DL-mode handler tries to establish communication via a wake-up request (PL_WakeUp.req) followed by a test message with M-sequence TYPE_0 (read "MinCycleTime") according to the sequence shown in Figure 31.

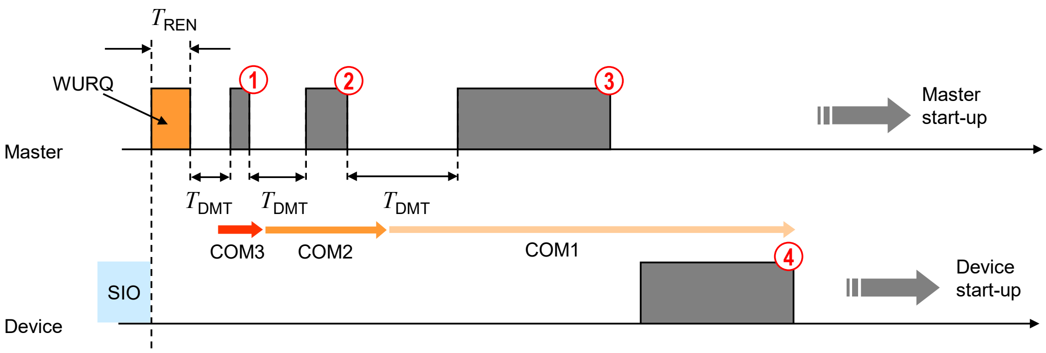


Figure 31 – Example of an attempt to establish communication

After the wake-up request (WURQ), specified in 5.3.3.3, the DL-mode handler requests the message handler to send the first test message after a time T_{REN} (see Table 10) and T_{DMT} (see Table 42). The specified transmission rates of COM1, COM2, and COM3 are used in descending order until a response is obtained, as shown in the example of Figure 31:

Step ①: Master message with transmission rate of COM3 (see Table 9).

Step ②: Master message with transmission rate of COM2 (see Table 9).

Step ③: Master message with transmission rate of COM1 (see Table 9).

Step ④: Device response message with transmission rate of COM1.

Before initiating a (new) message, the DL-mode handler shall wait at least for a time of T_{DMT}. T_{DMT} is specified in Table 42.

The following conformity rule applies for Devices regarding support of transmission rates:

- a Device shall support only one of the transmission rates of COM1, COM2, or COM3.

If an attempt to establish communication fails, the Master DL-mode handler shall not start a new retry wake-up procedure until after a time T_{DWU} as shown in Figure 32 and specified in Table 42.

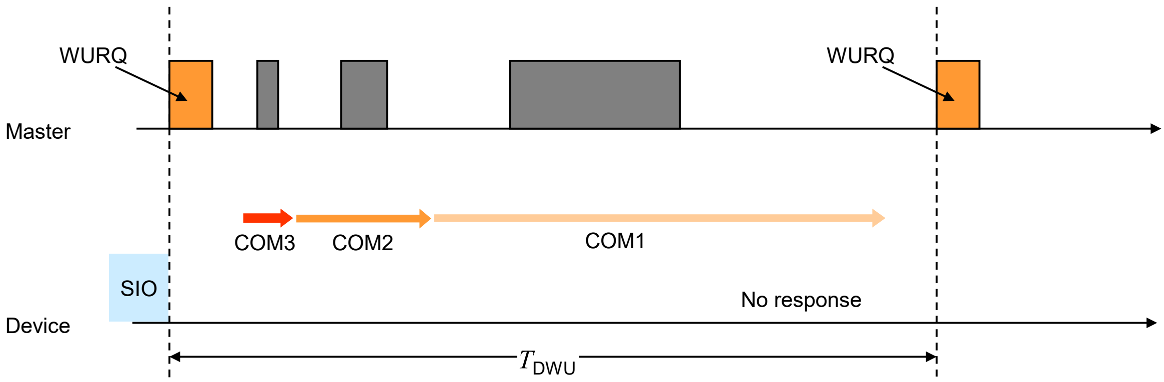


Figure 32 – Failed attempt to establish communication

The Master shall make up to n_{WU}+1 successive wake-up requests as shown in Figure 33. If this initial wake-up retry sequence fails, the Device shall reset its C/Q line to SIO mode after a time T_{DSIO} (T_{DSIO} is retrigged in the Device after each detected WURQ). The Master shall not trigger a new wake-up retry sequence until after a time T_{SD}.

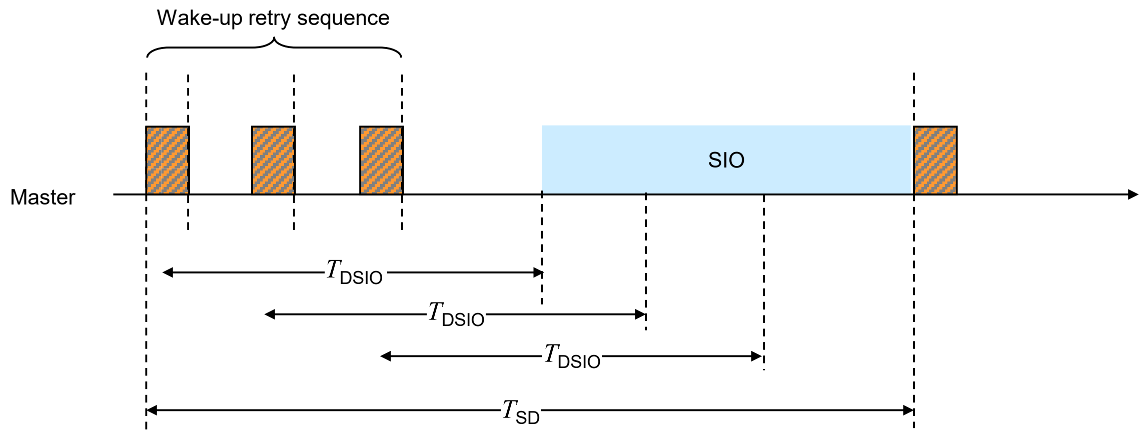


Figure 33 – Retry strategy to establish communication

The DL of the Master shall request the PL to go to Inactive mode after a failed wake-up retry sequence.

The values for the timings of the wake-up procedures and retries are specified in Table 10 and Table 42. They are defined from a Master's point of view.

Table 42 – Wake-up procedure and retry characteristics

Property	Designation	Minimum	Typical	Maximum	Unit	Remark
T_{DMT}	Master message delay	27	n/a	37	T_{BIT}	Bit time of subsequent data transmission rate
T_{DSIO}	Standard IO delay	60	n/a	300	ms	After T_{DSIO} the Device falls back to SIO mode (if supported)
T_{DWU}	Wake-up retry delay	30	n/a	50	ms	After T_{DWU} the Master repeats the wake-up request
n_{WU}	Wake-up retry count	2	2	2		Number of wake-up request retries
T_{SD}	Device detection time	0,5	n/a	1	s	Time between 2 wake-up request sequences (See NOTE)
NOTE Characteristic of the Master.						

The Master's data link layer shall stop the establishing communication procedure once it finds a communicating Device and shall report the detected COMx-Mode to System Management using a DL_Mode indication. If the procedure fails, a corresponding error is reported using the same service.

7.3.2.3 Fallback procedure

System Management induces the following actions on the data link layer with the help of the DL_SetMode service (mode = INACTIVE):

- A MasterCommand "Fallback" (see Table B.2) forces the Device to change to the SIO mode.
- The Device shall accomplish the transition to the SIO mode after 3 MasterCycleTimes and/or within maximum T_{FBD} after the MasterCommand "Fallback". This allows for possible retries if the MasterCommand failed indicated through a negative Device response.
- The Master shall ensure waiting at least maximum T_{FBD} before initiating the next start-up procedure.

Figure 34 shows the fallback procedure and its retry and timing constraints.

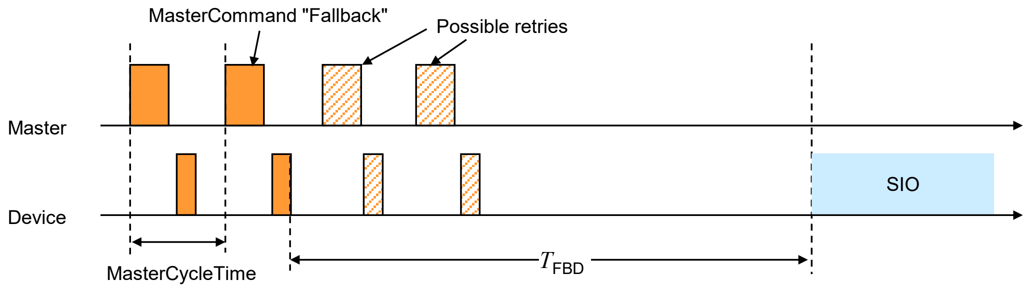


Figure 34 – Fallback procedure

Table 43 specifies the fallback timing characteristics. See A.2.6 for details.

Table 43 – Fallback timing characteristics

Property	Designation	Minimum	Typical	Maximum	Unit	Remark
T_{FBD}	Fallback delay	3 MasterCycle-Times (OPERATE) or 3 $T_{initcyc}$ (PREOPERATE)	n/a	500	ms	After a time T_{FBD} the Device shall be switched to SIO mode (see Figure 34)

7.3.2.4 State machine of the Master DL-mode handler

Figure 35 shows the state machine of the Master DL-mode handler.

NOTE The conventions of the UML diagram types are defined in 3.3.7.

After reception of the service DL_SetMode_STARTUP from System Management, the DL-mode handler shall first create a wake-up current pulse via the PL_WakeUp service and then establish communication. This procedure is specified in submachine 1 in Figure 36.

The purpose of state "Startup_2" is to check a Device's identity via the data of the Direct Parameter page (see Figure 6). In state "PreOperate_3", the Master assigns parameters to the Device using ISDUs. Cyclic exchange of Process Data is performed in state "Operate". Within this state additional On-request Data such as ISDUs, commands, and Events can be transmitted using appropriate M-sequence types (see Figure 39).

In state PreOperate_3 and Operate_4 different sets of handlers within the Master are activated.

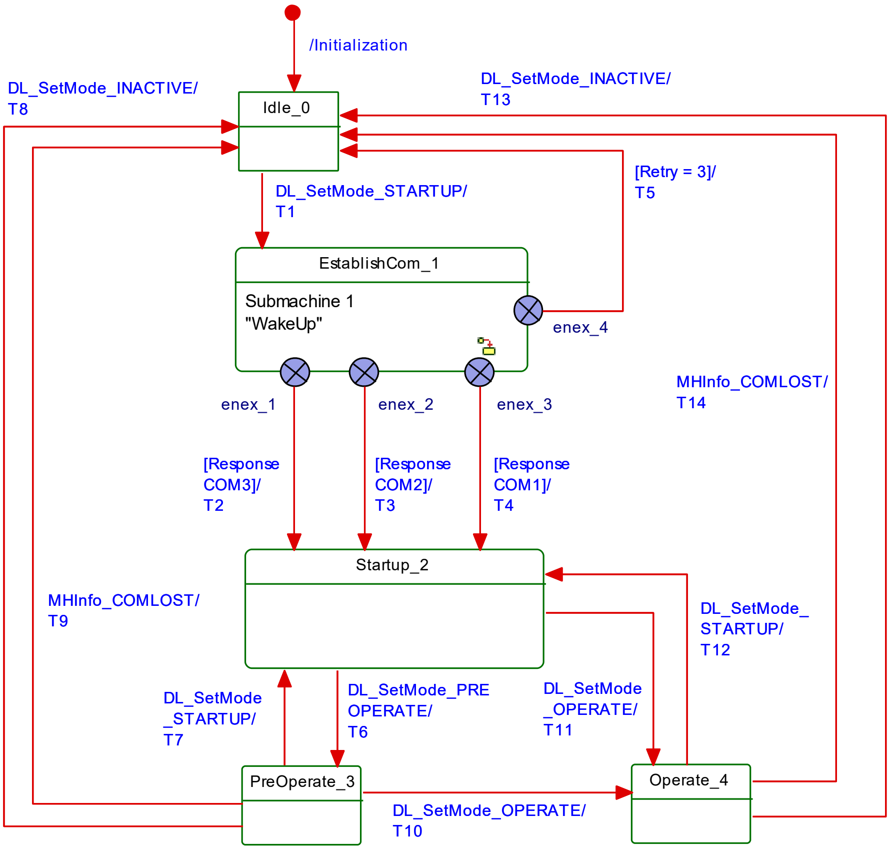


Figure 35 – State machine of the Master DL-mode handler

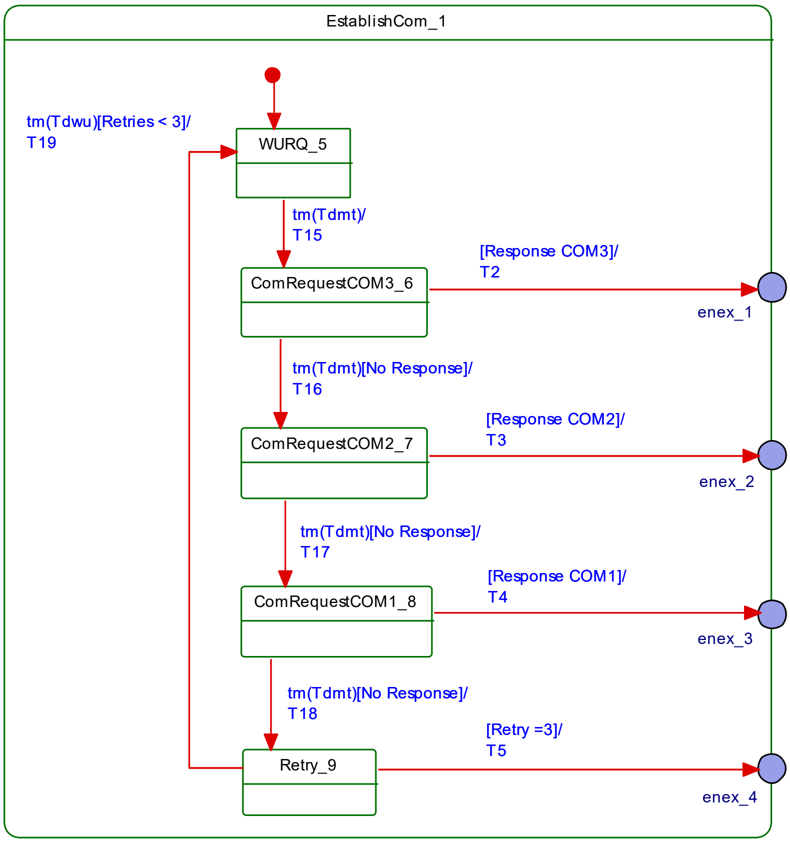


Figure 36 – Submachine 1 to establish communication

Table 44 shows the state transition tables of the Master DL-mode handler.

Table 44 – State transition tables of the Master DL-mode handler

STATE NAME	STATE DESCRIPTION
Idle_0	Waiting on wakeup request from System Management (SM): DL_SetMode (STARTUP)
EstablishComm_1	Perform wakeup procedure (submachine 1)
Startup_2	System Management uses the STARTUP state for Device identification, check, and communication configuration (see Figure 71)
Preoperate_3	On-request Data exchange (parameter, commands, Events) without Process Data
Operate_4	Process Data and On-request Data exchange (parameter, commands, Events)
SM: WURQ_5	Create wakeup current pulse: Invoke service PL-Wake-Up (see Figure 12 and 5.3.3.3) and wait T_{DMT} (see Table 42).
SM: ComRequestCOM3_6	Try test message with transmission rate of COM3 via the message handler: Call MH_Conf_COMx (see Figure 40) and wait T_{DMT} (see Table 42).
SM: ComRequestCOM2_7	Try test message with transmission rate of COM2 via the message handler: Call MH_Conf_COMx (see Figure 40) and wait T_{DMT} (see Table 42).
SM: ComRequestCOM1_8	Try test message with transmission rate of COM1 via the message handler: Call MH_Conf_COMx (see Figure 40) and wait T_{DMT} (see Table 42).
SM: Retry_9	Check number of Retries

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	Set Retry = 0.
T2	1	2	Transmission rate of COM3 successful. Message handler activated and configured to COM3 (see Figure 40, Transition T2). Activate command handler (call CH_Conf_ACTIVE in Figure 53). Return DL_Mode.ind (STARTUP) and DL_Mode.ind (COM3) to SM.
T3	1	2	Transmission rate of COM2 successful. Message handler activated and configured to COM2 (see Figure 40, Transition T2). Activate command handler (call CH_Conf_ACTIVE in Figure 53). Return DL_Mode.ind (STARTUP) and DL_Mode.ind (COM2) to SM.
T4	1	2	Transmission rate of COM1 successful. Message handler activated and configured to COM1 (see Figure 40, Transition T2). Activate command handler (call CH_Conf_ACTIVE in Figure 53). Return DL_Mode.ind (STARTUP) and DL_Mode.ind (COM1) to SM.
T5	1	0	Return DL_Mode.ind (INACTIVE) to SM.
T6	2	3	SM requested the PREOPERATE state. Activate On-request Data (call OH_Conf_ACTIVE in Figure 48), ISDU (call IH_Conf_ACTIVE in Figure 51), and Event handler (call EH_Conf_ACTIVE in Figure 55). Change message handler state to PREOPERATE (call MH_Conf_PREOPERATE in Figure 40). Return DL_Mode.ind (PREOPERATE) to SM.
T7	3	2	SM requested the STARTUP state. Change message handler state to STARTUP (call MH_Conf_STARTUP in Figure 40). Deactivate On-request Data (call OH_Conf_INACTIVE in Figure 48), ISDU (call IH_Conf_IN-ACTIVE in Figure 51), and Event handler (call EH_Conf_INACTIVE in Figure 55). Return DL_Mode.ind (STARTUP) to SM.
T8	3	0	SM requested the SIO mode. Deactivate all handlers (call xx_Conf_INACTIVE). Return DL_Mode.ind (INACTIVE) to SM. See 7.3.2.3.
T9	3	0	Message handler informs about lost communication via the DL-A service MHInfo (COMLOST). Deactivate all handlers (call xx_Conf_INACTIVE). Return DL_Mode.ind (COMLOST) to SM.
T10	3	4	SM requested the OPERATE state. Activate the Process Data handler (call PD_Conf_SINGLE if M-sequence type = TYPE_2_x, or PD_Conf_INTERLEAVE if M-sequence type = TYPE_1_1 in Figure 46). Change message handler state to OPERATE (call MH_Conf_OPERATE in Figure 40). Return DL_Mode.ind (OPERATE) to SM.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T11	2	4	SM requested the OPERATE state. Activate the Process Data handler (call PD_Conf_SINGLE or PD_Conf_INTERLEAVE in Figure 46 according to the Master port configuration). Activate On-request Data (call OH_Conf_ACTIVE in Figure 48), ISDU (call IH_Conf_ACTIVE in Figure 51), and Event handler (call EH_Conf_ACTIVE in Figure 55). Change message handler state to OPERATE (call MH_Conf_OPERATE in Figure 40). Return DL_Mode.ind (OPERATE) to SM.
T12	4	2	SM requested the STARTUP state. Change message handler state to STARTUP (call MH_Conf_STARTUP in Figure 40). Deactivate Process Data (call PD_Conf_INACTIVE in Figure 46), On-request Data (call OH_Conf_INACTIVE in Figure 48), ISDU (call IH_Conf_INACTIVE in Figure 51), and Event handler (call EH_Conf_INACTIVE in Figure 55). Return DL_Mode.ind (STARTUP) to SM.
T13	4	0	SM requested the SIO state. Deactivate all handlers (call xx_Conf_INACTIVE). Return DL_Mode.ind (INACTIVE) to SM. See 7.3.2.3.
T14	4	0	Message handler informs about lost communication via the DL-A service MHInfo (COMLOST). Deactivate all handlers (call xx_Conf_INACTIVE). Return DL_Mode.ind (COMLOST) to SM.
T15	5	6	Set transmission rate of COM3 mode.
T16	6	7	Set transmission rate of COM2 mode.
T17	7	8	Set transmission rate of COM1 mode.
T18	8	9	Increment Retry
T19	9	5	-

INTERNAL ITEMS	TYPE	DEFINITION
MH_Conf_COMx	Call	This call causes the message handler to send a message with the requested transmission rate of COMx and with M-sequence TYPE_0 (see Table 46).
MH_Conf_STARTUP	Call	This call causes the message handler to switch to the STARTUP state (see Figure 40)
MH_Conf_PREOPERATE	Call	This call causes the message handler to switch to the PREOPERATE state (see Figure 40)
MH_Conf_OPERATE	Call	This call causes the message handler to switch to the OPERATE state (see Figure 40)
xx_Conf_ACTIVE	Call	These calls activate the respective handler. xx is substitute for MH (message handler), OH (On-request Data handler), IH (ISDU handler), CH (Command handler), and/or EH (Event handler)
xx_Conf_INACTIVE	Call	These calls deactivate the respective handler. xx is substitute for MH (message handler), OH (On-request Data handler), IH (ISDU handler), CH (Command handler), and/or EH (Eventhandler)
Retry	Variable	Number of retries to establish communication

7.3.2.5 State machine of the Device DL-mode handler

Figure 37 shows the state machine of the Device DL-mode handler.

In state PreOperate_3 and Operate_4 different sets of handlers within the Device are activated.

The Master uses MasterCommands (see Table 44) to change the Device to SIO, STARTUP, PREOPERATE, and OPERATE states.

Whenever the message handler detects illegal (unexpected) M-sequence types, it will cause the DL-mode handler to change to the STARTUP state and to indicate this state to its system mangement (see 9.3.3.2) for the purpose of synchronization of Master and Device.

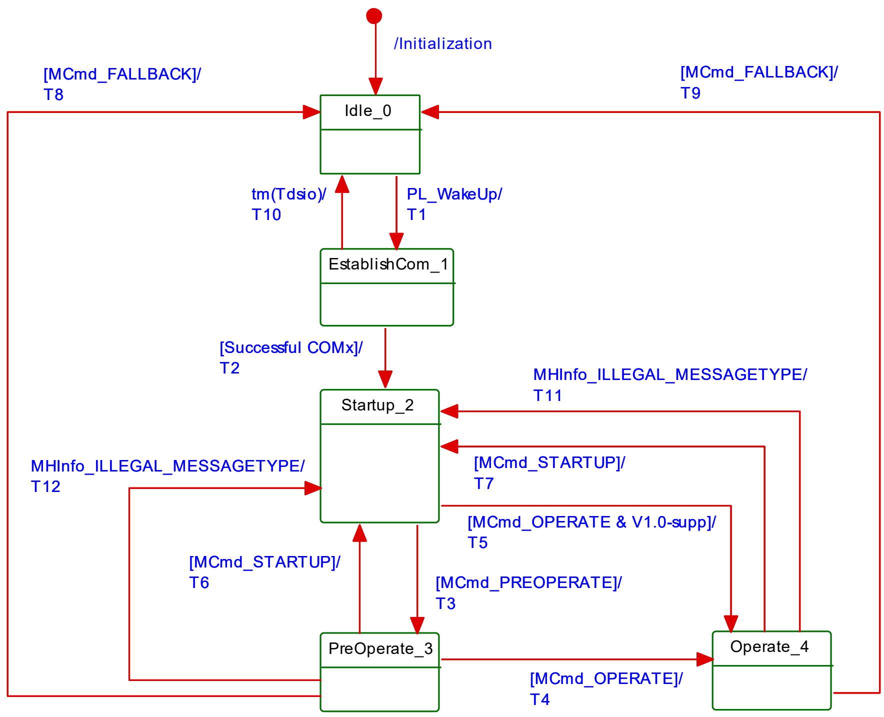


Figure 37 – State machine of the Device DL-mode handler

Table 45 shows the state transition tables of the Device DL-mode handler.

Table 45 – State transition tables of the Device DL-mode handler

STATE NAME	STATE DESCRIPTION
Idle_0	Waiting on a detected wakeup current pulse (PL_WakeUp.ind).
EstablishComm_1	Message handler activated and waiting for the COMx test messages (see Table 44)
Startup_2	Compatibility checks (see 9.2.3.3). Devices not supporting a Master according [8] will remain in STARTUP thus supporting further identification but no process data exchange in this case.
Preoperate_3	On-request Data exchange (parameter, commands, Events) without Process Data
Operate_4	Process Data (PD) and On-request Data exchange (parameter, commands, Events)

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	Wakeup current pulse detected. Activate message handler (call MH_Conf_ACTIVE in Figure 44). Indicate state via service DL_Mode.ind (ESTABCOM) to SM.
T2	1	2	One out of the three transmission rates of COM3, COM2, or COM1 mode established. Activate On-request Data (call OH_Conf_ACTIVE in Figure 49) and command handler (call CH_Conf_ACTIVE in Figure 54). Indicate state via service DL_Mode.ind (COM1, COM2, or COM3) to SM.
T3	2	3	Device command handler received MasterCommand (MCmd_PREOPERATE). Activate ISDU (call IH_Conf_ACTIVE in Figure 52) and Event handler (call EH_Conf_ACTIVE in Figure 56). Indicate state via service DL_Mode.ind (PREOPERATE) to SM.
T4	3	4	Device command handler received MasterCommand (MCmd_OPERATE). Activate Process Data handler (call PD_Conf_ACTIVE in Figure 47). Indicate state via service DL_Mode.ind (OPERATE) to SM.
T5	2	4	Device command handler received MasterCommand (MCmd_OPERATE). Activate Process Data handler (call PD_Conf_ACTIVE in Figure 47), ISDU (call IH_Conf_ACTIVE in Figure 52), and Event handler (call EH_Conf_ACTIVE in Figure 56). Indicate state via service DL_Mode.ind (OPERATE) to SM.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T6	3	2	Device command handler received MasterCommand (MCmd_STARTUP). Deactivate ISDU (call IH_Conf_INACTIVE in Figure 52) and Event handler (call EH_Conf_INACTIVE in Figure 56). Indicate state via service DL_Mode.ind (STARTUP) to SM.
T7	4	2	Device command handler received MasterCommand (MCmd_STARTUP). Deactivate Process Data handler (call PD_Conf_INACTIVE in Figure 47), ISDU (call IH_Conf_INACTIVE in Figure 52), and Event handler (call EH_Conf_INACTIVE in Figure 56). Indicate state via service DL_Mode.ind (STARTUP) to SM.
T8	3	0	Device command handler received MasterCommand (MCmd_FALLBACK). Wait until T_{FBD} elapsed, and then deactivate all handlers (call xx_Conf_INACTIVE). Indicate state via service DL_Mode.ind (INACTIVE) to SM (see Figure 81 and Table 95).
T9	4	0	Device command handler received MasterCommand (MCmd_FALLBACK). Wait until T_{FBD} elapsed, and then deactivate all handlers (call xx_Conf_INACTIVE). Indicate state via service DL_Mode.ind (INACTIVE) to SM (see Figure 81 and Table 95).
T10	1	0	After unsuccessful wakeup procedures (see Figure 32) the Device establishes the configured SIO mode after an elapsed time T_{DSIO} (see Figure 33). Deactivate all handlers (call xx_Conf_INACTIVE). Indicate state via service DL_Mode.ind (INACTIVE) to SM.
T11	4	2	Message handler detected an illegal M-sequence type. Deactivate Process Data (call PD_Conf_INACTIVE in Figure 47), ISDU (call IH_Conf_INACTIVE in Figure 52), and Event handler (call EH_Conf_INACTIVE in Figure 56). Indicate state via service DL_Mode.ind (STARTUP) to SM (see Figure 81 and Table 95).
T12	3	2	Message handler detected an illegal M-sequence type. Deactivate ISDU (call IH_Conf_INACTIVE in Figure 52) and Event handler (call EH_Conf_INACTIVE in Figure 56). Indicate state via service DL_Mode.ind (STARTUP) to SM (see Figure 81 and Table 95).

INTERNAL ITEMS	TYPE	DEFINITION
T_{FBD}	Time	See Table 43
T_{DSIO}	Time	See Figure 33
MCmd_XXXXXXX	Call	Any MasterCommand received by the Device command handler (see Table 44 and Figure 54, state "CommandHandler_2")
V1.0-supp	Flag	Device supports V1.0 mode

7.3.3 Message handler

7.3.3.1 General

The role of the message handler is specified in 7.1 and 7.2.2.1. This subclause specifies the structure and types of M-sequences and the behaviour (dynamics) of the message handler.

7.3.3.2 M-sequences

A Master and its Device exchange data by means of a sequence of messages (M-sequence). An M-sequence comprises a message from the Master followed by a message from the Device as shown in Figure 38. Each message consists of UART frames.

All the multi-octet data types shall be transmitted as a big-endian sequence, i.e. the most significant octet (MSO) shall be sent first, followed by less significant octets in descending order, with the least significant octet (LSO) being sent last, as shown in Figure 2.

The Master message starts with the "M-sequence Control" (MC) octet, followed by the "CHECK/TYPE" (CKT) octet, and optionally followed by either "Process Data" (PD) and/or "On-request Data" (OD) octets. The Device message in turn starts optionally with "Process Data" (PD) octets and/or "On-request Data" (OD) octets, followed by the "CHECK/STAT" (CKS) octet.

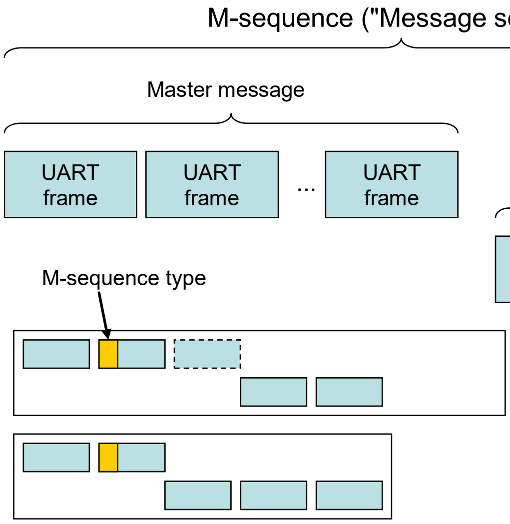


Figure 38 – SDCI message sequences

Various M-sequence types can be selected to meet the particular needs of an actuator or sensor (scan rate, amount of Process Data). The length of Master and Device messages may vary depending on the type of messages and the data transmission direction, see Figure 38.

Figure 39 presents an overview of the defined M-sequence types. Parts within dotted lines depend on the read or write direction within the M-sequence control octet.

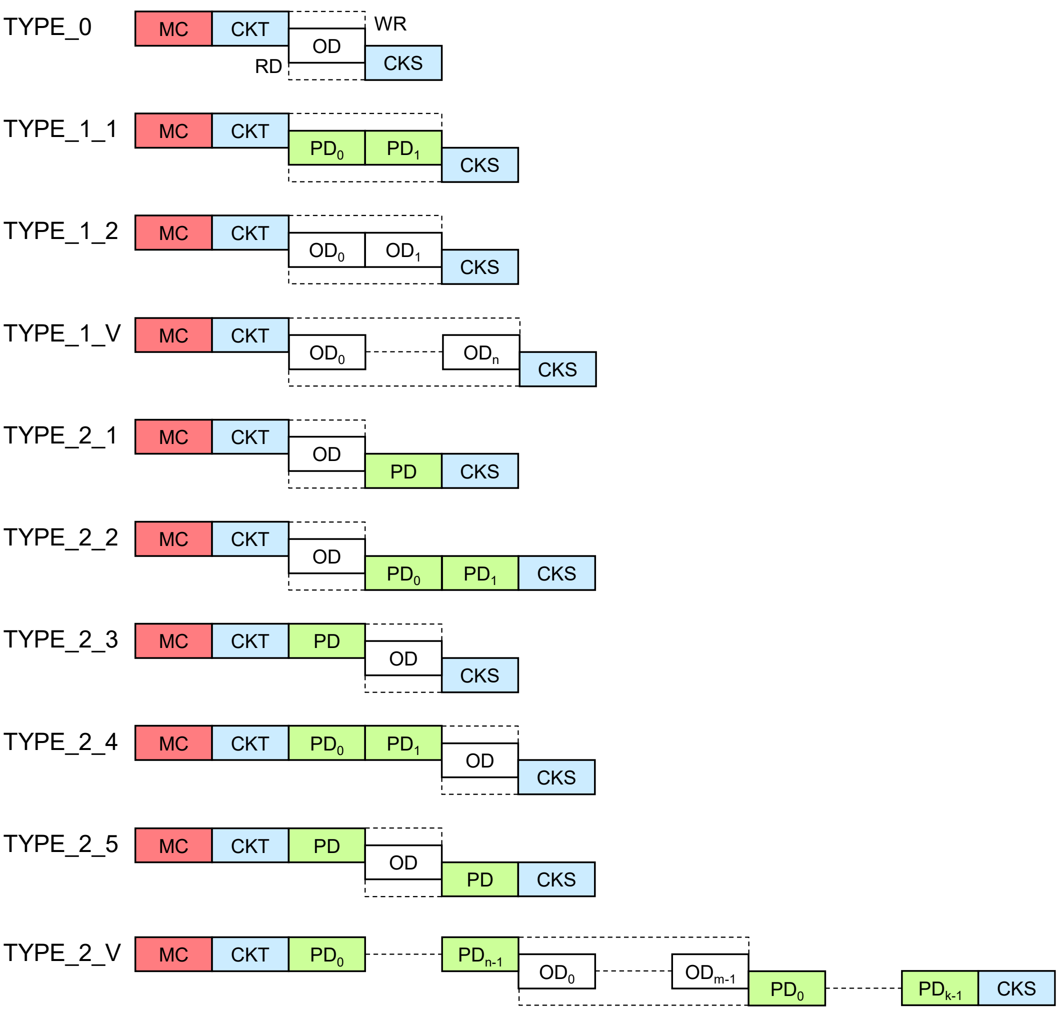


Figure 39 – Overview of M-sequence types

The fixed M-sequence types consist of TYPE_0, TYPE_1_1, TYPE_1_2, and TYPE_2_1 through TYPE_2_5. Caution: The former TYPE_2_6 is no more supported. The variable M-sequence types consist of TYPE_1_V and TYPE_2_V.

The different M-sequence types meet the various requirements of sensors and actuators regarding their Process Data width and respective conditions. See A.2 for details of M-sequence types. See A.3 for the timing constraints with M-sequences.

7.3.3.3 MasterCycleTime constraints

Within state STARTUP and PREOPERATE a Device is able to communicate in an acyclic manner. In order to detect the disconnecting of Devices it is highly recommended for the Master to perform from this point on a periodic communication ("keep-alive message") via acyclic M-sequences through the data link layer. The minimum recovery times for acyclic communication specified in A.2.6 shall be considered.

After these phases, cyclic Process Data communication can be started by the Master via the DL_SetMode (OPERATE) service. M-sequence types for the cyclic data exchange shall be used in this communication phase to exchange Process Data (PD) and On-request Data with a Device (see Table A.9 and Table A.10).

The Master shall use for time t_{CYC} the value indicated in the Device parameter "MasterCycleTime" (see Table B.1) with a relative tolerance of -1 % to +10 % (including jitter).

In cases, where a Device has to be switched back to SIO mode after parameterization, the Master shall send a command "Fallback" (see Table B.2), which is followed by a confirmation from the Device.

7.3.3.4 State machine of the Master message handler

Figure 40 shows the Master state machine of the Master message handler. Three submachines describing reactions on communication errors are shown in Figure 41, Figure 42, and Figure 43.

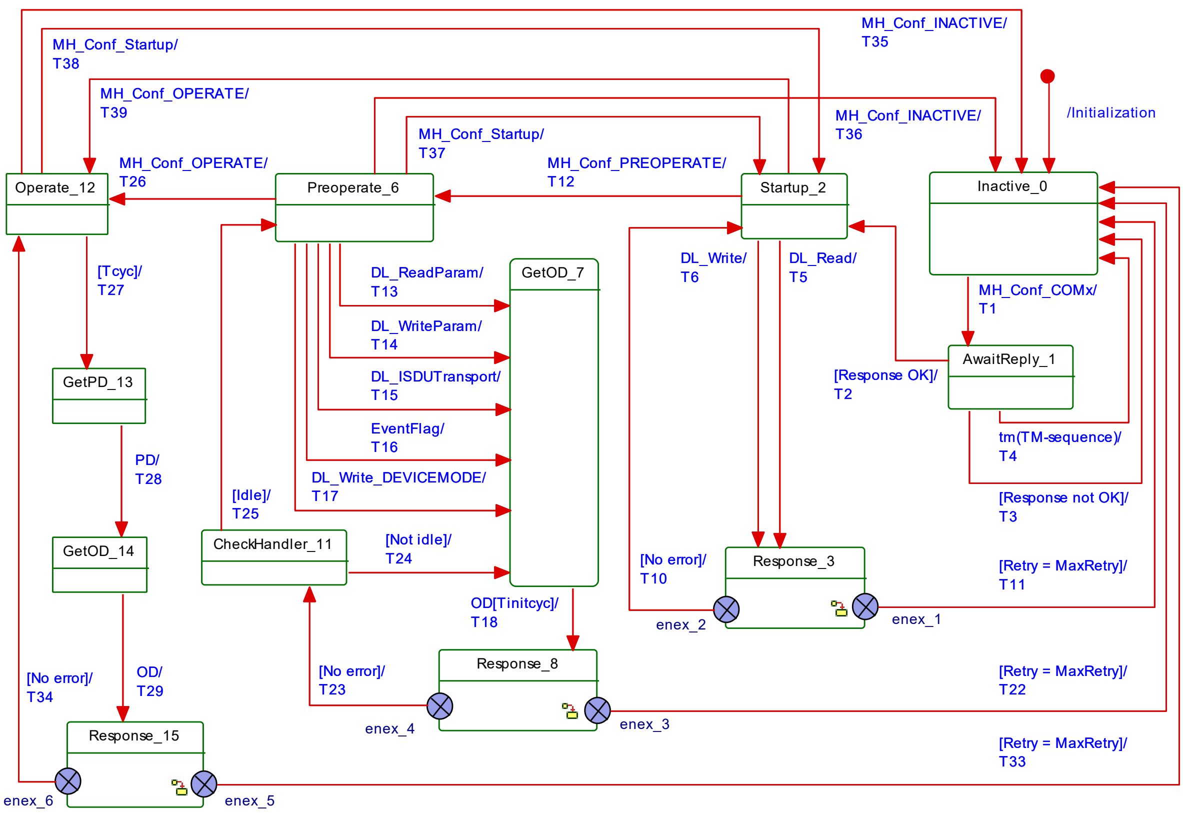


Figure 40 – State machine of the Master message handler

The message handler takes care of the special communication requirements within the states "EstablishCom", "Startup", "PreOperate", and "Operate" of the DL-Mode handler. An internal administrative call MH_Conf_COMx in state "Inactive_0" causes the message handler to send "test" messages with M-sequence TYPE_0 and different transmission rates of COM3, COM2, or COM1 during the establish communication sequence.

The state "Startup_2" provides all the communication means to support the identity checks of System Management with the help of DL_Read and DL_Write services. The message handler waits on the occurrence of these services to send and receive messages (acyclic communication). The state "Preoperate_6" is the checkpoint for all On-request Data activities such as ISDUs, commands, and Events for parameterization of the Device. The message handler waits on the occurrence of the services shown in Figure 40 to send and receive messages (acyclic communication). The state "Operate_12" is the checkpoint for cyclic Process Data exchange. Depending on the M-sequence type the message handler generates Master messages with Process Data acquired from the Process Data handler via the PD service and optionally On-request Data acquired from the On-request Data handler via the OD service.

Figure 41 shows the submachine of state "Response 3".

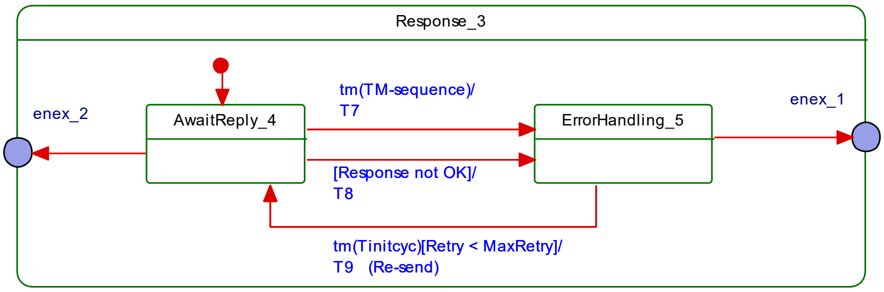


Figure 41 – Submachine "Response 3" of the message handler

Figure 42 shows the submachine of state "Response 8".

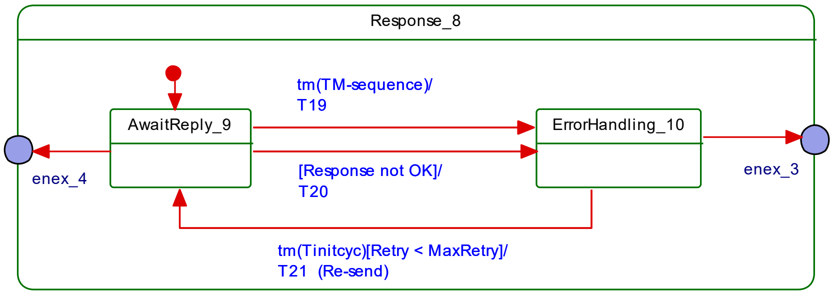


Figure 42 – Submachine "Response 8" of the message handler

Figure 43 shows the submachine of state "Response 15".

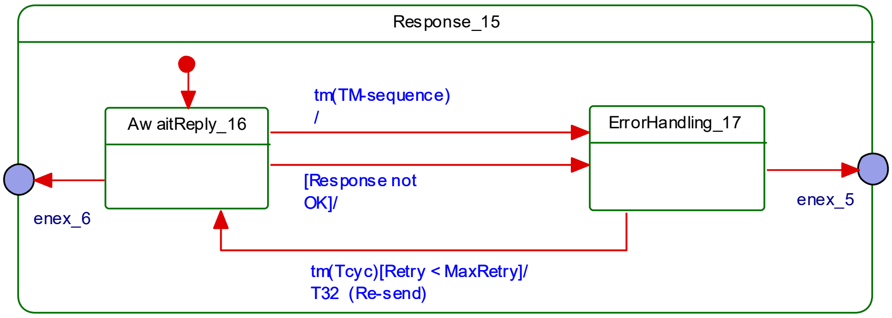


Figure 43 – Submachine "Response 15" of the message handler

Table 46 shows the state transition tables of the Master message handler.

Table 46 – State transition table of the Master message handler

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting on demand for a "test" message via MH_Conf_COMx call (see Figure 36 and Table 44) from DL-mode handler.
AwaitReply_1	Waiting on response from the Device to the "test" message. Return to Inactive_0 state whenever the time $T_{\text{M-sequence}}$ elapsed without response from the Device or the response to the "test" message could not be decoded. In case of a correct response from the Device, the message handler changes to the Startup_2 state.
Startup_2	When entered via transition T2, this state is responsible to control acyclic On-request Data exchange according to conditions specified in Table A.7. Any service DL_Write or DL_Read from System Management causes a transition.
Response_3	The OD service caused the message handler to send a corresponding message. The submachine in this pseudo state waits on the response and checks its correctness.

STATE NAME	STATE DESCRIPTION
SM: AwaitReply_4	This state checks whether the time $T_{\text{M-sequence}}$ elapsed and the response is correct.
SM: ErrorHandling_5	In case of an incorrect response the message handler will re-send the message after a waiting time T_{initcyc}. After too many retries the message handler will change to the Inactive_0 state.
Preoperate_6	Upon reception of a call MH_Conf_PREOPERATE the message handler changed to this state. The message handler is now responsible to control acyclic On-request Data exchange according to conditions specified in Table A.8. Any service DL_ReadParam, DL_WriteParam, DL_ISDUTransport, DL_Write, or EventFlag causes a transition.
GetOD_7	The message handler used the ODTrig service to aquire OD from the On-request Data handler. The message handler waits on the OD service to send a message after a time T_{initcyc}.
Response_8	The OD service caused the message handler to send a corresponding message. The submachine in this pseudo state waits on the response and checks its correctness.
SM: AwaitReply_9	This state checks whether the time $T_{\text{M-sequence}}$ elapsed and the response is correct.
SM: ErrorHandling_10	In case of an incorrect response the message handler will re-send the message after a waiting time T_{initcyc}. After too many retries the message handler will change to the Inactive_0 state.
CheckHandler_11	Some services require several OD acquisition cycles to exchange the OD. Whenever the affected OD, ISDU, or Event handler returned to the idle state, the message handler can leave the OD acquisition loop.
Operate_12	Upon reception of a call MH_Conf_OPERATE the message handler changed to this state and after an initial time T_{initcyc}, it is responsible to control cyclic Process Data and On-request Data exchange according to conditions specified in Table A.9 and Table A.10. The message handler restarts on its own a new message cycle after the time t_{CYC} elapsed.
GetPD_13	The message handler used the PDTrig service to aquire PD from the Process Data handler. The message handler waits on the PD service and then changes to state GetOD_14.
GetOD_14	The message handler used the ODTrig service to aquire OD from the On-request Data handler. The message handler waits on the OD service to complement the already acquired PD and to send a message with the acquired PD/OD.
Response_15	The message handler sent a message with the acquired PD/OD. The submachine in this pseudo state waits on the response and checks its correctness.
SM: AwaitReply_16	This state checks whether the time $T_{\text{M-sequence}}$ elapsed and the response is correct.
SM: ErrorHandling_17	In case of an incorrect response the message handler will re-send the message after a waiting time t_{CYC}. After too many retries the message handler will change to the Inactive_0 state.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	Send a message with the requested transmission rate of COMx and with M-sequence TYPE_0: Read Direct Parameter page 1, address 0x02 ("MinCycleTime"), compiling into an M-sequence control MC = 0xA2 (see A.1.2). Start timer with $T_{\text{M-sequence}}$.
T2	1	2	Return value of "MinCycleTime" via DL_Read service confirmation.
T3	1	0	Reset timer ($T_{\text{M-sequence}}$).
T4	1	0	Reset timer ($T_{\text{M-sequence}}$).
T5	2	3	Send message using the established transmission rate, the page communication channel, and the read access option (see A.1.2). Start timer with $T_{\text{M-sequence}}$.
T6	2	3	Send message using the established transmission rate, the page communication channel, and the write access option (see A.1.2). Start timer with $T_{\text{M-sequence}}$.
T7	4	5	Reset timer ($T_{\text{M-sequence}}$).
T8	4	5	Reset timer ($T_{\text{M-sequence}}$).
T9	5	4	Re-send message after a time T_{initcyc}. Restart timer with $T_{\text{M-sequence}}$.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T10	3	2	Return DL_Read or DL_Write service confirmation respectively to System Management.
T11	3	0	Message handler returns MH_Info (COMLOST) to DL-mode handler.
T12	2	6	-
T13	6	7	The Message handler invokes the ODTrig service for the On-request handler (see Figure 48), which is in state "ISDU_1". In this state it causes the ISDU handler to provide the OD service in correspondence to the DL_ReadParam service (see Figure 51, Transition T13).
T14	6	7	The Message handler invokes the ODTrig service for the On-request handler (see Figure 48), which is in state "ISDU_1". In this state it causes the ISDU handler to provide the OD service in correspondence to the DL_WriteParam service (see Figure 51, Transition T13).
T15	6	7	The Message handler invokes the ODTrig service for the On-request handler (see Figure 48), which is in state "ISDU_1". In this state it causes the ISDU handler to provide the OD service in correspondence to the DL_ISDUTransort service (see Figure 51, Transition T2). The message handler may need several cycles until the ISDU handler returns to the "idle" state.
T16	6	7	The Message handler invokes the ODTrig service for the On-request handler (see Figure 48), which is in state "Event_3". In this state it causes the Event handler to provide the OD service in correspondence to the EventFlag service (see Figure 55, Transition T2). The message handler may need several cycles until the Event handler returns to the "idle" state.
T17	6	7	The Message handler invokes the ODTrig service for the On-request handler (see Figure 48), which is in state "ISDU_1". In this state it causes the ISDU handler to provide the OD service in correspondence to the DL_Write service (see Figure 51, Transition T13).
T18	7	8	Send message after a recovery time T_{initcyc} caused by the OD.req service. Start timer with $T_{\text{M-sequence}}$.
T19	9	10	Reset timer ($T_{\text{M-sequence}}$).
T20	9	10	Reset timer ($T_{\text{M-sequence}}$).
T21	10	9	Re-send message after a time T_{initcyc}. Restart timer with $T_{\text{M-sequence}}$.
T22	8	0	Message handler changes to state Inactive_0 and returns MH_Info (COMLOST) to DL-mode handler.
T23	8	11	-
T24	11	7	Acquire OD through invocation of the ODTrig service to the On-request Data handler, which in turn triggers the current handler in charge via the ISDU or EventTrig call.
T25	11	6	Return result via service primitive OD.cnf
T26	6	12	Message handler changes to state Operate_12.
T27	12	13	Start the t_{CYC} -timer. Acquire PD through invocation of the PDTrig service to the Process Data handler (see Figure 46).
T28	13	14	Acquire OD through invocation of the ODTrig service to the On-request Data handler (see Figure 48).
T29	14	15	PD and OD ready through PD.req service from PD handler and OD.req service via the OD handler. Message handler sends message. Start timer with $T_{\text{M-sequence}}$.
T30	16	17	Reset timer ($T_{\text{M-sequence}}$).
T31	16	17	Reset timer ($T_{\text{M-sequence}}$).
T32	17	16	Re-send message after a time t_{CYC}. Restart timer with $T_{\text{M-sequence}}$.
T33	15	0	Message handler changes to state Inactive_0 and returns MH_Info (COMLOST) to DL-mode handler.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T34	15	12	Device response message is correct. Return PD via service PD.cnf and via call PDTrig to the PD handler (see Table 48). Return OD via service OD.cnf and via call ODTrig to the On-request Data hander, which redirects it to the ISDU (see Table 53), Command (see Table 56), or Event handler (see Table 59) in charge.
T35	12	0	Message handler changes to state Inactive_0 and returns MH_Info (COMLOST) to the DL-mode handler.
T36	6	0	Message handler changes to state Inactive_0 and returns MH_Info (COMLOST) to the DL-mode handler.
T37	6	2	-
T38	12	2	-
T39	2	12	-

INTERNAL ITEMS	TYPE	DEFINITION
Retry	Variable	Retry counter
MaxRetry	Constant	MaxRetry = 2, see Table 102
$t_{\text{M-sequence}}$	Time	See equation (A.6)
t_{CYC}	Time	The DL_SetMode service provides this value with its parameter "M-sequenceTime". See equation (A.7)
t_{initcyc}	Time	See A.2.6
MH_Conf_xxx	Call	See Table 44

7.3.3.5 State machine of the Device message handler

Figure 44 shows the state machine of the Device message handler.

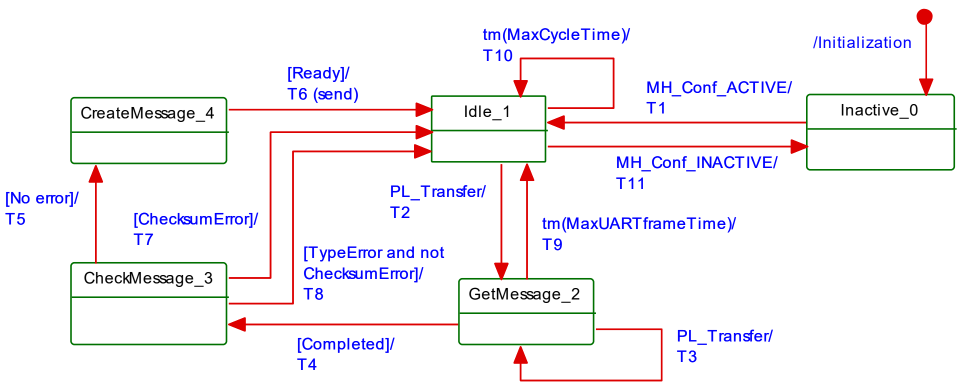


Figure 44 – State machine of the Device message handler

Table 47 shows the state transition tables of the Device message handler.

Table 47 – State transition tables of the Device message handler

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting for activation by the Device DL-mode handler through MH_Conf_ACTIVE (see Table 45, Transition T1).
Idle_1	Waiting on first UART frame of the Master message through PL_Transfer service indication. Check whether time "MaxCycleTime" elapsed.
GetMessage_2	Receive a Master message UART frame. Check number of received UART frames (Device detects M-sequence type by means of the first two received octets depending on the current communication state and thus knows the number of the UART frames). Check whether the time "MaxUARTframeTime" elapsed.
CheckMessage_3	Check M-sequence type and checksum of received message.
CreateMessage_4	Compile message from OD.rsp, PD.rsp, EventFlag, and PDStatus services.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	–
T2	1	2	Start "MaxUARTframeTime" and "MaxCycleTime" when in OPERATE.
T3	2	2	Restart timer "MaxUARTframeTime".
T4	2	3	Reset timer "MaxUARTframeTime".
T5	3	4	Invoke OD.ind and PD.ind service indications
T6	4	1	Compile and invoke PL_Transfer.rsp service response (Device sends response message)
T7	3	1	–
T8	3	1	Indicate error to DL-mode handler via MHInfo (ILLEGAL_MESSAGETYPE)
T9	2	1	Reset both timers "MaxUARTframeTime" and "MaxCycleTime".
T10	1	1	Indicate error to actuator technology that shall observe this information and take corresponding actions (see 10.2 and 10.8.3).
T11	1	0	Device message handler changes state to Inactive_0.

INTERNAL ITEMS	TYPE	DEFINITION
MaxUARTFrameTime	Time	Time for the transmission of a UART frame (11 T_{BIT}) plus maximum of t_1 (1 T_{BIT}) = 12 T_{BIT}.
MaxCycleTime	Time	The purpose of the timer "MaxCycleTime" is to check, whether cyclic Process Data exchange took too much time or has been interrupted. (see A.3.7). See NOTE for implementation hint.
TypeError	Guard	One of the possible errors detected: ILLEGAL_MESSAGETYPE, or COMLOST
ChecksumError	Guard	Checksum error of message detected

NOTE: To achieve the expected failure reaction, the loss of communication check should be placed in Figure 47 with a timeout supervision, respecting all possible retries, relevant errors and MasterCycleTime. Upcoming specifications will define this type of detection.

7.3.4 Process Data handler

7.3.4.1 General

The transport of output Process Data is performed using the DL_OutputUpdate services and for input Process Data using the DL_InputTransport services (see Figure 28). A Process Data cycle is completed when the entire set of Process Data has been transferred between Master and Device in the requested direction. Such a cycle can last for more than one M-sequence.

All Process Data are transmitted within one M-sequence when using M-sequences of TYPE_2_x (see Figure 39). In this case the execution time of a Process Data cycle is equal to the cycle time t_{CYC}.

7.3.4.2 Interleave mode

All Process Data and On-request Data are transmitted in this case with multiple alternating M-sequences TYPE_1_1 (Process Data) and TYPE_1_2 (On-request Data) as shown in Figure 45. It demonstrates the Master messages writing output Process Data to a Device. The service parameter PDOutAddress indicates the partition of the output PD to be transmitted (see 7.2.2.3). For input Process Data the service parameter PDInAddress correspondingly indicates the partition of the input PD. Within a Process Data cycle all input PD shall be read first followed by all output PD to be written. A Process Data cycle comprises all cycle times required to transmit the complete Process Data.

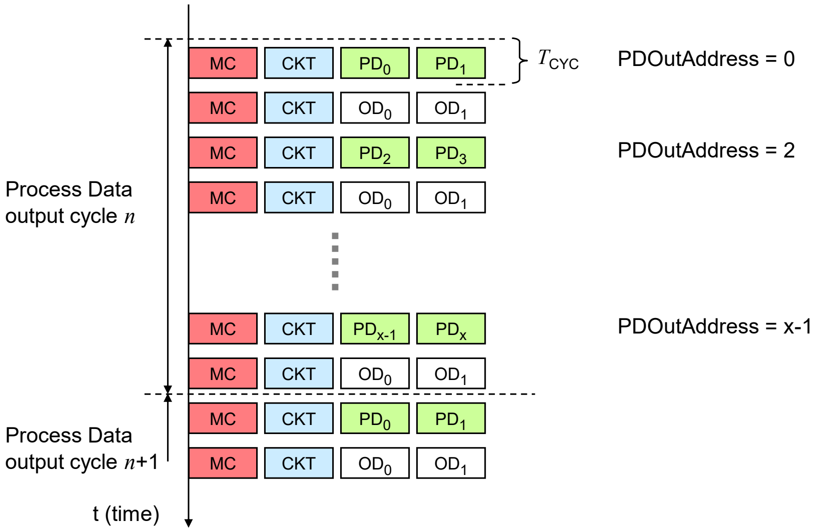


Figure 45 – Interleave mode for the segmented transmission of Process Data

Interleave mode is for legacy Devices only.

7.3.4.3 State machine of the Master Process Data handler

Figure 46 shows the state machine of the Master Process Data handler.

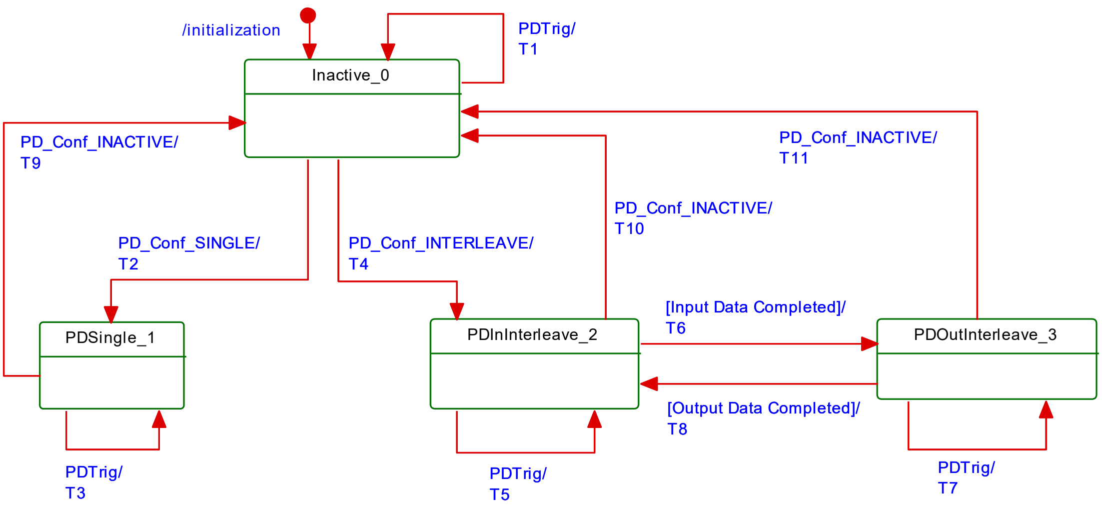


Figure 46 – State machine of the Master Process Data handler

Table 48 shows the state transition tables of the Master Process Data handler.

Table 48 – State transition tables of the Master Process Data handler

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting for activation
PDSingle_1	Process Data communication within one single M-sequence
PDInInterleave_2	Input Process Data communication in interleave mode
PDOutInterleave_3	Output Process Data communication in interleave mode

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	0	Invoke PD.req with no Process Data
T2	0	1	NOTE The DL-mode handler configured the Process Data handler for single PD transmission (see Table 44, T10 or T11).
T3	1	1	Take data from DL_PDOutputUpdate service and invoke PD.req to propagate output PD to the message handler. Take data from PD.cnf and invoke DL_PDInputTransport.ind and DL_PDCycle.ind to propagate input PD to the AL.
T4	0	2	NOTE Configured for interleave PD transmission (see Table 44, T10 or T11).
T5	2	2	Invoke PD.req and use PD.cnf to prepare DL_PDInputTransport.ind.
T6	2	3	Invoke DL_PDInputTransport.ind and DL_PDCycle.ind to propagate input PD to the AL (see 7.2.1.11).
T7	3	3	Take data from DL_PDOutputUpdate service and invoke PD.req to propagate output PD to the message handler.
T8	3	2	Invoke DL_PDCycle.ind to indicate end of Process Data cycle to the AL (see 7.2.1.12).
T9	1	0	-
T10	2	0	-
T11	3	0	-

INTERNAL ITEMS	TYPE	DEFINITION
<None>		

7.3.4.4 State machine of the Device Process Data handler

Figure 47 shows the state machine of the Device Process Data handler.

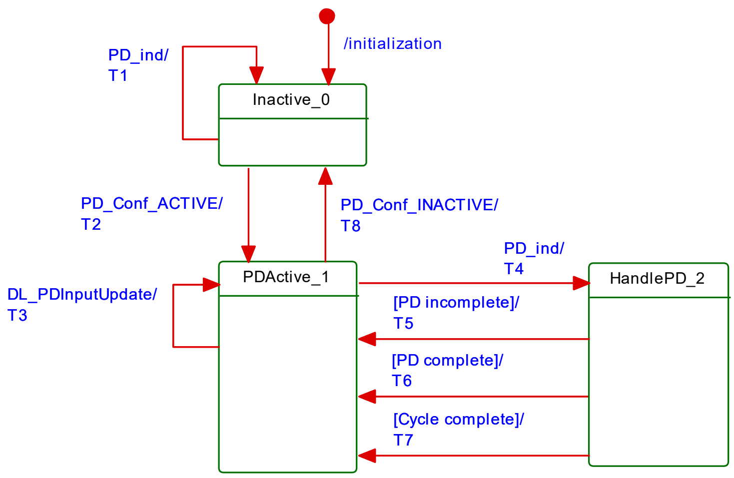


Figure 47 – State machine of the Device Process Data handler

See sequence diagrams in Figure 67 and

Figure 68 for context.

Table 49 shows the state transition tables of the Device Process Data handler

Table 49 – State transition tables of the Device Process Data handler

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting on activation
PDActive_1	Handler active and waiting on next message handler demand via PD service or DL_PDInputUpdate service from AL.
HandlePD_2	Check Process Data for completeness in interleave mode

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	0	Ignore Process Data
T2	0	1	-
T3	1	1	Prepare input Process Data for PD.rsp for next message handler demand
T4	1	2	Message handler demands input PD via a PD.ind service and delivers output PD or segment of output PD. Invoke PD.rsp with input Process Data when in non-interleave mode (see 7.2.2.3).
T5	2	1	-
T6	2	1	Invoke DL_PDOutputTransport.ind (see 7.2.1.9)
T7	2	1	Invoke DL_PDCycle.ind (see 7.2.1.12)
T8	1	0	-

INTERNAL ITEMS	TYPE	DEFINITION
PD_ind	Label	Invocation of service PD.ind occurred from message handler

7.3.5 On-request Data handler

7.3.5.1 General

The Master On-request Data handler is a subordinate state machine active in the "Startup_2", "PreOperate_3", and "Operate_4" state of the DL-mode handler (see Figure 35). It controls three other state machines, the so-called ISDU handler, the command handler, and the Event handler. It always starts with the ISDU handler by default.

Whenever an EventFlag.ind is received, the state machine will change to the Event handler. After the complete readout of the Event information it will return to the ISDU handler state.

Whenever a DL_Control.req or PDInStatus.ind service is received while in the ISDU handler or in the Event handler, the state machine will change to the command handler. Once the command has been served, the state machine will return to the previously active state (ISDU or Event).

7.3.5.2 State machine of the Master On-request Data handler

Figure 48 shows the Master state machine of the On-request Data handler.

The On-request Data handler redirects the ODTrig.ind service primitive for the next message content to the currently active subsidiary handler (ISDU, command, or Event). This is performed through one of the ISDUTrig, CommandTrig, or EventTrig calls.

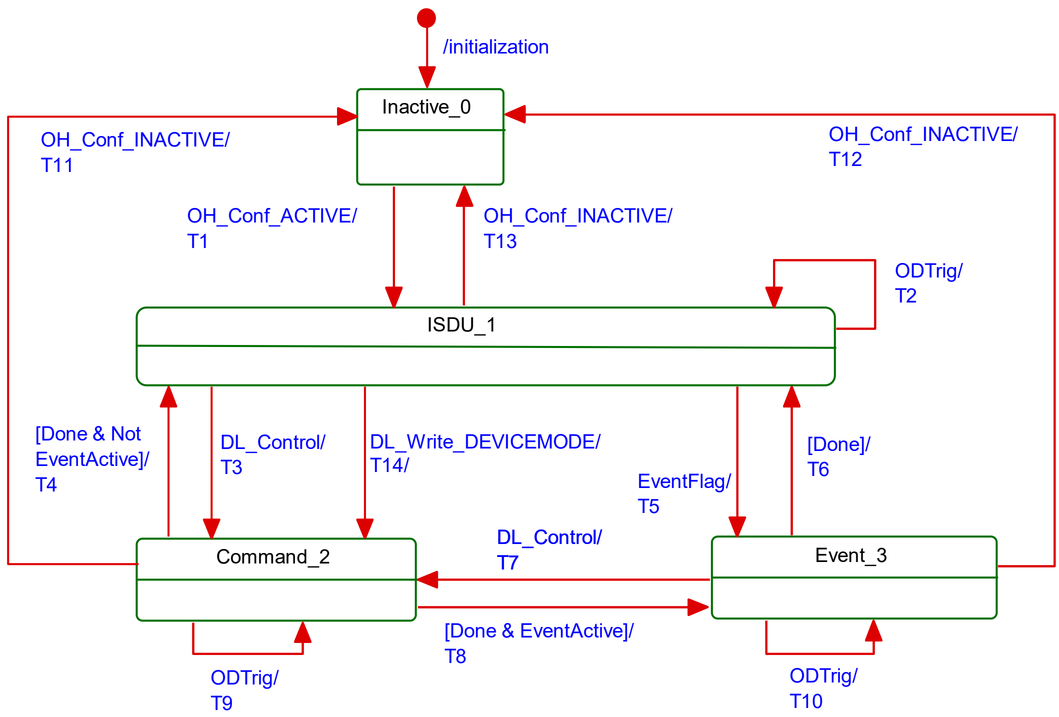


Figure 48 – State machine of the Master On-request Data handler

Table 50 shows the state transition tables of the Master On-request Data handler.

Table 50 – State transition tables of the Master On-request Data handler

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting on activation
ISDU_1	Default state of the On-request Data handler (lowest priority)
Command_2	State to control the Device via commands with highest priority
Event_3	State to convey Event information (errors, warnings, notifications) with higher priority

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	1	1	On-request Data handler propagates the ODTrig.ind service now named ISDUTrig to the ISDU handler (see Figure 51). In case of DL_Read, DL_Write, DL_ReadParam, or DL_WriteParam services, the ISDU handler will use a separate transition (see Figure 51, T13).
T3	1	2	-
T4	2	1	-
T5	1	3	EventActive = TRUE
T6	3	1	EventActive = FALSE
T7	3	2	-
T8	2	3	-
T9	2	2	On-request Data handler propagates the ODTrig.ind service now named CommandTrig to the command handler (see Figure 53)
T10	3	3	On-request Data handler propagates the ODTrig.ind service now named EventTrig to the Event handler (see Figure 55)
T11	2	0	-

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T12	3	0	-
T13	1	0	-
T14	1	2	-

INTERNAL ITEMS	TYPE	DEFINITION
EventActive	Bool	Flag to indicate return direction after interruption of Event processing by a high priority command request

7.3.5.3 State machine of the Device On-request Data handler

Figure 49 shows the state machine of the Device On-request Data handler.

The Device On-request Data handler obtains information on the communication channel and the parameter or FlowCTRL address via the OD.ind service. The communication channels are totally independent. In case of a valid access, the corresponding ISDU, command or Event state machine is addressed via the associated communication channel.

The Device shall respond to read requests to not implemented address ranges with the value "0". It shall ignore write requests to not implemented address ranges.

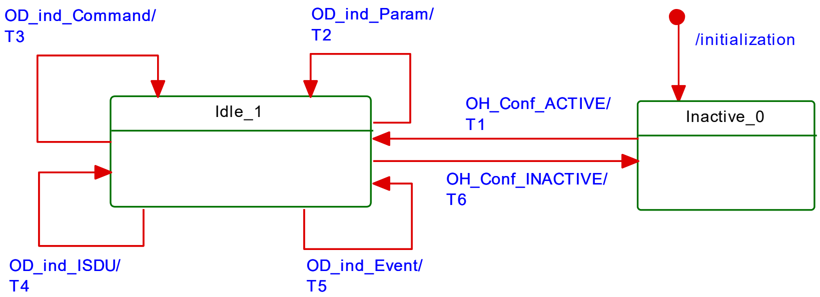


Figure 49 – State machine of the Device On-request Data handler

In case of an ISDU access in a Device without ISDU support, the Device shall respond with "No Service" (see Table A.12). An error message is not created.

NOTE OD.ind (R, ISDU, FlowCTRL = IDLE) is the default message if there are no On-request Data pending for transmission.

Table 51 shows the state transition tables of the Device On-request Data handler.

Table 51 – State transition tables of the Device On-request Data handler

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting on activation
Idle_1	Waiting on messages with On-request Data via service OD indication. Decomposition and analysis.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	1	1	Provide data content of requested parameter or perform appropriate write action
T3	1	1	Redirect to command handler
T4	1	1	Redirect to ISDU handler
T5	1	1	Redirect to Event handler
T6	1	0	-

INTERNAL ITEMS	TYPE	DEFINITION
OD_ind_Param	Service	Alias for Service OD.ind (R/W, PAGE, 1 to 31, Data) in case of DL_ReadParam or DL_WriteParam
OD_ind_Command	Service	Alias for Service OD.ind (W, PAGE, 0, MasterCommand)
OD_ind_ISDU	Service	Alias for Service OD.ind (R/W, ISDU, FlowCtrl, Data)
OD_ind_Event	Service	Alias for Service OD.ind (R/W, DIAGNOSIS, n, Data)

7.3.6 ISDU handler

7.3.6.1 Indexed Service Data Unit (ISDU)

The general structure of an ISDU is demonstrated in Figure 50 and specified in detail in Clause A.5.

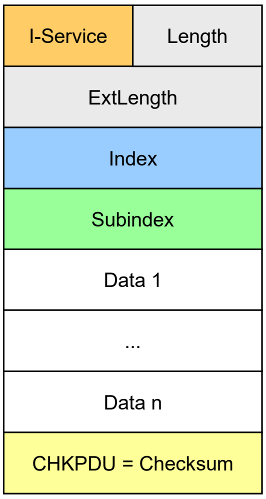


Figure 50 – Structure of the ISDU

The sequence of the elements corresponds to the transmission sequence. The elements of an ISDU can take various forms depending on the type of I-Service (see A.5.2 and Table A.12).

The ISDU allows accessing data objects (parameters and commands) to be transmitted (see Figure 6). The data objects shall be addressed by the “Index” element.

All multi-octet data types shall be transmitted as a big-endian sequence, i.e. the most significant octet (MSO) shall be sent first, followed by less significant octets in descending order, with the least significant octet (LSO) being sent last, as shown in Figure 2.

7.3.6.2 Transmission of ISDUs

An ISDU is transmitted via the ISDU communication channel (see Figure 8 and A.1.2). A number of messages are typically required to perform this transmission (segmentation). The Master transfers an ISDU by sending an I-Service (Read/Write) request to the Device via the ISDU communication channel. It then receives the Device’s response via the same channel.

In the ISDU communication channel, the "Address" element within the M-sequence control octet accommodates a counter (= FlowCTRL). FlowCTRL is controlling the segmented data flow (see A.1.2) by counting the M-sequences necessary to transmit an ISDU.

The receiver of an ISDU expects a FlowCTRL + 1 in the next message in case of undisturbed communication. If FlowCTRL is unchanged, the previously transmitted message is repeated. In any other case the ISDU structure is violated.

The Master uses the "Length" element of the ISDU and FlowCTRL to check the accomplishment of the complete transmission.

Permissible values for FlowCTRL are specified in Table 52.

Table 52 – FlowCTRL definitions

FlowCTRL	Definition
0x00 to 0x0F	COUNT M-sequence counter within an ISDU. Increments beginning with 1 after an ISDU START. Jumps back from 15 to 0 in the Event of an overflow.
0x10	START Start of an ISDU I-Service, 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 (see Table A.12).
0x11	IDLE 1 No request for ISDU transmission.
0x12	IDLE 2: Reserved for future use No request for ISDU transmission.
0x13 to 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.

In state Idle_1, values 0x12 to 0x1F shall not lead to a communication error.

7.3.6.3 State machine of the Master ISDU handler

Figure 51 shows the state machine of the Master ISDU handler.

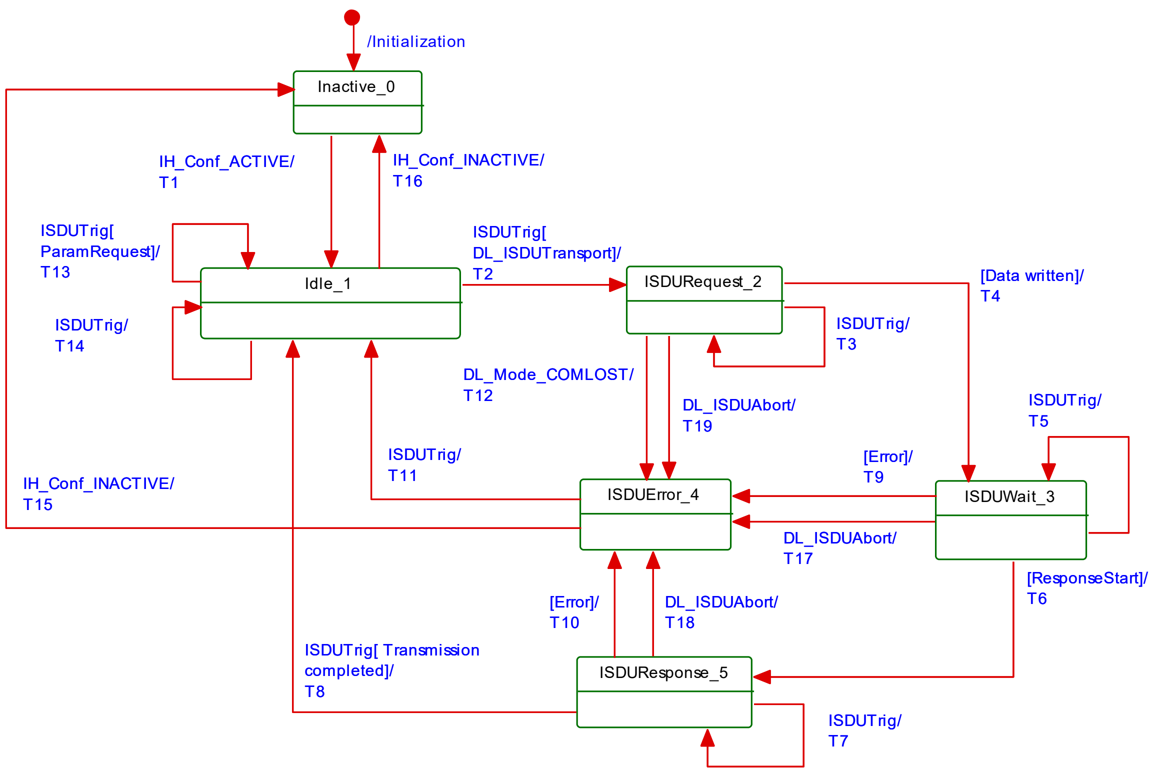


Figure 51 – State machine of the Master ISDU handler

1718 Table 53 shows the state transition tables of the Master ISDU handler.

1719 **Table 53 – State transition tables of the Master ISDU handler**

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting on activation
Idle_1	Waiting on transmission of next On-request Data
ISDURequest_2	Transmission of ISDU request data
ISDUWait_3	Waiting on response from Device. Observe ISDUTime
ISDUError_4	Error handling after detected errors: Invoke negative DL_ISDU_Transport response with ISDUTransportErrorInfo
ISDUResponse_5	Get response data from Device

1720

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	1	2	Invoke OD.req with ISDU write start condition: OD.req (W, ISDU, flowCtrl = START, data)
T3	2	2	Invoke OD.req with ISDU data write and FlowCTRL under conditions of Table 52
T4	2	3	Start timer (ISDUTime)
T5	3	3	Invoke OD.req with ISDU read start condition: OD.req (R, ISDU, flowCtrl = START)
T6	3	5	Stop timer (ISDUTime)
T7	5	5	Invoke OD.req with ISDU data read and FlowCTRL under conditions of Table 52
T8	5	1	OD.req (R, ISDU, flowCtrl = IDLE) Invoke positive DL_ISDUTransport confirmation
T9	3	4	-
T10	5	4	-
T11	4	1	Invoke OD.req with ISDU abortion: OD.req (R, ISDU, flowCtrl = ABORT). Invoke negative DL_ISDUTransport confirmation
T12	2	4	-
T13	1	1	Invoke OD.req with appropriate data. Invoke positive DL_ReadParam/DL_WriteParam confirmation
T14	1	1	Invoke OD.req with idle message: OD.req (R, ISDU, flowCtrl = IDLE)
T15	4	1	In case of lost communication, the message handler informs the DL_Mode handler which in turn uses the administrative call IH_Conf_INACTIVE. No actions during this transition required.
T16	1	0	-
T17	3	4	-
T18	5	4	-
T19	2	4	-

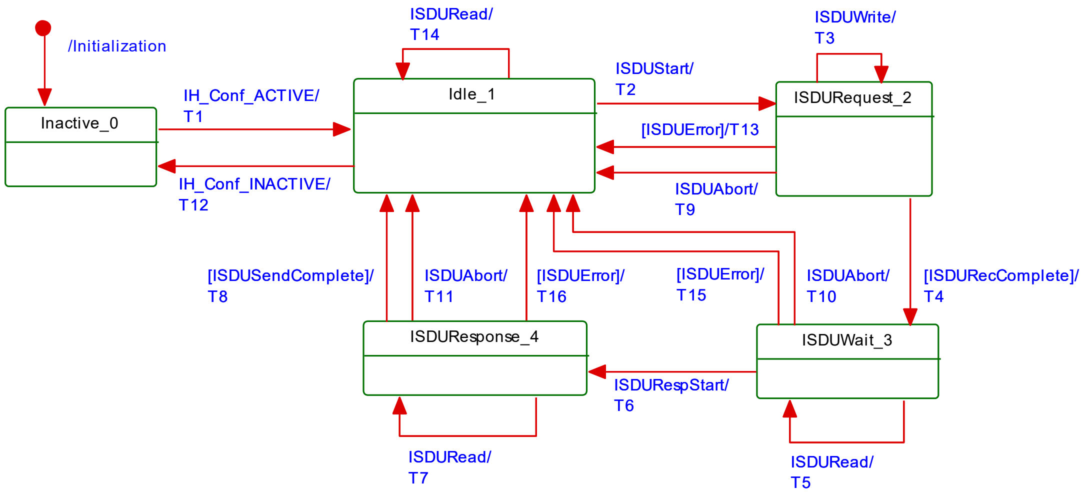
1721

INTERNAL ITEMS	TYPE	DEFINITION
ISDUTime	Time	Measurement of Device response time (watchdog, see Table 102)
ResponseStart	Service	OD.cnf without "busy" indication (see Table A.14)
ParamRequest	Service	DL_ReadParam or DL_WriteParam
Error	Variable	Any detectable error within the ISDU transmission or DL_ISDUAbort requests, or any violation of the ISDU acknowledgment time (see Table 102)

1722

1723 ### 7.3.6.4 State machine of the Device ISDU handler

1724 Figure 52 shows the state machine of the Device ISDU handler.



1725

1726 **Figure 52 – State machine of the Device ISDU handler**

1727 Table 54 shows the state transition tables of the Device ISDU handler.

1728 **Table 54 – State transition tables of the Device ISDU handler**

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting on activation
Idle_1	Waiting on next ISDU transmission
ISDURequest_2	Reception of ISDU request
ISDUWait_3	Waiting on data from application layer to transmit (see DL_ISDUTransport)
ISDUResponse_4	Transmission of ISDU response data

1729

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	1	2	Start receiving of ISDU request data
T3	2	2	Receive ISDU request data
T4	2	3	Invoke DL_ISDUTransport.ind to AL (see 7.2.1.6)
T5	3	3	Invoke OD.rsp with "busy" indication (see Table A.14)
T6	3	4	-
T7	4	4	Invoke OD.rsp with ISDU response data
T8	4	1	-
T9	2	1	-
T10	3	1	Invoke DL_ISDUAbort
T11	4	1	Invoke DL_ISDUAbort
T12	1	0	-
T13	2	1	Invoke DL_ISDUAbort
T14	1	1	Invoke OD.rsp with "no service" indication (see Table A.12 and Table A.14)
T15	3	1	Invoke DL_ISDUAbort
T16	4	1	Invoke DL_ISDUAbort

1730

INTERNAL ITEMS	TYPE	DEFINITION
ISDUStart	Service	OD.ind(W, ISDU, Start, Data)
ISDUWrite	Service	OD.ind(W, ISDU, FlowCtrl, Data)
ISDURecComplete	Guard	If OD.ind(R, ISDU, Start, ...) received
ISDURespStart	Service	DL_ISDUTransport.rsp()
ISDURead	Service	OD.ind(R, ISDU, Start or FlowCtrl, ...)
ISDUSendComplete	Guard	If OD.ind(R, ISDU, IDLE, ...) received
ISDUAbort	Service	OD.ind(R/W, ISDU, Abort, ...)
ISDUError	Guard	If ISDU structure is incorrect or FlowCTRL error detected

7.3.7 Command handler

7.3.7.1 General

The command handler passes the control code (PDOUTVALID or PDOUTINVALID) contained in the DL_Control.req service primitive to the cyclically operating message handler via the OD.req service and MasterCommands. The message handler uses the page communication channel.

The permissible control codes for output Process Data are listed in Table 55.

Table 55 – Control codes

Control code	MasterCommand	Description
PDOUTVALID	ProcessDataOutputOperate	Output Process Data valid
PDOUTINVALID	DeviceOperate	Output Process Data invalid or missing

The command handler receives input Process Data status information via the PDInStatus service and propagates it within a DL_Control.ind service primitive.

In addition, the command handler translates Device mode change requests from System Management into corresponding MasterCommands (see Table B.2).

7.3.7.2 State machine of the Master command handler

Figure 53 shows the state machine of the Master command handler.

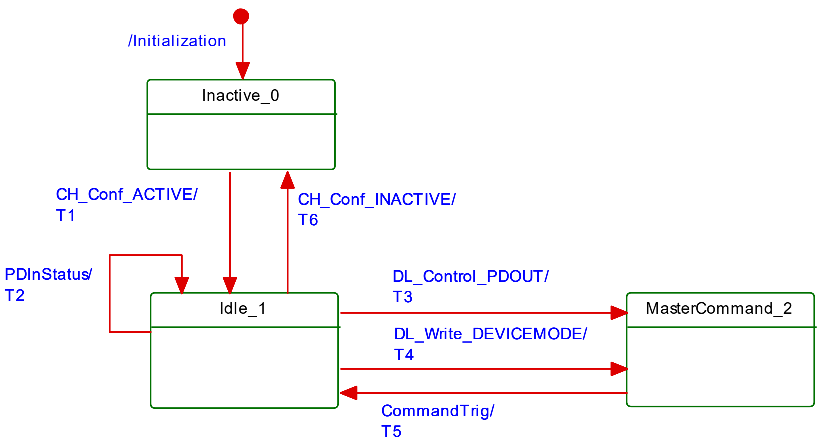


Figure 53 – State machine of the Master command handler

Table 56 shows the state transition tables of the Master command handler.

Table 56 – State transition tables of the Master command handler

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting on activation by DL-mode handler
Idle_1	Waiting on new command from AL: DL_Control (status of output PD) or from SM: DL_Write (change Device mode, for example to OPERATE), or waiting on PDInStatus.ind service primitive.
MasterCommand_2	Prepare data for OD.req service primitive. Waiting on demand from OD handler (CommandTrig).

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	1	1	If service PDInStatus.ind = VALID invoke DL_Control.ind (VALID) to signal valid input Process Data to AL. If service PDInStatus.ind = INVALID invoke DL_Control.ind (INVALID) to signal invalid input Process Data to AL.
T3	1	1	If service DL_Control.req = PDOUTVALID invoke OD.req (WRITE, PAGE, 0, 1, MasterCommand = 0x98). If service DL_Control.req = PDOUTINVALID invoke OD.req (WRITE, PAGE, 0, 1, MasterCommand = 0x99). See Table B.2.
T4	1	2	The services DL_Write_DEVICEMODE translate into: INACTIVE: OD.req (WRITE, PAGE, 0, 1, MasterCommand = 0x5A) STARTUP: OD.req (WRITE, PAGE, 0, 1, MasterCommand = 0x97) PREOPERATE: OD.req (WRITE, PAGE, 0, 1, MasterCommand = 0x9A) OPERATE: OD.req (WRITE, PAGE, 0, 1, MasterCommand = 0x99)
T5	2	1	A call CommandTrig from the OD handler causes the command handler to invoke the OD.req service primitive and subsequently the message handler to send the appropriate MasterCommand to the Device.
T6	1	0	-

INTERNAL ITEMS	TYPE	DEFINITION
DEVICEMODE	Label	Any of the Device modes: INACTIVE, STARTUP, PREOPERATE, or OPERATE
PDOUT	Label	Any of the two output control codes: PDOUTVALID or PDOUTINVALID (see Table 55)

7.3.7.3 State machine of the Device command handler

Figure 54 shows the Device state machine of the command handler. It is mainly driven by MasterCommands from the Master's command handler to control the Device modes and the status of output Process Data. It also controls the status of input Process Data via the PDInStatus service.

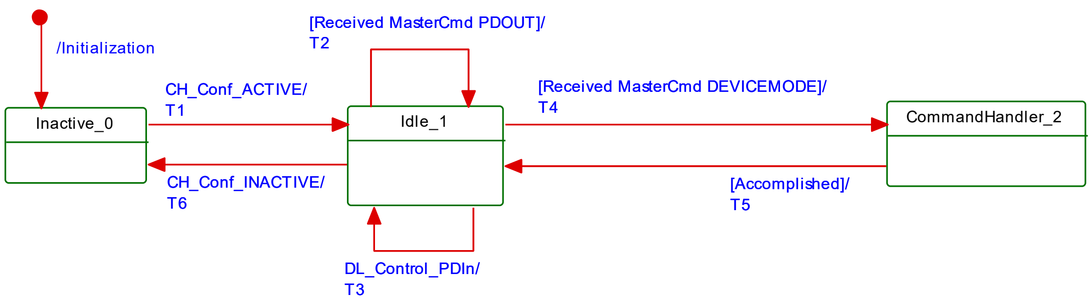


Figure 54 – State machine of the Device command handler

Table 57 shows the state transition tables of the Device command handler.

Table 57 – State transition tables of the Device command handler

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting on activation
Idle_1	Waiting on next MasterCommand
CommandHandler_2	Decompose MasterCommand and invoke specific actions (see B.1.2): If MasterCommand = 0x5A then change Device state to INACTIVE. If MasterCommand = 0x97 then change Device state to STARTUP. If MasterCommand = 0x9A then change Device state to PREOPERATE. If MasterCommand = 0x99 then change Device state to OPERATE.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	1	1	Invoke DL_Control.ind (PDOUTVALID) if received MasterCommand = 0x98. Invoke DL_Control.ind (PDOUTINVALID) if received MasterCommand = 0x99.
T3	1	1	If service DL_Control.req (VALID) then invoke PDInStatus.req (VALID). If service DL_Control.req (INVALID) then invoke PDInStatus.req (INVALID). Message handler uses PDInStatus service to set/reset the PD status flag (see A.1.5)
T4	1	2	-
T5	2	1	-
T6	1	0	-

INTERNAL ITEMS	TYPE	DEFINITION
<none>		

7.3.8 Event handler

7.3.8.1 Events

There are two types of Events, one without details, and another one with details. Events without details may have been implemented in legacy Devices, but they shall not be used for Devices in accordance with this standard. However, all Masters shall support processing of both Events with details and Events without details.

The general structure and coding of Events is specified in A.6. Event codes without details are specified in Table A.16. EventCodes with details are specified in Annex D. The structure of the Event memory for EventCodes with details within a Device is specified in Table 58.

Table 58 – Event memory

Address	Event slot number	Parameter Name	Description
0x00		StatusCode	Summary of status and error information. Also used to control read access for individual messages.
0x01	1	EventQualifier 1	Type, mode and source of the Event
0x02		EventCode 1	16-bit EventCode of the Event
0x03			
0x04	2	EventQualifier 2	Type, mode and source of the Event
0x05		EventCode 2	16-bit EventCode of the Event
0x06			
...			

Address	Event slot number	Parameter Name	Description
0x10	6	EventQualifier 6	Type, mode and source of the Event
0x11		EventCode 6	16-bit EventCode of the Event
0x12			
0x13 to 0x1F			Reserved for future use

7.3.8.2 Event processing

The Device AL writes an Event to the Event memory and then sets the "Event flag" bit, which is sent to the Master in the next message within the CKS octet (see 7.3.3.2 and A.1.5).

Upon reception of a Device reply message with the "Event flag" bit = 1, the Master shall switch from the ISDU handler to the Event handler. The Event handler starts reading the StatusCode.

If the "Event Details" bit is set (see Figure A.22), the Master shall read the Event details of the Events indicated in the StatusCode from the Event memory. Once it has read an Event detail, it shall invoke the service DL_Event.ind. After reception of the service DL_EventConf, the Master shall write any data to the StatusCode to reset the "Event flag" bit. The Event handling on the Master shall be completed regardless of the contents of the Event data received (EventQualifier, EventCode).

If the "Event Details" bit is not set (see Figure A.21) the Master Event handler shall generate the standardized Events according to Table A.16 beginning with the most significant bit in the EventCode.

Write access to the StatusCode indicates the end of Event processing to the Device. The Device shall ignore the data of this Master Write access. The Device then resets the "Event flag" bit and may now change the content of the fields in the Event memory.

7.3.8.3 State machine of the Master Event handler

Figure 55 shows the Master state machine of the Event handler.

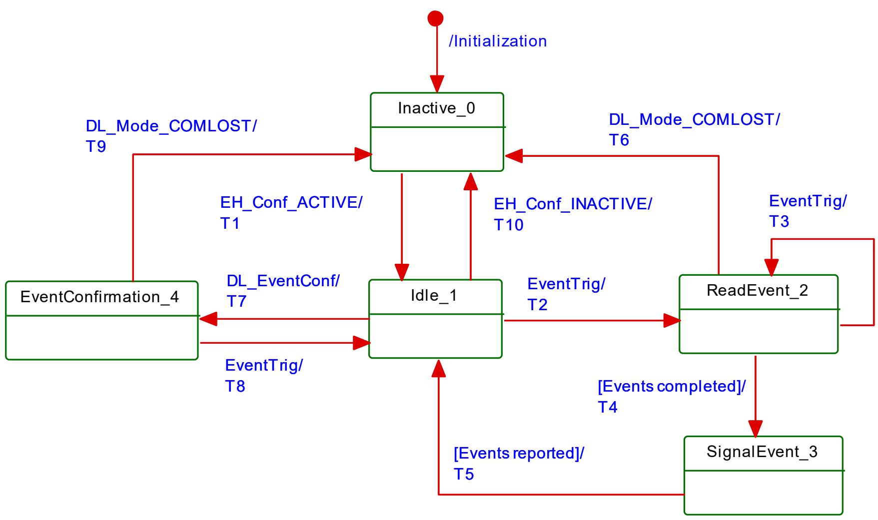


Figure 55 – State machine of the Master Event handler

Table 59 shows the state transition tables of the Master Event handler.

Table 59 – State transition tables of the Master Event handler

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting on activation
Idle_1	Waiting on next Event indication ("EventTrig" through On-request Data handler) or Event confirmation through service DL_EventConf from Master AL.
ReadEvent_2	Read Event data set from Device message by message through Event memory address. Check StatusCode for number of activated Events (see Table 58).
SignalEvent_3	Analyze Event data and invoke DL_Event indication to Master AL (see 7.2.1.15) for each available Event.
EventConfirmation_4	Waiting on Event confirmation transmission via service OD.req to the Device

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	1	2	Read Event StatusCode octet via service OD.req (R, DIAGNOSIS, Event memory address = 0, 1)
T3	2	2	Read octets from Event memory via service OD.req (R, DIAGNOSIS, incremented Event memory address, 1)
T4	2	3	-
T5	3	1	-
T6	2	0	-
T7	1	4	-
T8	4	1	Invoke OD.req (W, DIAGNOSIS, 0, 1, any data) with Write access to "StatusCode" (see Table 58) to confirm Event readout to Device
T9	4	0	-
T10	1	0	-

INTERNAL ITEMS	TYPE	DEFINITION
<None>		

7.3.8.4 State machine of the Device Event handler

Figure 56 shows the state machine of the Device Event handler.

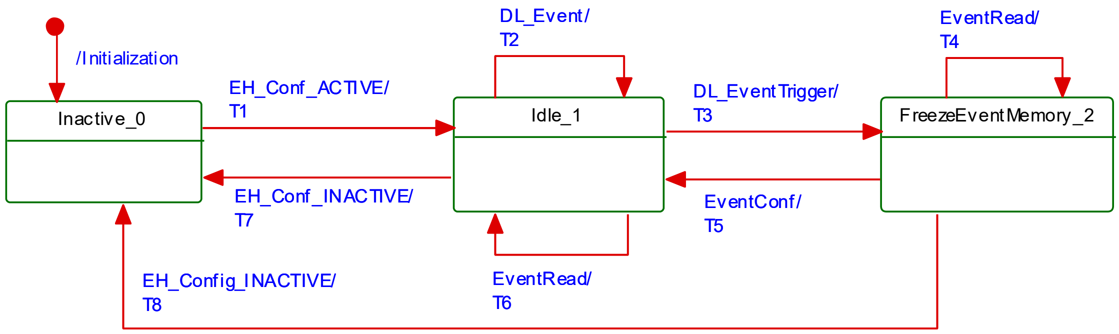


Figure 56 – State machine of the Device Event handler

Table 60 shows the state transition tables of the Device Event handler.

Table 60 – State transition tables of the Device Event handler

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting on activation
Idle_1	Waiting on DL-Event service from AL providing Event data and the DL_EventTrigger service to fire the "Event flag" bit (see A.1.5)
FreezeEventMemory_2	Waiting on readout of the Event memory and on Event memory readout confirmation through write access to the StatusCode

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	1	1	Change Event memory entries with new Event data (see Table 58)
T3	1	2	Invoke service EventFlag.req (Flag = TRUE) to indicate Event activation to the Master via the "Event flag" bit. Mark all Event slots in memory as not changeable.
T4	2	2	Master requests Event memory data via EventRead (= OD.ind). Send Event data by invoking OD.rsp with Event data of the requested Event memory address.
T5	2	1	Invoke service EventFlag.req (Flag = FALSE) to indicate Event deactivation to the Master via the "Event flag" bit. Mark all Event slots in memory as invalid according to A.6.3.
T6	1	1	Send contents of Event memory by invoking OD.rsp with Event data
T7	1	0	-
T8	2	0	Discard Event memory data

INTERNAL ITEMS	TYPE	DEFINITION
EventRead	Service	OD.ind (R, DIAGNOSIS, Event memory address, length, data)
EventConf	Service	OD.ind (W, DIAGNOSIS, address = 0, data = don't care)

8 Application layer (AL)

8.1 General

Figure 57 shows an overview of the structure and services of the Master application layer (AL).

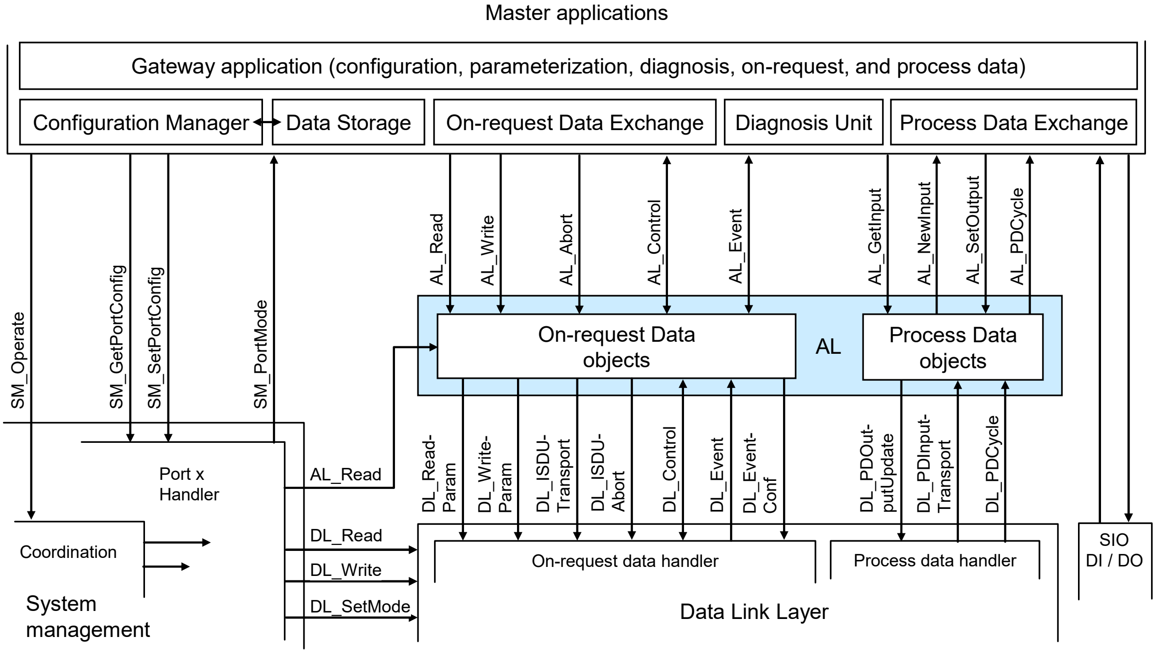


Figure 57 – Structure and services of the application layer (Master)

Figure 58 shows an overview of the structure and services of the Device application layer (AL).

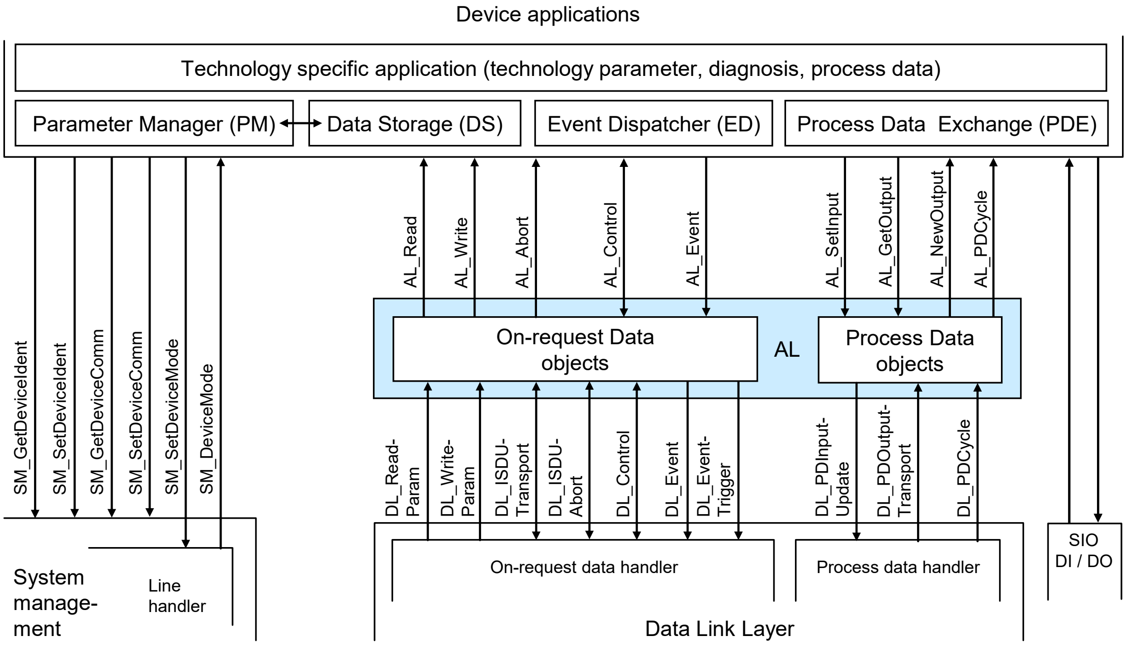


Figure 58 – Structure and services of the application layer (Device)

8.2 Application layer services

8.2.1 AL services within Master and Device

This clause defines the services of the application layer (AL) to be provided to the Master and Device applications and System Management via its external interfaces. Table 61 lists the assignments of Master and Device to their roles as initiator or receiver for the individual AL services. Empty fields indicate no availability of this service on Master or Device.

Table 61 – AL services within Master and Device

Service name	Master	Device
AL_Read	R	I
AL_Write	R	I
AL_Abort	R	I
AL_GetInput	R	
AL_NewInput	I	
AL_SetInput		R
AL_PDCycle	I	I
AL_GetOutput		R
AL_NewOutput		I
AL_SetOutput	R	
AL_Event	I / R	R
AL_Control	I / R	R / I
Key (see 3.3.4) I Initiator of service R Receiver (Responder) of service		

8.2.2 AL Services

8.2.2.1 AL_Read

The AL_Read service is used to read On-request Data from a Device connected to a specific port. The parameters of the service primitives are listed in Table 62.

Table 62 – AL_Read

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

Argument
The service-specific parameters are transmitted in the argument.

Port
This parameter contains the port number for the On-request Data to be read.

Parameter type: Unsigned8

Index
This parameter indicates the address of On-request Data objects to be read from the Device. Index 0 in conjunction with Subindex 0 addresses the entire set of Direct Parameters from 0 to 15 (see Direct Parameter page 1 in Table B.1) or in conjunction with Subindices 1 to 16 the individual parameters from 0 to 15. Index 1 in conjunction with Subindex 0 addresses the entire set of Direct Parameters from addresses 16 to 31 (see Direct Parameter page 2 in Table B.1) or in conjunction with Subindices 1 to 16 the individual parameters from

16 to 31. It uses the page communication channel (see Figure 7) for both and always returns a positive result. For all the other indices (see B.2) the ISDU communication channel is used.

Permitted values: 0 to 65535 (See B.2.1 for constraints)

Subindex
This parameter indicates the element number within a structured On-request Data object. A value of 0 indicates the entire set of elements.

Permitted values: 0 to 255

Result (+):
This selection parameter indicates that the service has been executed successfully.

Port
This parameter contains the port number of the requested On-request Data.

Data
This parameter contains the read values of the On-request Data.

Parameter type: Octet string

Result (-):
This selection parameter indicates that the service failed.

Port
This parameter contains the port number for the requested On-request Data.

ErrorInfo
This parameter contains error information.

Permitted values: see Annex C

NOTE The AL maps DL ErrorInfos into its own AL ErrorInfos using Annex C.

8.2.2.2 AL_Write

The AL_Write service is used to write On-request Data to a Device connected to a specific port. The parameters of the service primitives are listed in Table 63.

Table 63 – AL_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 are transmitted in the argument.

Port
This parameter contains the port number for the On-request Data to be written.

Parameter type: Unsigned8

Index
This parameter indicates the address of On-request Data objects to be written to the Device. Index 0 always returns a negative result except for use in conjunction with Subindex 16 at Devices without ISDU support. Index 1 in conjunction with Subindex 0 addresses the entire set of Direct Parameters from addresses 16 to 31 (see Direct Parameter page 2 in Table

B.1) or in conjunction with Subindices 1 to 16 the individual parameters from 16 to 31. It uses the page communication channel (see Figure 7) in case of Index 1 and always returns a positive result. For all other Indices (see B.2) the ISDU communication channel is used.

Permitted values: 1 to 65535 (see Table 102)

Subindex
This parameter indicates the element number within a structured On-request Data object. A value of 0 indicates the entire set of elements.

Permitted values: 0 to 255

Data
This parameter contains the values of the On-request Data.

Parameter type: Octet string

Result (+):
This selection parameter indicates that the service has been executed successfully.

Port
This parameter contains the port number of the On-request Data.

Result (-):
This selection parameter indicates that the service failed.

Port
This parameter contains the port number of the On-request Data.

ErrorInfo
This parameter contains error information.

Permitted values: see Annex C

8.2.2.3 AL_Abort

The AL_Abort service is used to abort a current AL_Read or AL_Write service on a specific port. Invocation of this service abandons the response to an AL_Read or AL_Write service in progress on the Master. The parameters of the service primitives are listed in Table 64.

Table 64 – AL_Abort

Parameter name	.req	.ind
Argument	M	M
Port	M	

Argument
The service-specific parameter is transmitted in the argument.

Port
This parameter contains the port number of the service to be abandoned.

8.2.2.4 AL_GetInput

The AL_GetInput service reads the input data within the Process Data provided by the data link layer of a Device connected to a specific port. The parameters of the service primitives are listed in Table 65.

Table 65 – AL_GetInput

Parameter name	.req	.cnf
Argument	M	
Port	M	

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

Argument
The service-specific parameters are transmitted in the argument.

Port
This parameter contains the port number for the Process Data to be read.

Result (+):
This selection parameter indicates that the service has been executed successfully.

Port
This parameter contains the port number for the Process Data.

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

Parameter type: Octet string

Result (-):
This selection parameter indicates that the service failed.

Port
This parameter contains the port number for the Process Data.

ErrorInfo
This parameter contains error information.

Permitted values:
NO_DATA (DL did not provide Process Data)

8.2.2.5 AL_NewInput

The AL_NewInput local service indicates the receipt of updated input data within the Process Data of a Device connected to a specific port. The parameters of the service primitives are listed in Table 66.

Table 66 – AL_NewInput

Parameter name	.ind
Argument	M
Port	M

Argument
The service-specific parameter is transmitted in the argument.

Port
This parameter specifies the port number of the received Process Data.

8.2.2.6 AL_SetInput

The AL_SetInput local service updates the input data within the Process Data of a Device. The parameters of the service primitives are listed in Table 67.

Table 67 – AL_SetInput

Parameter name	.req	.cnf
Argument	M	
InputData	M	

Result (+)		S
Result (-) ErrorInfo		S M

Argument
The service-specific parameters are transmitted in the argument.

InputData
This parameter contains the Process Data values of the input data to be transmitted.

Parameter type: Octet string

Result (+):
This selection parameter indicates that the service has been executed successfully.

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
STATE_CONFLICT (Service unavailable within current state)

8.2.2.7 AL_PDCycle

The AL_PDCycle local service indicates the end of a Process Data cycle. The Device application can use this service to transmit new input data to the application layer via AL_SetInput. The parameters of the service primitives are listed in Table 68.

Table 68 – AL_PDCycle

Parameter name	.ind
Argument Port	 O

Argument
The service-specific parameter is transmitted in the argument.

Port
This parameter contains the port number of the received new Process Data (Master only).

8.2.2.8 AL_GetOutput

The AL_GetOutput service reads the output data within the Process Data provided by the data link layer of the Device. The parameters of the service primitives are listed in Table 69.

Table 69 – AL_GetOutput

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

Argument
The service-specific parameters are transmitted in the argument.

Result (+):
This selection parameter indicates that the service has been executed successfully.

OutputData
This parameter contains the Process Data values of the requested output data.

Parameter type: Octet string

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
NO_DATA (DL did not provide Process Data)

8.2.2.9 AL_NewOutput

The AL_NewOutput local service indicates the receipt of updated output data within the Process Data of a Device. This service has no parameters. The service primitives are shown in Table 70.

Table 70 – AL_NewOutput

Parameter name	.ind
<None>	

8.2.2.10 AL_SetOutput

The AL_SetOutput local service updates the output data within the Process Data of a Master. The parameters of the service primitives are listed in Table 71.

Table 71 – AL_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 are transmitted in the argument.

Port

This parameter contains the port number of the Process Data to be written.

OutputData
This parameter contains the output data to be written at the specified port.

Parameter type: Octet string

Result (+):
This selection parameter indicates that the service has been executed successfully.

Port
This parameter contains the port number for the Process Data.

Result (-):
This selection parameter indicates that the service failed.

Port
This parameter contains the port number for the Process Data.

ErrorInfo
This parameter contains error information.

Permitted values:
STATE_CONFLICT (Service unavailable within current state)

8.2.2.11 AL_Event

The AL_Event service indicates up to 6 pending status or error messages. The source of one Event can be local (Master) or remote (Device). The Event can be triggered by a communication layer or by an application. The parameters of the service primitives are listed in Table 72.

Table 72 – AL_Event

Parameter name		.req	.ind	.rsp	.cnf
Argument		M	M	M	M
Port			M	M	M
EventCount		M	M		
Event(1)	Instance	M	M		
	Mode	M	M		
	Type	M	M		
	Origin		M		
	EventCode	M	M		
…					
Event(n)	Instance	M	M		
	Mode	M	M		
	Type	M	M		
	Origin		M		
	EventCode	M	M		

Argument
The service-specific parameters are transmitted in the argument.

Port
This parameter contains the port number of the Event data.

EventCount
This parameter indicates the number n (1 to 6) of Events in the Event memory.

Event(x)
Depending on EventCount this parameter exists n times. Each instance contains the following elements.

Instance
This parameter indicates the Event source.

Permitted values: Application (see Table A.17)

Mode
This parameter indicates the Event mode.

Permitted values: SINGLESHOT, APPEARS, DISAPPEARS (see Table A.20)

Type
This parameter indicates the Event category.

Permitted values: ERROR, WARNING, NOTIFICATION (see Table A.19)

Origin
This parameter indicates whether the Event was generated in the local communication section or remotely (in the Device).

Permitted values: LOCAL, REMOTE

EventCode
This parameter contains a code identifying a certain Event.

Permitted values: see Annex D

8.2.2.12 AL_Control

The AL_Control service contains the Process Data qualifier status information transmitted to and from the Device application. This service shall be synchronized with AL_GetInput and AL_SetOutput respectively (see 11.7.2.1). The parameters of the service primitives are listed in Table 73.

Table 73 – AL_Control

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

Argument
The service-specific parameters are transmitted in the argument.

Port
This parameter contains the number of the related port.

ControlCode
This parameter contains the qualifier status of the Process Data (PD).

Permitted values:
VALID (Input Process Data valid)
INVALID (Input Process Data invalid)
PDOUTVALID (Output Process Data valid, see Table 55)
PDOUTINVALID (Output Process Data invalid, see Table 55)

8.3 Application layer protocol

8.3.1 Overview

Figure 8 shows that the application layer offers services for data objects which are transformed into the special communication channels of the data link layer.

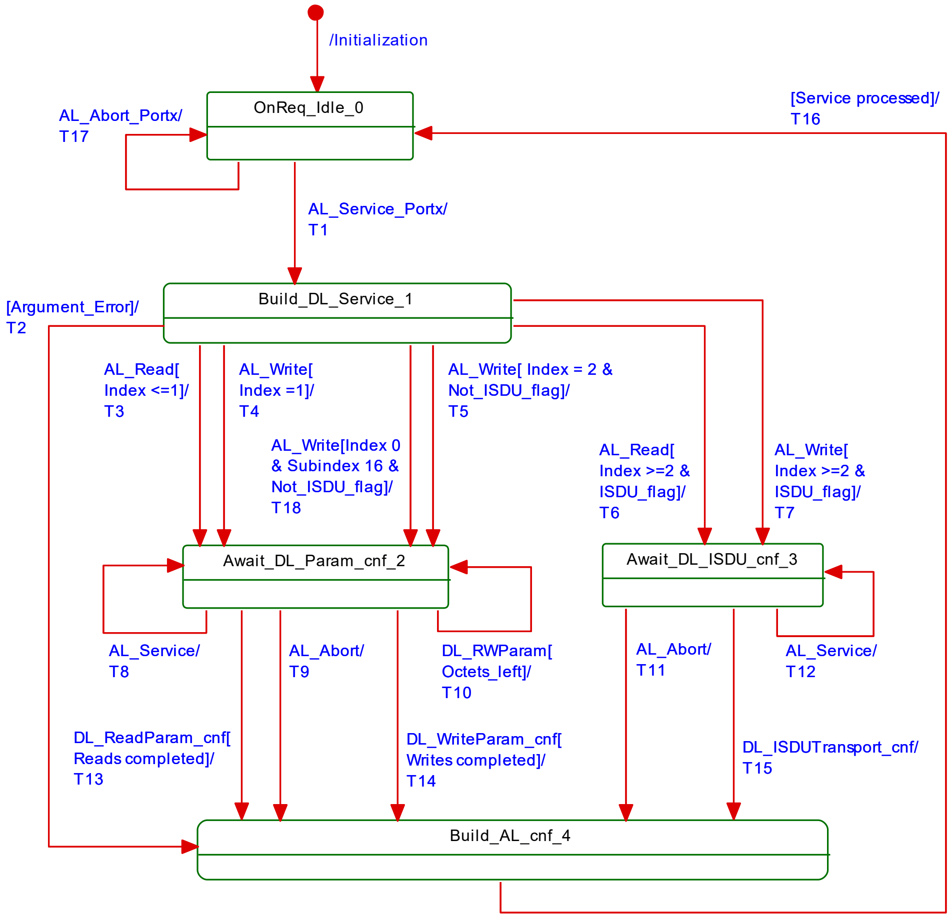
The application layer manages the data transfer with all its assigned ports. That means, AL service calls need to identify the particular port they are related to.

8.3.2 On-request Data transfer

8.3.2.1 OD state machine of the Master AL

Figure 59 shows the state machine for the handling of On-request Data (OD) within the application layer.

"AL_Service" represents any AL service in Table 61 related to OD. "Portx" indicates a particular port number.



2096

2097 **Figure 59 – OD state machine of the Master AL**

2098 Table 74 shows the states and transitions for the OD state machine of the Master AL.

2099 **Table 74 – States and transitions for the OD state machine of the Master AL**

STATE NAME	STATE DESCRIPTION
OnReq_Idle_0	AL service invocations from the Master applications or from the SM Portx handler (see Figure 57) can be accepted within this state.
Build_DL_Service_1	Within this state AL service calls are checked, and corresponding DL services are created within the subsequent states. In case of an error in the arguments of the AL service a negative AL confirmation is created and returned.
Await_DL_Param_cnf_2	Within this state the AL service call is transformed in a sequence of as many DL_ReadParam or DL_WriteParam calls as needed (Direct Parameter page access; see page communication channel in Figure 7). All asynchronously occurred AL service invocations except AL_Abort are rejected (see 3.3.7).
Await_DL_ISDU_cnf_3	Within this state the AL service call is transformed in a DL_ISDUTransport service call (see ISDU communication channel in Figure 7). All asynchronously occurred AL service invocations except AL_Abort are rejected (see 3.3.7).
Build_AL_cnf_4	Within this state an AL service confirmation is created depending on an argument error, the DL service confirmation, or an AL_Abort.

2100

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	Memorize the port number "Portx".
T2	1	4	Prepare negative AL service confirmation.
T3	1	2	Prepare DL_ReadParam for Index 0 or 1.
T4	1	2	Prepare DL_WriteParam for Index 1.
T5	1	2	Prepare DL_Write for Address 0x0F if the Device does not support ISDU.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T6	1	3	Prepare DL_ISDUTransport (read)
T7	1	3	Prepare DL_ISDUTransport (write)
T8	2	2	Return negative AL service confirmation on this asynchronous service call.
T9	2	4	All current DL service actions are abandoned, and a negative AL service confirmation is prepared.
T10	2	2	Call next DL_ReadParam or DL_WriteParam service if not all OD are transferred.
T11	3	4	All current DL service actions are abandoned, and a negative AL service confirmation is prepared.
T12	3	3	Return negative AL service confirmation on this asynchronous service call.
T13	2	4	Prepare positive AL service confirmation.
T14	2	4	Prepare positive AL service confirmation.
T15	3	4	Prepare positive AL service confirmation.
T16	4	0	Return positive AL service confirmation with port number "Portx".
T17	0	0	Return negative AL service confirmation with port number "Portx".
T18	1	2	Prepare DL_Write for Address 0x0F if the Device does not support ISDU.

INTERNAL ITEMS	TYPE	DEFINITION
Argument_Error	Bool	Illegal values within the service body, for example "Port number or Index out of range"
DL_RWParam	Label	"DL_RWParam": DL_WriteParam_cnf or DL_ReadParam_cnf
Completed	Bool	No more OD left for transfer
Octets_left	Bool	More OD for transfer
Portx	Variable	Service body variable indicating the port number
ISDU_Flag	Bool	Device supports ISDU
AL_Service	Label	"AL_Service" represents any AL service in Table 61 related to OD

8.3.2.2 OD state machine of the Device AL

Figure 60 shows the state machine for the handling of On-request Data (OD) within the application layer of a Device.

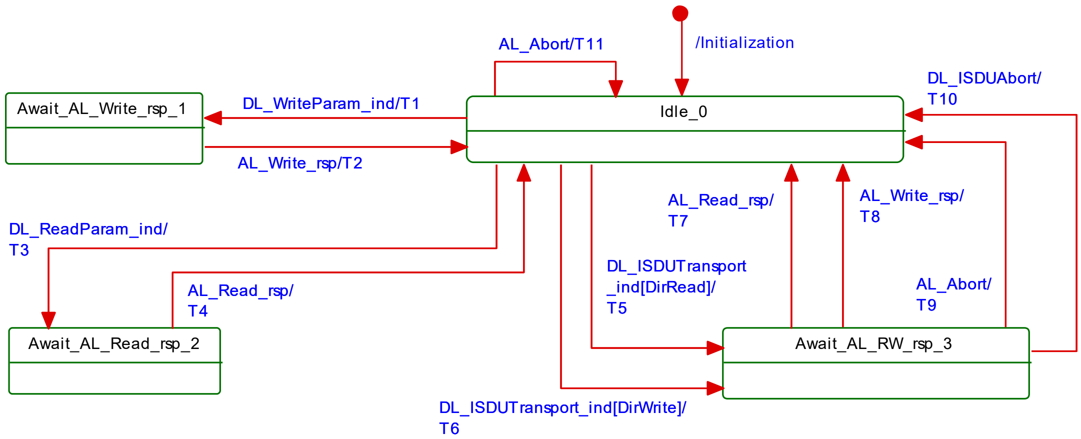


Figure 60 – OD state machine of the Device AL

Table 75 shows the states and transitions for the OD state machine of the Device AL.

Table 75 – States and transitions for the OD state machine of the Device AL

STATE NAME	STATE DESCRIPTION
Idle_0	The Device AL is waiting on subordinated DL service calls triggered by Master messages.
Await_AL_Write_rsp_1	The Device AL is waiting on a response from the technology specific application (write access to Direct Parameter page).
Await_AL_Read_rsp_2	The Device AL is waiting on a response from the technology specific application (read access to Direct Parameter page).
Await_AL_RW_rsp_3	The Device AL is waiting on a response from the technology specific application (read or write access via ISDU).

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	Invoke AL_Write.
T2	1	0	Invoke DL_WriteParam (16 to 31).
T3	0	2	Invoke AL_Read.
T4	2	0	Invoke DL_ReadParam (0 to 31).
T5	0	3	Invoke AL_Read.
T6	0	3	Invoke AL_Write.
T7	3	0	Invoke DL_ISDUTransport (read)
T8	3	0	Invoke DL_ISDUTransport (write)
T9	3	0	Current AL_Read or AL_Write abandoned upon this asynchronous AL_Abort service call. Return negative DL_ISDUTransport (see 3.3.7).
T10	3	0	Current waiting on AL_Read or AL_Write abandoned.
T11	0	0	Current DL_ISDUTransport abandoned. All OD are set to "0".

INTERNAL ITEMS	TYPE	DEFINITION
DirRead	Bool	Access direction: DL_ISDUTransport (read) causes an AL_Read
DirWrite	Bool	Access direction: DL_ISDUTransport (write) causes an AL_Read

8.3.2.3 Sequence diagrams for On-request Data

Figure 61 through Figure 63 demonstrate complete interactions between Master and Device for several On-request Data exchange use cases.

Figure 61 demonstrates two examples for the exchange of On-request Data. For Indices > 1 this is performed with the help of ISDUs and corresponding DL services (ISDU communication channel according to Figure 7). Access to Direct Parameter pages 0 and 1 uses different DL services (page communication channel according to Figure 7)

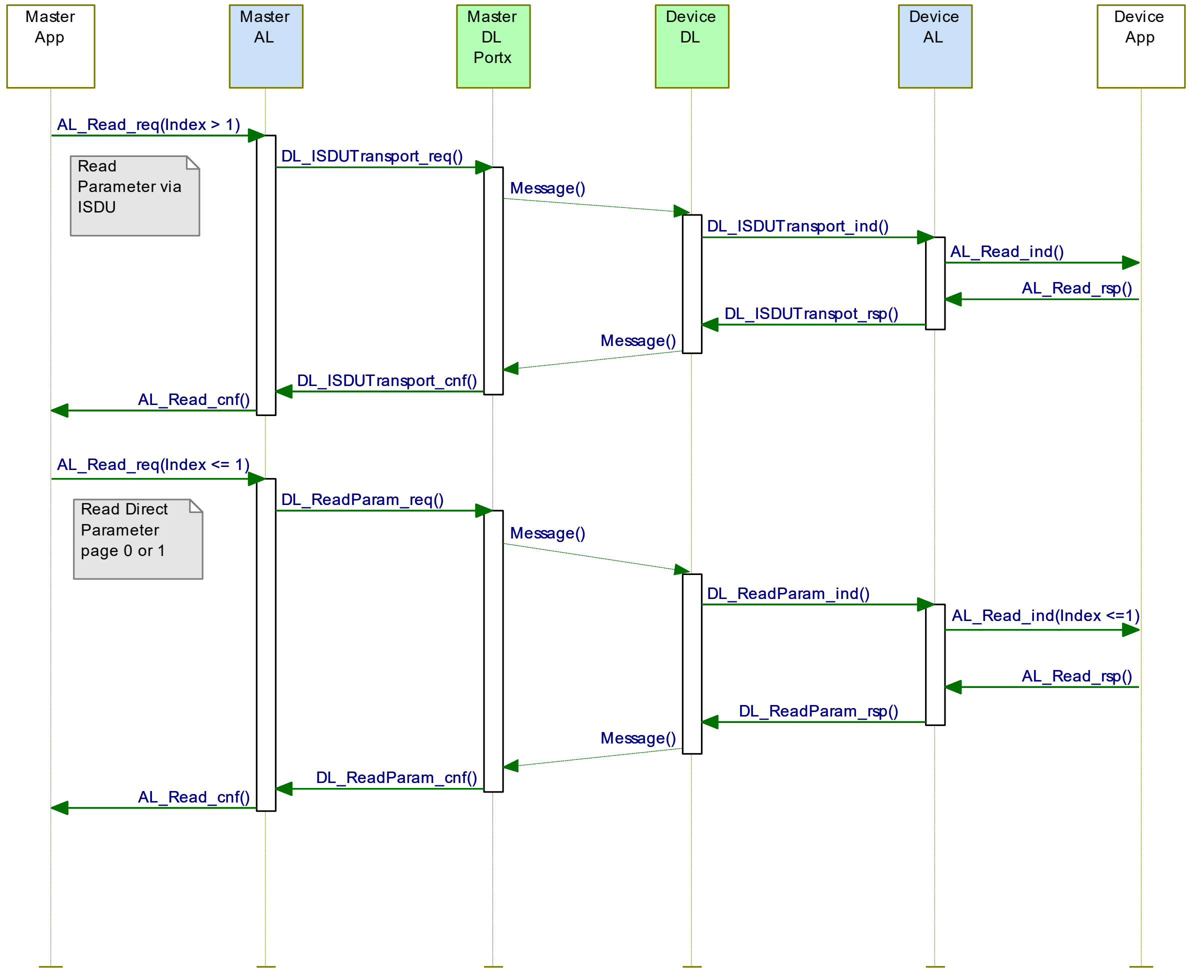


Figure 61 – Sequence diagram for the transmission of On-request Data

Figure 62 demonstrates the behaviour of On-request Data exchange in case of an error such as requested Index not available (see Table C.1).

Another possible error occurs when the Master application (gateway) tries to read an Index > 1 from a Device, which does not support ISDU. The Master AL would respond immediately with "NO_ISDU_SUPPORTED" as the features of the Device are acquired during start-up through reading the Direct Parameter page 1 via the parameter "M-sequence Capability" (see Table B.1).

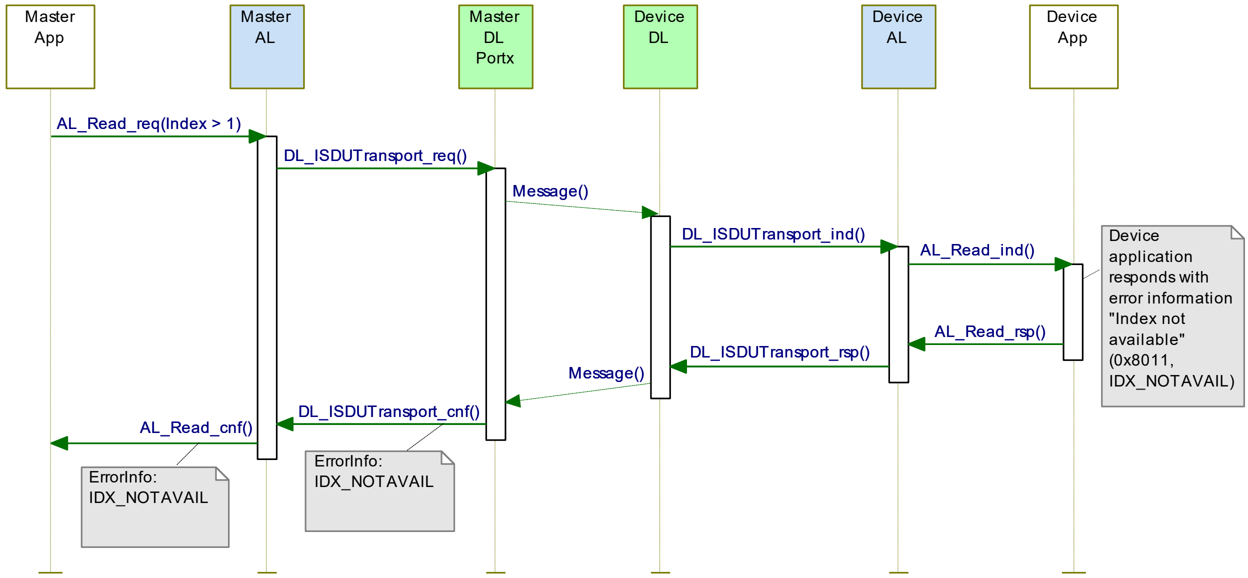


Figure 62 – Sequence diagram for On-request Data in case of errors

Figure 63 demonstrates the behaviour of On-request Data exchange in case of an ISDU timeout (5 000 ms). A Device shall respond within less than the "ISDU acknowledgment time" (see 10.8.5).

NOTE See Table 102 for system constants such as "ISDU acknowledgment time".

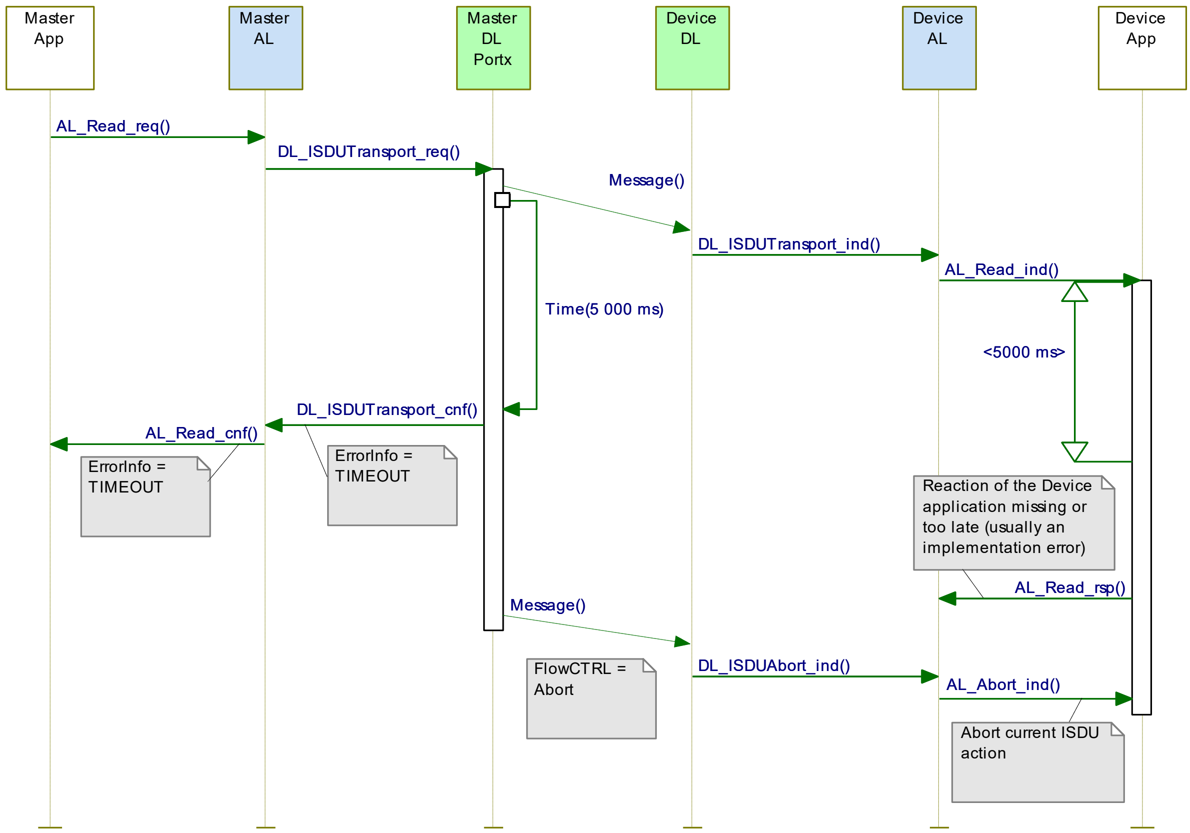


Figure 63 – Sequence diagram for On-request Data in case of timeout

8.3.3 Event processing

8.3.3.1 Event state machine of the Master AL

Figure 64 shows the Event state machine of the Master application layer.

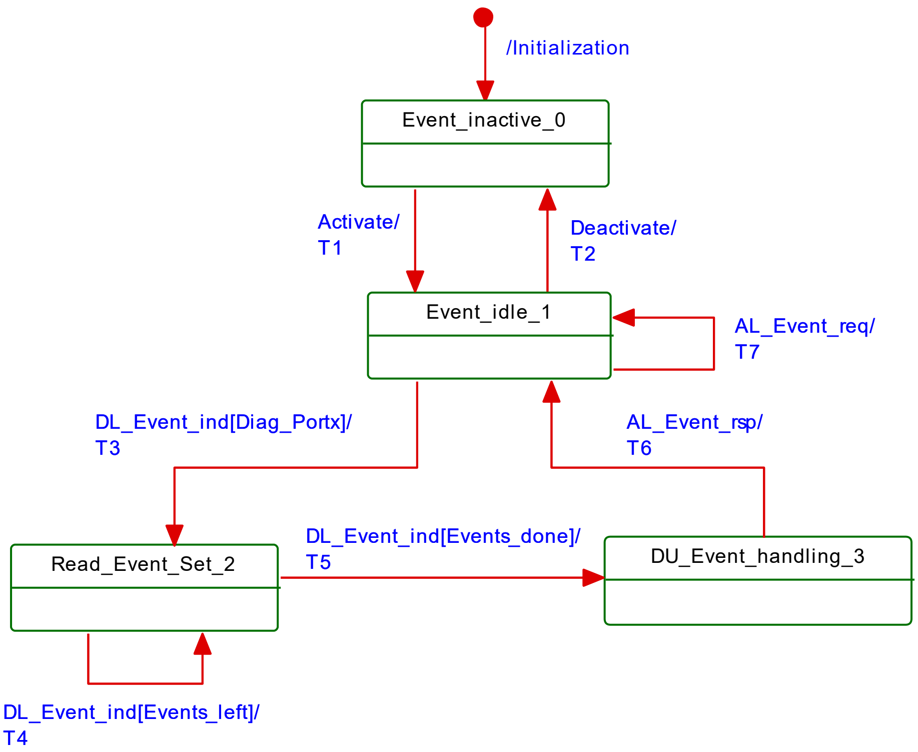


Figure 64 – Event state machine of the Master AL

Table 76 specifies the states and transitions of the Event state machine of the Master application layer.

Table 76 – State and transitions of the Event state machine of the Master AL

STATE NAME	STATE DESCRIPTION
Event_inactive_0	The AL Event handling of the Master is inactive.
Event_idle_1	The Master AL is ready to accept DL_Events (diagnosis information) from the DL.
Read_Event_Set_2	The Master AL received a DL_Event_ind with diagnosis information. After this first DL_Event.ind, the AL collects the complete set (1 to 6) of DL_Events of the current EventTrigger (see 11.6).
DU_Event_handling_3	The Master AL remains in this state as long as the Diagnosis Unit (see 11.6) did not acknowledge the AL_Event.ind.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	1	0	-
T3	1	2	-
T4	2	2	-
T5	2	3	AL_Event.ind
T6	3	1	DL_EventConf.req
T7	1	1	AL_Event.ind

INTERNAL ITEMS	TYPE	DEFINITION
Diag_Portx	Bool	Event set contains diagnosis information with details.
Events_done	Bool	Event set is processed.
Events_left	Bool	Event set not yet completed.

8.3.3.2 Event state machine of the Device AL

Figure 65 shows the Event state machine of the Device application layer

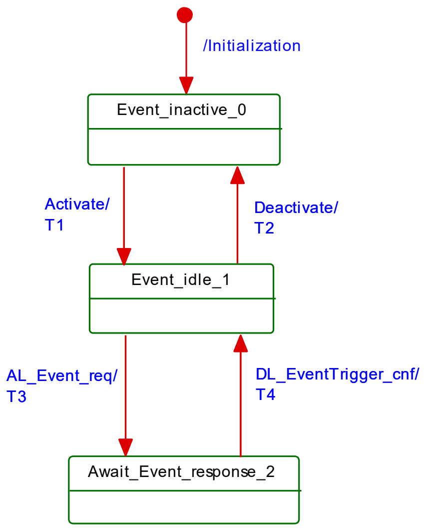


Figure 65 – Event state machine of the Device AL

Table 77 specifies the states and transitions of the Event state machine of the Device application layer.

Table 77 – State and transitions of the Event state machine of the Device AL

STATE NAME	STATE DESCRIPTION
Event_inactive_0	The AL Event handling of the Device is inactive.
Event_idle_1	The Device AL is ready to accept AL_Events (diagnosis information) from the technology specific Device applications for the transfer to the DL. The Device applications can create new Events during this time.
Await_event_response_2	The Device AL propagated an AL_Event with diagnosis information and waits on a DL_EventTrigger confirmation of the DL. The Device AL shall not accept any new AL_Event during this time.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	1	0	-
T3	1	2	An AL_Event request triggers a DL_Event and the corresponding DL_EventTrigger service. The DL_Event carries the diagnosis information from AL to DL. The DL_EventTrigger sets the Event flag within the cyclic data exchange (see A.1.5).
T4	2	1	A DL_EventTrigger confirmation triggers an AL_Event confirmation.

INTERNAL ITEMS	TYPE	DEFINITION
none		

8.3.3.3 Single Event scheduling

Figure 66 shows how a single Event from a Device is processed, in accordance with the relevant state machines.

- The Device application creates an Event request (Step 1), which is passed from the AL to the DL and buffered within the Event memory (see Table 58).
- The Device AL activates the EventTrigger service to raise the Event flag, which causes the Master to read the Event from the Event memory.

- The Master then propagates this Event to the gateway application (Step 2), and waits for an Event acknowledgment.
- Once the Event acknowledgment is received (Step 3), it is indicated to the Device by writing to the StatusCode (Step 4).
- The Device confirms the original Event request to its application (Step 5), which may now initiate a new Event request.

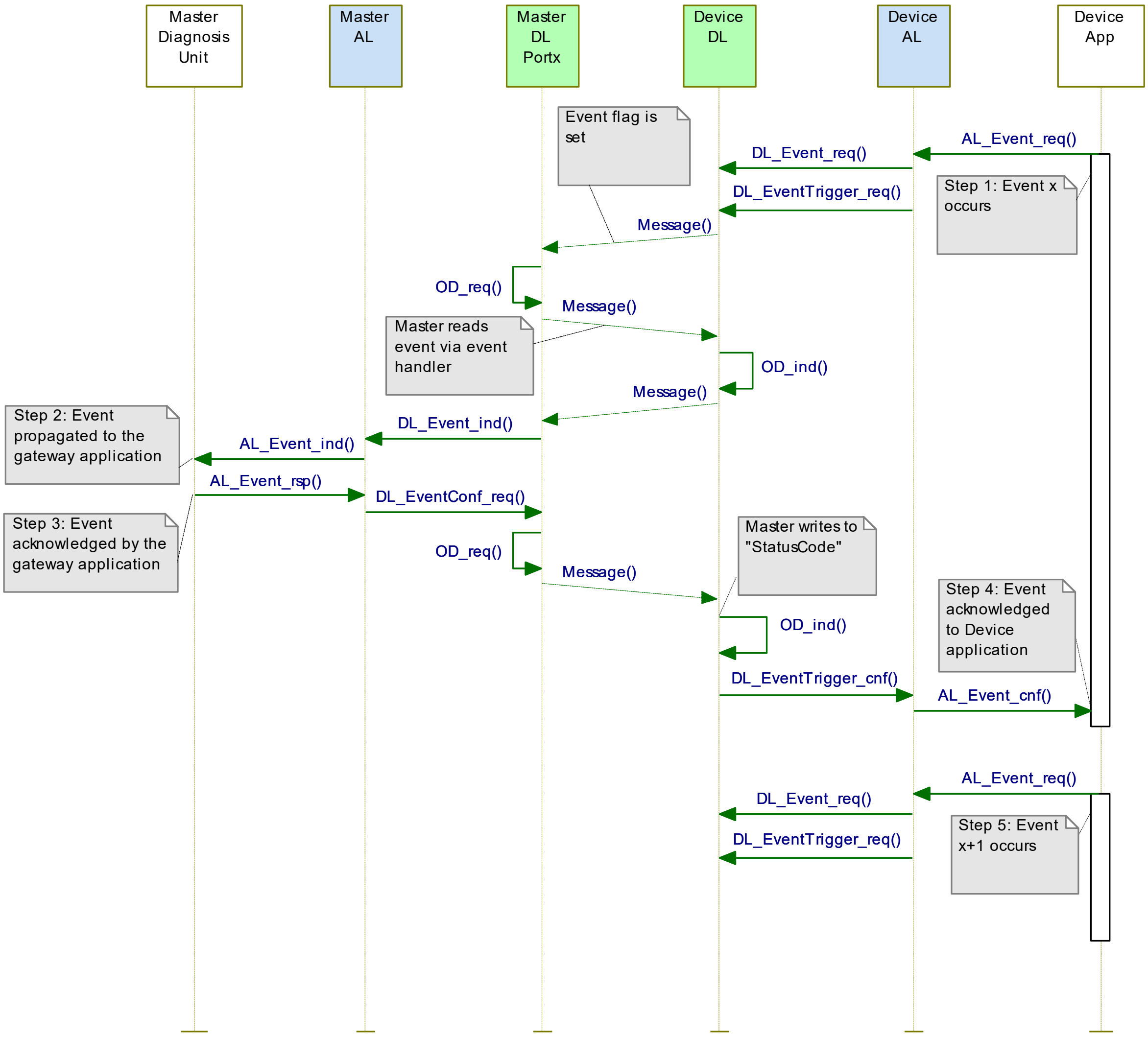


Figure 66 – Single Event scheduling

8.3.3.4 Multi Event transport (legacy Devices only)

Besides the method specified in 0 in which each single Event is conveyed through the layers and acknowledged by the gateway application, all Masters shall support a so-called "multi Event transport" which allows up to 6 Events to be transferred at a time. The Master AL transfers the Event set as a single diagnosis indication to the gateway application and returns a single acknowledgment for the entire set to the legacy Device application.

Figure 66 also applies for the multi Event transport, except that this transport uses one DL_Event indication for each Event memory slot, and a single AL_Event indication for the entire Event set.

One AL_Event.req carries up to 6 Events and one AL_Event.ind indicates up to 6 pending Events. AL_Event.rsp and AL_Event.cnf refer to the indicated entire Event set.

8.3.4 Process Data cycles

Figure 67 and

Figure 68 demonstrate complete interactions between Master and Device for output and input Process Data use cases.

Figure 67 demonstrates how the AL and DL services of Master and Device are involved in the cyclic exchange of output Process Data. The Device application is able to acquire the current values of output PD via the AL_GetOutput service.

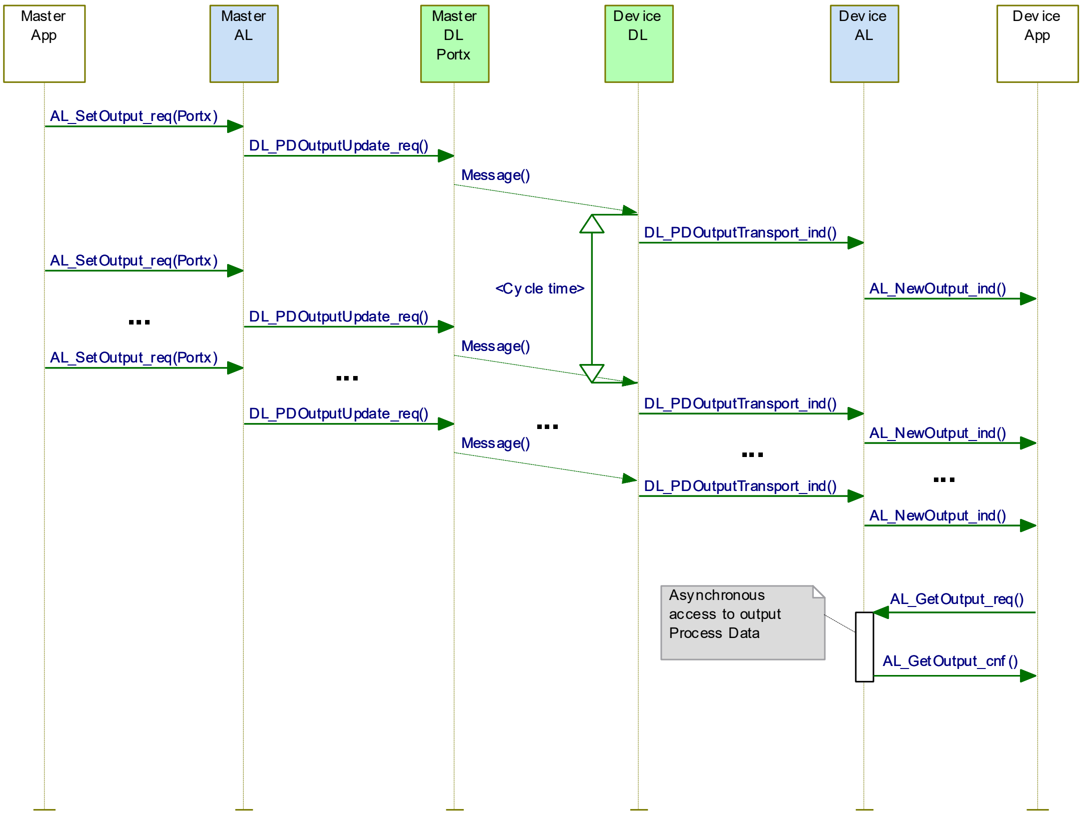


Figure 67 – Sequence diagram for output Process Data

Figure 68 demonstrates how the AL and DL services of Master and Device are involved in the cyclic exchange of input Process Data. The Master application is able to acquire the current values of input PD via the AL_GetInput service.

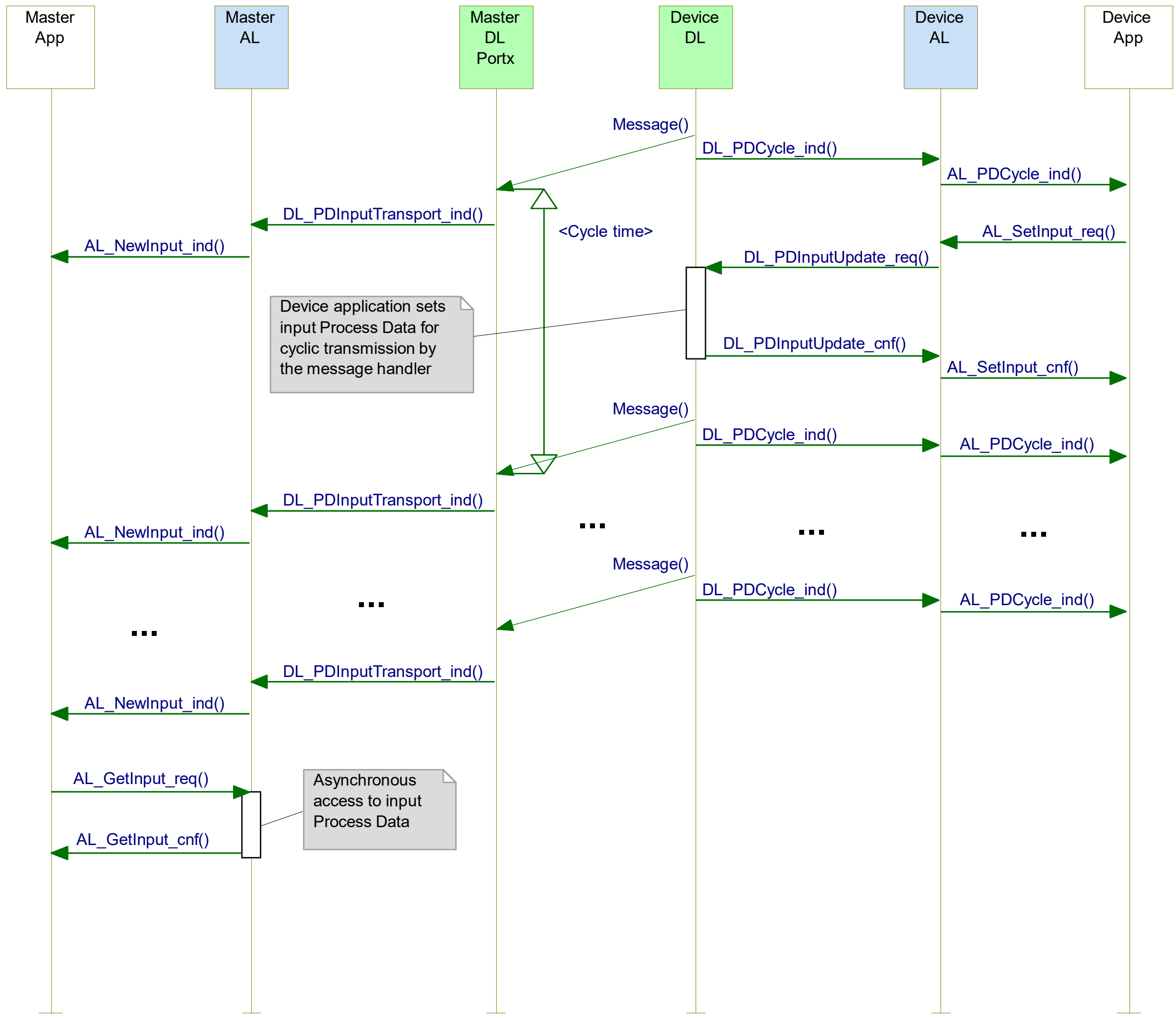


Figure 68 – Sequence diagram for input Process Data

9 System Management (SM)

9.1 General

The SDCI System Management is responsible for the coordinated startup of the ports within the Master and the corresponding operations within the connected Devices. The difference between the SM of the Master and the Device is more significant than with the other layers. Consequently, the structure of this clause separates the services and protocols of Master and Device.

9.2 System Management of the Master

9.2.1 Overview

The Master System Management services are used to set up the Master ports and the system for all possible operational modes.

The Master SM adjusts ports through

- establishing the required communication protocol revision
- checking the Device compatibility (actual Device identifications match expected values)
- adjusting adequate Master M-sequence types and MasterCycleTimes

For this it uses the following services shown in Figure 69:

- SM_SetPortConfig transfers the necessary Device parameters (configuration data) from Configuration Management (CM) to System Mangement (SM). The port is then started implicitly.
- SM_PortMode reports the positive result of the port setup back to CM in case of correct port setup and inspection. It reports the negative result back to CM via corresponding "errors" in case of mismatching revisions and incompatible Devices.
- SM_GetPortConfig reads the actual and effective parameters.
- SM_Operate switches a single port into the "OPERATE" mode.

Figure 69 provides an overview of the structure and services of the Master System Management.

The Master System Management needs one application layer service (AL_Read) to acquire data (communication and identification parameter) from special Indices for inspection.

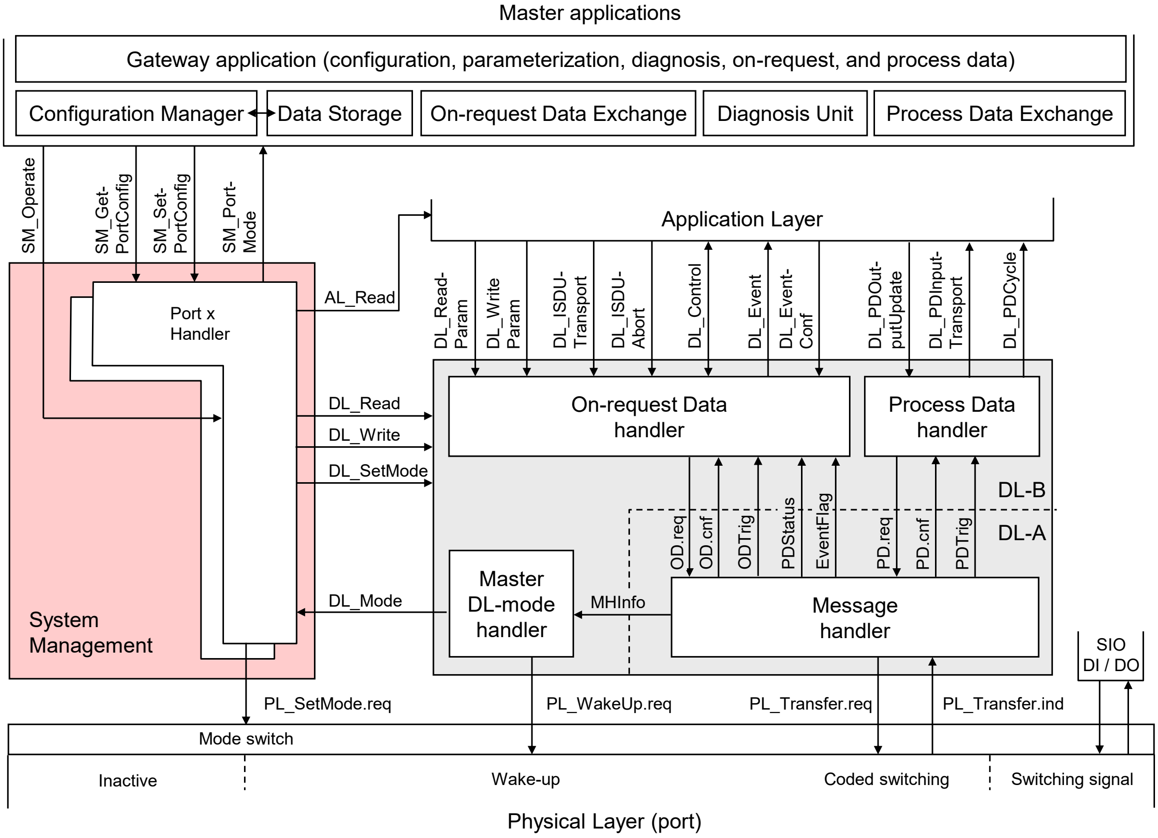


Figure 69 – Structure and services of the Master System Management

Figure 70 demonstrates the actions between the layers Master application (Master App), Configuration Management (CM), System Management (SM), Data Link (DL) and Application Layer (AL) for the startup use case of a particular port.

This particular use case is characterized by the following statements:

- The Device for the available configuration is connected and inspection is successful
- The Device uses the correct protocol version according to this specification
- The configured InspectionLevel is "type compatible" (SerialNumber is read out of the Device and not checked).

Dotted arrows in Figure 70 represent response services to an initial service.

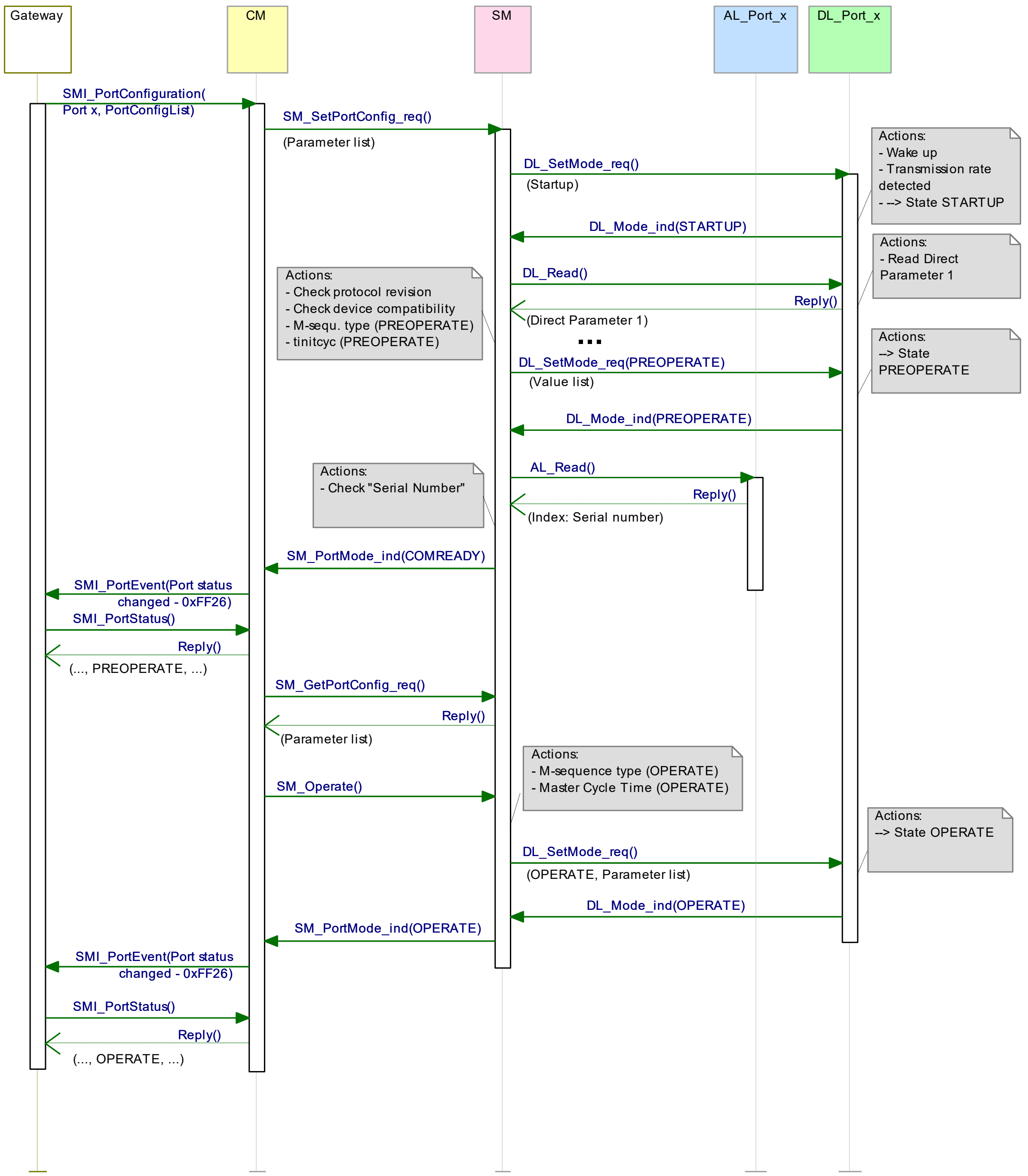


Figure 70 – Sequence chart of the use case "port x setup"

9.2.2 SM Master services

9.2.2.1 Overview

System Management provides the SM Master services to the user via its upper interface. Table 78 lists the assignment of the Master to its role as initiator or receiver for the individual SM services.

Table 78 – SM services within the Master

Service name	Master
SM_SetPortConfig	R
SM_GetPortConfig	R
SM_PortMode	I
SM_Operate	R
Key (see 3.3.4) I Initiator of service R Receiver (Responder) of service	

9.2.2.2 SM_SetPortConfig

The SM_SetPortConfig service is used to set up the requested Device configuration. The parameters of the service primitives are listed in Table 79.

Table 79 – SM_SetPortConfig

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

Argument
The service-specific parameters are transmitted in the argument.

ParameterList
This parameter contains the configured port and Device parameters of a Master port.

Parameter type: Record

Record Elements:

Port Number
This parameter contains the port number

ConfiguredCycleTime
This parameter contains the requested cycle time for the OPERATE mode

Permitted values:
0 (FreeRunning)
Time (see Table B.3)

TargetMode
This parameter indicates the requested operational mode of the port

Permitted values: INACTIVE, DI, DO, CFGCOM, AUTOCOM (see Table 81)

ConfiguredRevisionID (CRID):
Data length: 1 octet for the protocol version (see B.1.5)

InspectionLevel:
Permitted values: NO_CHECK, TYPE_COMP, IDENTICAL (see Table 80)

ConfiguredVendorID (CVID)
Data length: 2 octets

NOTE VendorIDs are assigned by the IO-Link community

ConfiguredDeviceID (CDID)
Data length: 3 octets

ConfiguredFunctionID (CFID)
Data length: 2 octets

ConfiguredSerialNumber (CSN)
Data length: up to 16 octets (see Table 80)

Result (+):
This selection parameter indicates that the service has been executed successfully

Port Number
This parameter contains the port number

Result (-):
This selection parameter indicates that the service failed

Port Number
This parameter contains the port number

ErrorInfo
This parameter contains error information

Permitted values:
PARAMETER_CONFLICT (consistency of parameter set violated)

Table 80 specifies the coding of the different inspection levels (values of the InspectionLevel parameter). See 9.2.3.2 and 11.3.2.

Table 80 – Definition of the InspectionLevel (IL)

Parameter	InspectionLevel (IL)		
	NO_CHECK	TYPE_COMP	IDENTICAL
DeviceID (DID) (compatible)	-	Yes (RDID=CDID)	Yes (RDID=CDID)
VendorID (VID)	-	Yes (RVID=CVID)	Yes (RVID=CVID)
SerialNumber (SN)	-	-	Yes (RSN = CSN)
NOTE "IDENTICAL" = optional (not recommended for new developments)			

Table 81 specifies the coding of the different Target Modes.

Table 81 – Definitions of the Target Modes

Target Mode	Definition
CFGCOM	Device communicating in mode CFGCOM after successful inspection
AUTOCOM	Device communicating in mode AUTOCOM without inspection
INACTIVE	Communication disabled, no DI, no DO
DI	Port in digital input mode (SIO)
DO	Port in digital output mode (SIO)

CFGCOM is a Target Mode based on a user configuration (for example with the help of an IODD) and consistency checking of RID, VID, DID.

AUTOCOM is a Target Mode without configuration. That means no checking of CVID and CDID. The CRID is set to the highest revision the Master is supporting. AUTOCOM should only be selectable together with Inspection Level "NO_CHECK" (see Table 80).

9.2.2.3 SM_GetPortConfig

The SM_GetPortConfig service is used to acquire the real (actual) Device configuration. The parameters of the service primitives are listed in Table 82.

Table 82 – SM_GetPortConfig

Parameter name	.req	.cnf
Argument	M	
Port Number	M	
Result (+)		S(=)
Parameterlist		M
Result (-)		S(=)
Port Number		M
ErrorInfo		M

Argument
The service-specific parameters are transmitted in the argument.

Port Number
This parameter contains the port number

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ParameterList
This parameter contains the configured port and Device parameter of a Master port.

Parameter type: Record

Record Elements:

PortNumber
This parameter contains the port number.

TargetMode
This parameter indicates the operational mode

Permitted values: INACTIVE, DI, DO, CFGCOM, AUTOCOM (see Table 81)

RealBaudrate
This parameter indicates the actual transmission rate

Permitted values:
COM1 (transmission rate of COM1)
COM2 (transmission rate of COM2)
COM3 (transmission rate of COM3)

RealCycleTime
This parameter contains the real (actual) cycle time

RealRevision (RRID)
Data length: 1 octet for the protocol version (see B.1.5)

RealVendorID (RVID)
Data length: 2 octets

NOTE VendorIDs are assigned by the IO-Link community

RealDeviceID (RDID)
Data length: 3 octets

RealFunctionID (RFID)
Data length: 2 octets

RealSerialNumber (RSN)
Data length: up to 16 octets

Result (-):
This selection parameter indicates that the service failed

Port Number
This parameter contains the port number

ErrorInfo
This parameter contains error information

Permitted values:
PARAMETER_CONFLICT (consistency of parameter set violated)

All parameters shall be set to "0" if there is no information available.

9.2.2.4 SM_PortMode

The SM_PortMode service is used to indicate changes or faults of the local communication mode. These shall be reported to the Master application. The parameters of the service primitives are listed in Table 83.

Table 83 – SM_PortMode

Parameter name	.ind

Argument
  Port Number
  Mode
	M
	M
	M

Argument
The service-specific parameters are transmitted in the argument.

Port Number
This parameter contains the port number

Mode
Permitted values:
INACTIVE (Communication disabled, no DI, no DO)
DI (Port in digital input mode (SIO))
DO (Port in digital output mode (SIO))
COMREADY (Communication established and inspection successful)
SM_OPERATE (Port is ready to exchange Process Data)
COMLOST (Communication failed, new wake-up procedure required)
REVISION_FAULT (Incompatible protocol revision)
COMP_FAULT (Incompatible Device or Legacy-Device according to the Inspection Level)
SERNUM_FAULT (Mismatching SerialNumber according to the InspectionLevel)
CYCTIME_FAULT (Device does not support the configured cycle time)

9.2.2.5 SM_Operate

The SM_Operate service prompts System Management to calculate the MasterCycleTime for the ports if the service is acknowledged positively with Result (+). This service is effective at the indicated port. The parameters of the service primitives are listed in Table 84.

Table 84 – SM_Operate

Parameter name	.req	.cnf
Argument Port number	M M	
Result (+)		S

Result (-)
  Port Number
  ErrorInfo
		S
		M
		M

Argument
The service-specific parameters are transmitted in the argument.

Port Number

This parameter contains the port number

Result (+):
This selection parameter indicates that the service has been executed successfully.

Result (-):
This selection parameter indicates that the service failed.

Port Number
This parameter contains the port number

ErrorInfo
This parameter contains error information.

Permitted values:
STATE_CONFLICT (service unavailable within current state, for example if port is already in OPERATE state)

9.2.3 SM Master protocol

9.2.3.1 Overview

Due to the comprehensive configuration, parameterization, and operational features of SDCI the description of the behavior with the help of state diagrams becomes rather complex. Similar to the DL state machines clause 9.2.3 uses the possibility of submachines within the main state machines.

Comprehensive compatibility check methods are performed within the submachine states. These methods are indicated by "do *method*" fields within the state graphs, for example in

Figure 72.

The corresponding decision logic is demonstrated via activity diagrams (see Figure 73, Figure 74, Figure 75, and Figure 78).

9.2.3.2 SM Master state machine

Figure 71 shows the main state machine of the System Mangement Master.

Two submachines for the compatibility and serial number check are specified in subsequent sections.

In case of communication disruption the System Management is informed via the service DL_Mode (COMLOST).

Only the SM_SetPortConfig service allows reconfiguration of a port.

The service SM_Operate causes no effect in any state except in state "wait_4".

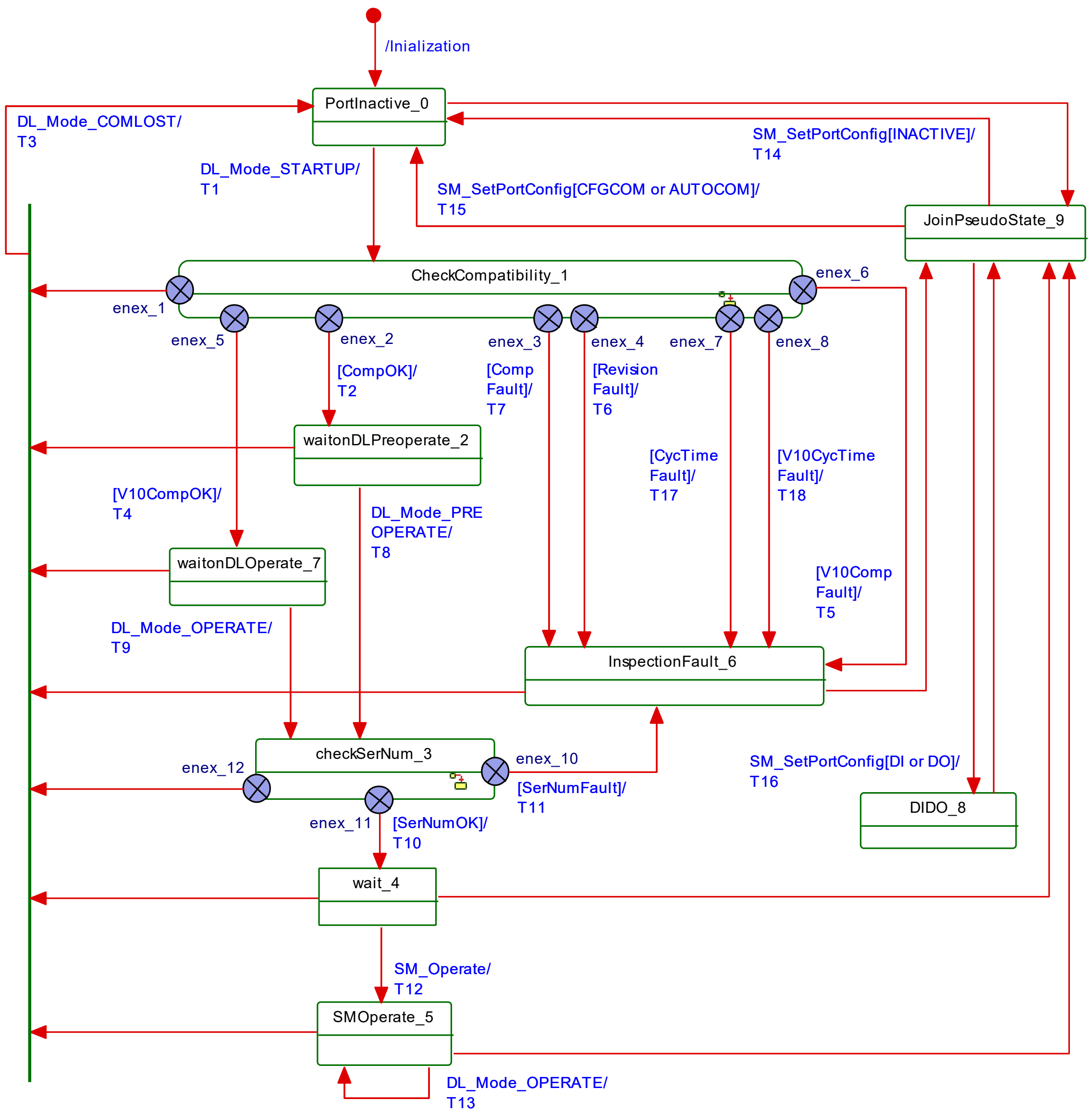


Figure 71 – Main state machine of the Master System Management

Table 85 shows the state transition tables of the Master System Management.

Table 85 – State transition tables of the Master System Management

STATE NAME	STATE DESCRIPTION
PortInactive_0	No communication
CheckCompatibility_1	Port is started and revision and Device compatibility is checked. See Figure 72.
waitonDLPreoperate_2	Wait until the PREOPERATE state is established and all the On-Request handlers are started. Port is ready to communicate.
checkSerNum_3	SerialNumber is checked depending on the InspectionLevel (IL). See Figure 77.
wait_4	Port is ready to communicate and waits on service SM_Operate from CM.
SM Operate_5	Port is in state OPERATE and performs cyclic Process Data exchange.
InspectionFault_6	Port is ready to communicate. However, cyclic Process Data exchange cannot be performed due to incompatibilities.
waitonDLOperate_7	Wait on the requested state OPERATE in case the Master is connected to a legacy Device. The SerialNumber can be read thereafter.

STATE NAME	STATE DESCRIPTION
DIDO_8	Port will be switched into the DI or DO mode (SIO, no communication).
JoinPseudoState_9	This pseudo state is used instead of a UML join bar. It allows execution of individual SM_SetPortConfig services depending on the system status (INACTIVE, CFGCOM, AUTOCOM, DI, or DO)

2425

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	CompRetry = 0
T2	1	2	DL_SetMode.req (PREOPERATE, ValueList)
T3	1,2,3,4,5, 6,7	0	DL_SetMode.req (INACTIVE) and SM_PortMode.ind (COMLOST) due to communication fault
T4	1	7	DL_SetMode.req (OPERATE, ValueList)
T5	1	6	SM_PortMode.ind (COMP_FAULT) triggering SMI_PortEvent(0x1802) or SMI_PortEvent(0x1803) depending on mismatch reason, DL_SetMode.req (OPERATE, ValueList)
T6	1	6	SM_PortMode.ind (REVISION_FAULT)
T7	1	6	SM_PortMode.ind (COMP_FAULT) triggering SMI_PortEvent(0x1802) or SMI_PortEvent(0x1803) depending on mismatch reason, DL_SetMode.req (PREOPERATE, ValueList)
T8	2	3	-
T9	7	3	-
T10	3	4	SM_PortMode.ind (COMREADY)
T11	3	6	SM_PortMode.ind (SERNUM_FAULT)
T12	4	5	DL_SetMode.req (OPERATE, ValueList)
T13	5	5	-
T14	0,4,5,6,8	0	SM_PortMode.ind (INACTIVE), DL_SetMode.req (INACTIVE)
T15	0,4,5,6,8	0	DL_SetMode.req (STARTUP, ValueList), PL_SetMode.req (SDCI)
T16	0,4,5,6,8	8	PL_SetMode.req (SIO), SM_PortMode.ind (DI or DO), DL_SetMode.req (INACTIVE)
T17	1	6	SM_PortMode.ind (CYCTIME_FAULT), DL_SetMode.req (PREOPERATE, ValueList)
T18	1	6	SM_PortMode.ind (CYCTIME_FAULT), DL_SetMode.req (OPERATE, ValueList), ValueList.M-sequenceTime = MinCycleTime of Device

2426

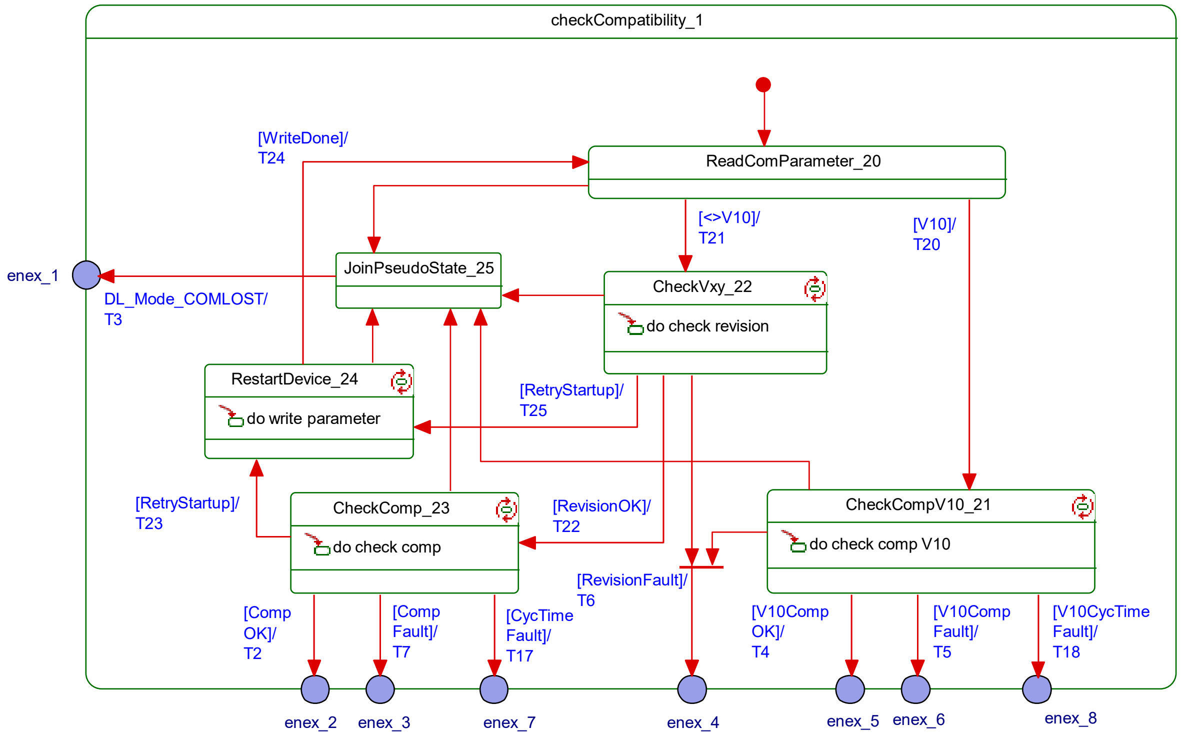
INTERNAL ITEMS	TYPE	DEFINITION
CompOK	Bool	See Figure 75
CompFault	Bool	See Figure 75; error variable COMP_FAULT
CycTimeFault	Bool	See Figure 75; error variable CYCTIME_FAULT
RevisionFault	Bool	See Figure 73; error variable REVISION_FAULT
SerNumFault	Bool	See Figure 78; error variable SERNUM_FAULT
SerNumOK	Bool	See Figure 78
V10CompFault	Bool	See Figure 74; error variable COMP_FAULT
V10CompOK	Bool	See Figure 74
V10CycTimeFault	Bool	See Figure 74; error variable CYCTIME_FAULT
INACTIVE	Variable	A target mode in service SM_SetPortConfig
CFGCOM, AUTOCOM	Variables	Target Modes in service SM_SetPortConfig

2427

2428 ### 9.2.3.3 SM Master submachine "Check Compatibility"

2429

2430 Figure 72 shows the SM Master submachine checkCompatibility_1.



2431

2432 **Figure 72 – SM Master submachine CheckCompatibility_1**

2433 Table 86 shows the state transition tables of the Master submachine checkCompatibility_1.

2434 **Table 86 – State transition tables of the Master submachine CheckCompatibility_1**

STATE NAME	STATE DESCRIPTION
ReadComParameter_20	Acquires communication parameters from Direct Parameter Page 1 (0x02 to 0x06) via service DL_Read (see Table B.1).
CheckCompV10_21	Acquires identification parameters from Direct Parameter Page 1 (0x07 to 0x0D) via service DL_Read (see Table B.1). The configured InspectionLevel (IL) defines the decision logic of the subsequent compatibility check "CheckCompV10" with parameters RVID, RDID, and RFID according to Figure 74.
CheckVxy_22	A check is performed whether the configured revision (CRID) matches the real (actual) revision (RRID) according to Figure 73.
CheckComp_23	Acquires identification parameters from Direct Parameter Page 1 (0x07 to 0x0D) via service DL_Read (see Table B.1). The configured InspectionLevel (IL) defines the decision logic of the subsequent compatibility check "CheckComp" according to Figure 75.
RestartDevice_24	Writes the configured protocol revision (CRID) and configured DeviceID (CDID) into the Device depending on the Target Mode of communication CFGCOM or AUTOCOM (see Table 81) according to Figure 76.
JoinPseudoState_25	This pseudo state is used instead of a UML join bar. No guards involved.

2435

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T20	20	21	-
T21	20	22	DL_Write (0x00, MCmd_MASTERIDENT), see Table B.2
T22	22	23	-
T23	23	24	-
T24	24	20	-
T25	22	24	CompRetry = CompRetry +1

2436

INTERNAL ITEMS	TYPE	DEFINITION
CompOK	Bool	See Figure 75
CompFault	Bool	See Figure 75; error variable COMP_FAULT
RevisionFault	Bool	See Figure 73; error variable REVISION_FAULT
RevisionOK	Bool	See Figure 73
SerNumFault	Bool	See Figure 78; error variable SERNUM_FAULT
SerNumOK	Bool	See Figure 78
V10	Bool	Real protocol revision of connected Device is a legacy version (V1.0, see B.1.5)
<>V10	Bool	Real protocol revision of connected Device is in accordance with this standard
V10CompFault	Bool	See Figure 74; error variable COMP_FAULT
V10CompOK	Bool	See Figure 74
RetryStartup	Bool	See Figure 73 and Figure 75
CompRetry	Variable	Internal counter
WriteDone	Bool	Finalization of the restart service sequence
MCmd_XXXXXXX	Call	See Table 45

Some states contain complex logic to deal with the compatibility and validity checks. Figure 73 to Figure 76 are demonstrating the context.

Figure 73 shows the decision logic for the protocol revision check in state "CheckVxy". In case of configured Devices the following rule applies: if the configured revision (CRID) and the real revision (RRID) do not match, the CRID will be transmitted to the Device. If the Device does not accept, the Master returns an indication via the SM_PortMode service with REV_FAULT.

In case of not configured Devices the operational mode AUTOCOM shall be used. See 9.2.2.2 and 9.2.2.3 for the parameter name abbreviations.

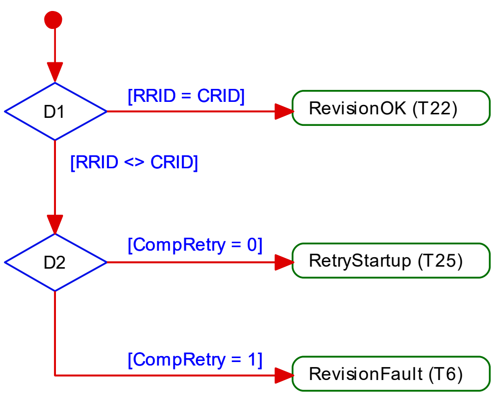


Figure 73 – Activity for state "CheckVxy"

Figure 74 shows the decision logic for the legacy compatibility check in state "CheckCompV10".

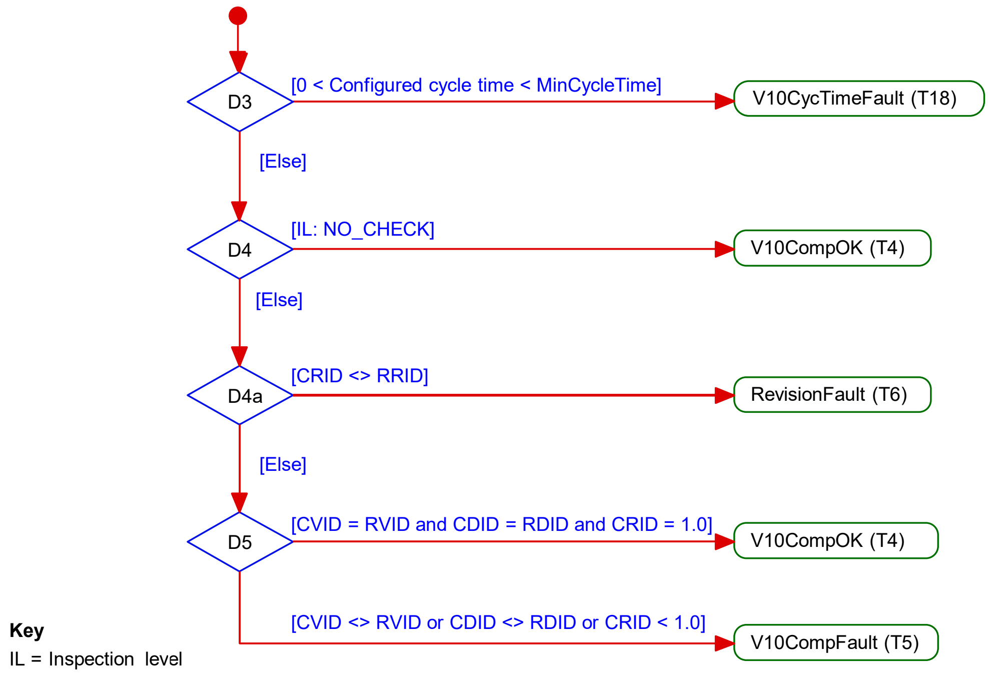


Figure 74 – Activity for state "CheckCompV10"

Figure 75 shows the decision logic for the compatibility check in state "CheckComp".

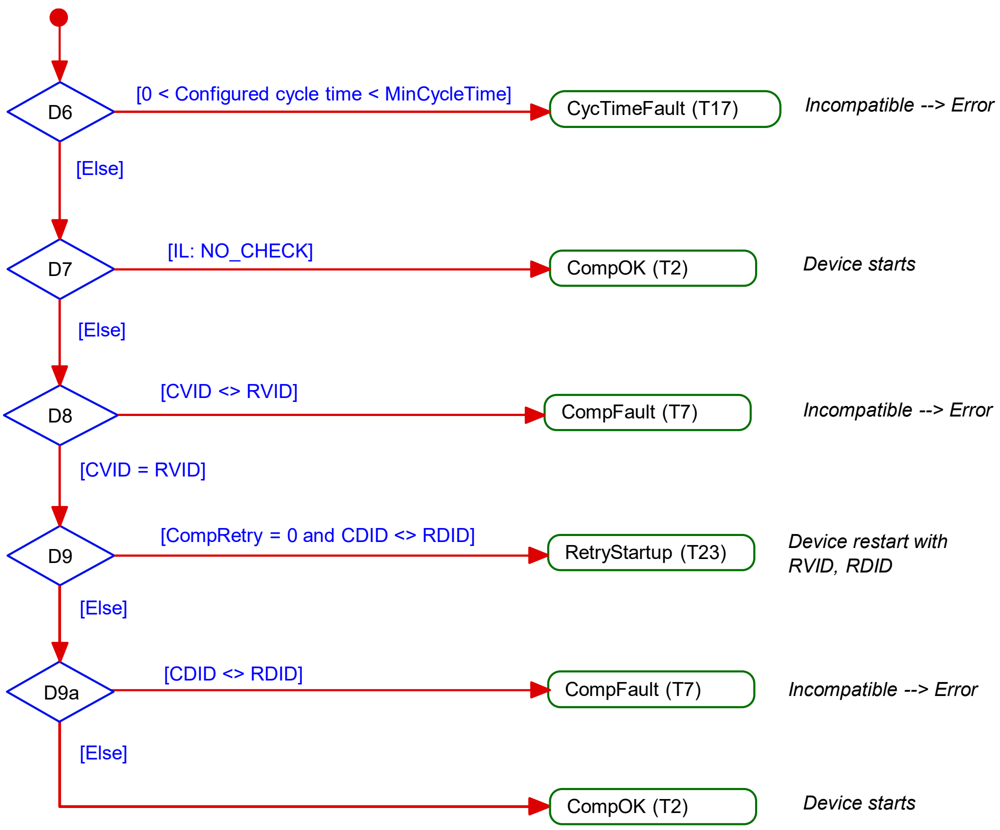


Figure 75 – Activity for state "CheckComp"

Figure 76 shows the activity (write parameter) in state "RestartDevice".

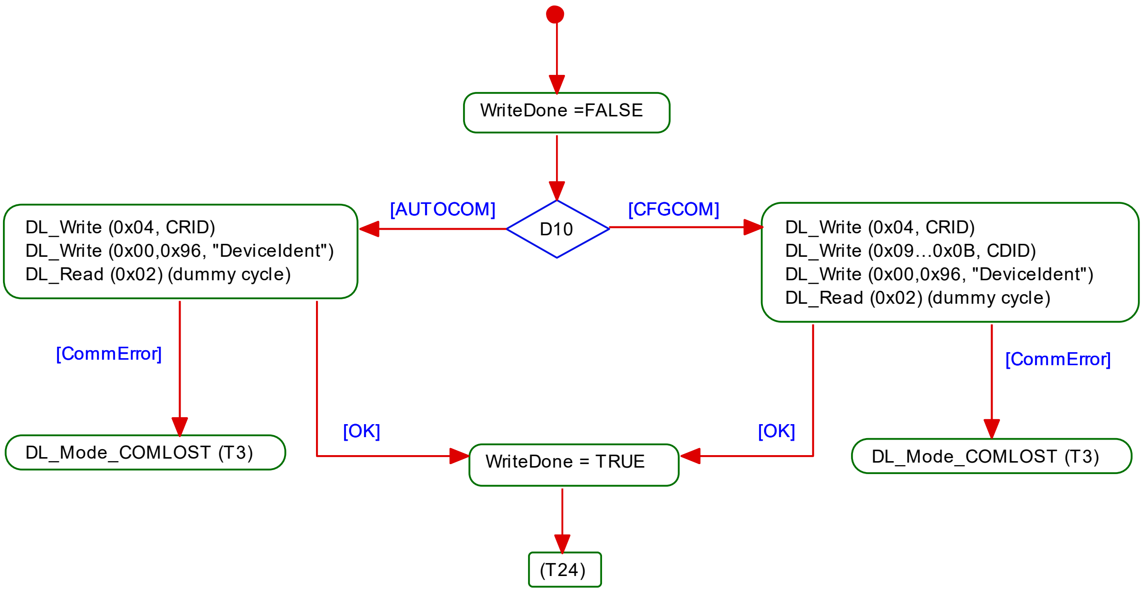


Figure 76 – Activity (write parameter) in state "RestartDevice"

9.2.3.4 SM Master submachine "Check serial number"

Figure 77 shows the SM Master submachine "checkSerNum_3". State CheckSernum_31 can be skipped (option).

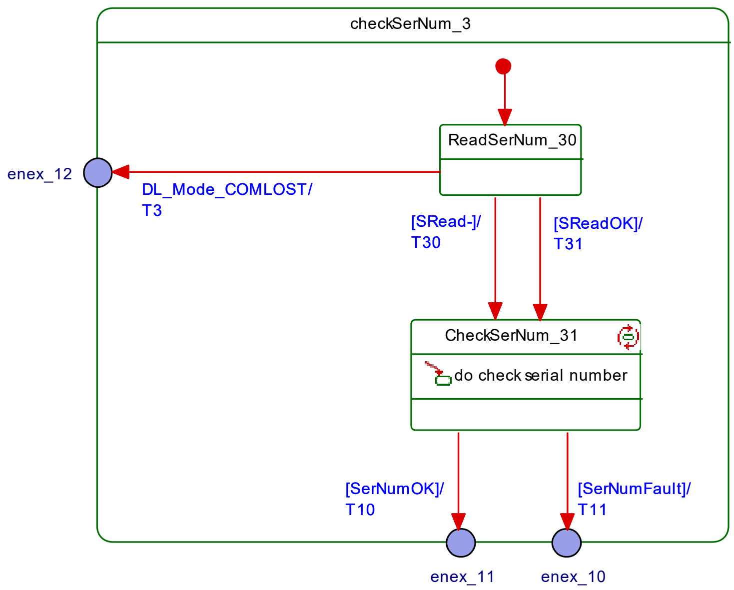


Figure 77 – SM Master submachine checkSerNum_3

Table 87 shows the state transition tables of the Master submachine checkSerNum_3

Table 87 – State transition tables of the Master submachine checkSerNum_3

STATE NAME	STATE DESCRIPTION
ReadSerNum_30	Acquires the SerialNumber from the Device via AL_Read.req (Index: 0x0015). A positive response (AL_Read(+)) leads to SReadOK = true. A negative response (AL_Read(-)) leads to SRead- = true.
CheckSerNum_31	Optional: SerialNumber checking skipped or checked correctly.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T30	40	41	–
T31	40	41	–

INTERNAL ITEMS	TYPE	DEFINITION
SRead-	Bool	Negative response of service AL_Read (Index 0x0015)
SReadOK	Bool	SerialNumber read correctly
SerNumOK	Bool	See Figure 78
SerNumFault	Bool	See Figure 78

Figure 78 shows the decision logic (activity) for the state CheckSerNum_31.

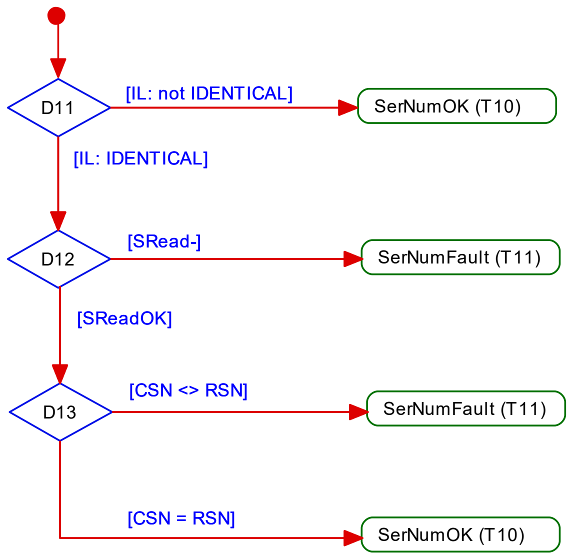


Figure 78 – Activity (check SerialNumber) for state CheckSerNum_31

9.2.3.5 Rules for the usage of M-sequence types

The System Management is responsible for setting up the correct M-sequence types. This occurs after the check compatibility actions (transition to PREOPERATE) and before the transition to OPERATE.

Different M-sequence types shall be used within the different operational states (see A.2.6). For example, when switching to the OPERATE state the M-sequence type relevant for cyclic operation shall be used. The M-sequence type to be used in operational state OPERATE is determined by the size of the input and output Process Data. The available M-sequence types in the three modes STARTUP, PREOPERATE, and OPERATE and the corresponding coding of the parameter M-sequenceCapability are specified in A.2.6. The input and output data formats shall be acquired from the connected Device in order to adjust the M-sequence type. It is mandatory for a Master to implement all the specified M-sequence types in A.2.6.

9.3 System Management of the Device

9.3.1 Overview

Figure 79 provides an overview of the structure and services of the Device System Management.

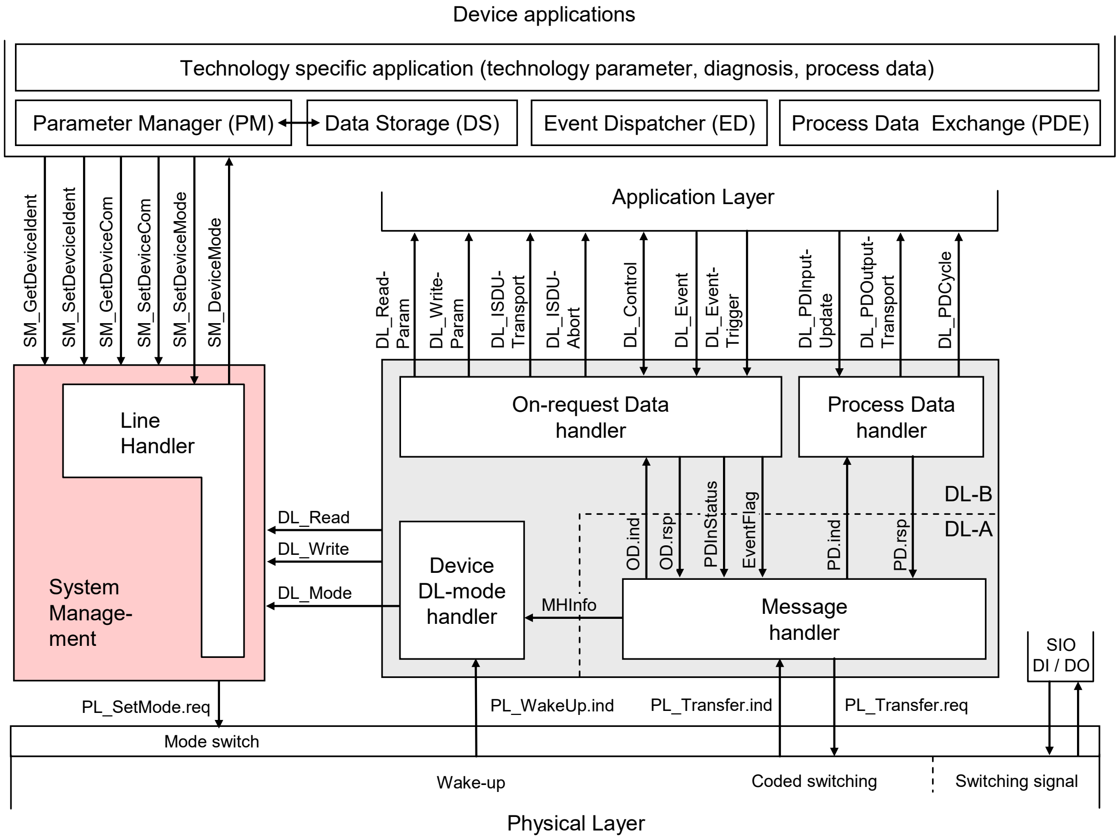


Figure 79 – Structure and services of the System Management (Device)

The System Management (SM) of the Device provides the central controlling instance via the Line Handler through all the phases of initialization, default state (SIO), communication startup, communication, and fallback to SIO mode.

The Device SM interacts with the PL to establish the necessary line driver and receiver adjustments (see Figure 16), with the DL to get the necessary information from the Master (wake-up, transmission rates, a.o.) and with the Device applications to ensure the Device identity and compatibility (communication and identification parameters).

The transitions between the line handler states (see Figure 81) are initiated by the Master port activities (wake-up and communication) and triggered through the Device Data Link Layer via the DL_Mode indications and DL_Write requests (commands).

The SM provides the Device communication and identification parameters through the Device applications interface.

The sequence chart in Figure 80 demonstrates a typical Device sequence from initialization to default SIO mode and via wake-up request from the Master to final communication. The sequence chart is complemented by the use case of a communication error such as T_{DSIO} expired, or communication fault, or a request from Master such as Fallback (caused by Event).

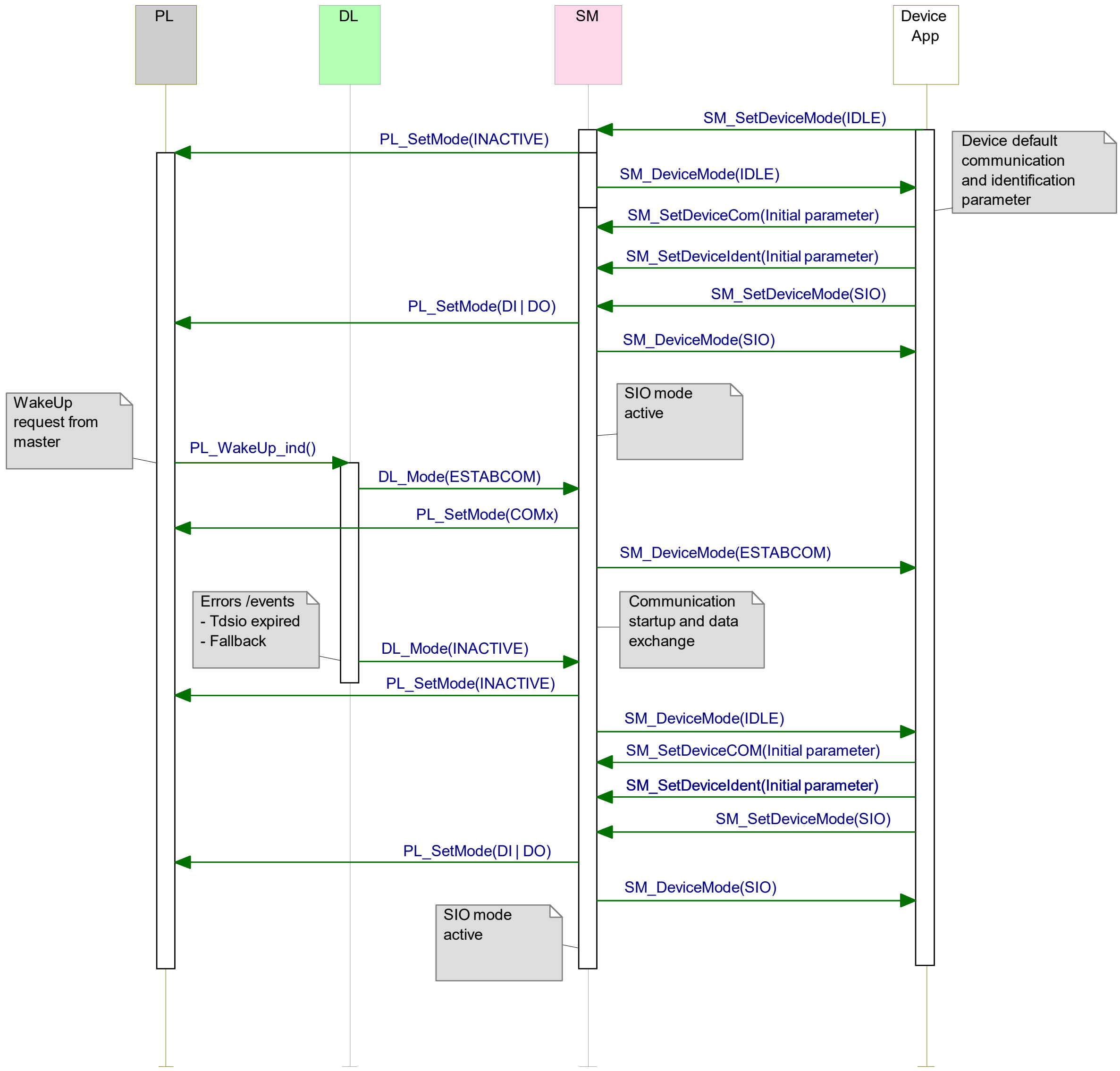


Figure 80 – Sequence chart of the use case "INACTIVE – SIO – SDCI – SIO"

The SM services shown in Figure 80 are specified in 9.3.2.

9.3.2 SM Device services

9.3.2.1 Overview

Subclause 9.3.2 describes the services the Device System Management provides to its applications as shown in Figure 79.

Table 88 lists the assignment of the Device to its role as initiator or receiver for the individual System Management service.

Table 88 – SM services within the Device

Service name	Device
SM_SetDeviceCom	R
SM_GetDeviceCom	R
SM_SetDeviceIdent	R
SM_GetDeviceIdent	R
SM_SetDeviceMode	R

Service name	Device
SM_DeviceMode	I
Key (see 3.3.4) I Initiator of service R Receiver (Responder) of service	

9.3.2.2 SM_SetDeviceCom

The SM_SetDeviceCom service is used to configure the communication properties supported by the Device in the System Management. The parameters of the service primitives are listed in Table 89.

Table 89 – SM_SetDeviceCom

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

Argument
The service-specific parameters are transmitted in the argument.

ParameterList
This parameter contains the configured communication and identification parameters for a Device.

Parameter type: Record

Record Elements:

SupportedSIOMode
This parameter indicates the SIO mode supported by the Device.

Permitted values:
INACTIVE (C/Q line in high impedance)
DI (C/Q line in digital input mode)
DO (C/Q line in digital output mode)

SupportedTransmissionrate
This parameter indicates the transmission rate supported by the Device.

Permitted values:
COM1 (transmission rate of COM1)
COM2 (transmission rate of COM2)
COM3 (transmission rate of COM3)

MinCycleTime
This parameter contains the minimum cycle time supported by the Device (see B.1.3).

M-sequence Capability
This parameter indicates the capabilities supported by the Device (see B.1.4):
- ISDU support
- OPERATE M-sequence types
- PREOPERATE M-sequence types

RevisionID (RID)
This parameter contains the protocol revision (see B.1.5) supported by the Device.

ProcessDataIn
This parameter contains the length of PD to be sent to the Master (see B.1.6).

ProcessDataOut
This parameter contains the length of PD to be sent by the Master (see B.1.7).

Result (+):
This selection parameter indicates that the service has been executed successfully.

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
PARAMETER_CONFLICT (consistency of parameter set violated)

9.3.2.3 SM_GetDeviceCom

The SM_GetDeviceCom service is used to read the current communication properties from the System Management. The parameters of the service primitives are listed in Table 90.

Table 90 – SM_GetDeviceCom

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

Argument
The service-specific parameters are transmitted in the argument.

Result (+):
This selection parameter indicates that the service has been executed successfully.

ParameterList
This parameter contains the configured communication parameter for a Device.

Parameter type: Record

Record Elements:

CurrentMode
This parameter indicates the current SIO or Communication Mode by the Device.

Permitted values:
INACTIVE (C/Q line in high impedance)
DI (C/Q line in digital input mode)
DO (C/Q line in digital output mode)
COM1 (transmission rate of COM1)
COM2 (transmission rate of COM2)
COM3 (transmission rate of COM3)

MasterCycleTime
This parameter contains the MasterCycleTime to be set by the Master System Management (see B.1.3). This parameter is only valid in the state SM_Operate.

M-sequence Capability
This parameter indicates the current M-sequence capabilities configured in the System Management of the Device (see B.1.4):
- ISDU support
- OPERATE M-sequence types
- PREOPERATE M-sequence types

RevisionID (RID)
This parameter contains the current protocol revision (see B.1.5) within the System Management of the Device.

ProcessDataIn

This parameter contains the current length of PD to be sent to the Master (see B.1.6).

ProcessDataOut
This parameter contains the current length of PD to be sent by the Master (see B.1.7).

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
STATE_CONFLICT (service unavailable within current state)

9.3.2.4 SM_SetDeviceIdent

The SM_SetDeviceIdent service is used to configure the Device identification data in the System Management. The parameters of the service primitives are listed in Table 91.

Table 91 – SM_SetDeviceIdent

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

Argument
The service-specific parameters are transmitted in the argument.

ParameterList
This parameter contains the configured identification parameter for a Device.

Parameter type: Record

Record Elements:

VendorID (VID)
This parameter contains the VendorID assigned to a Device (see B.1.8)

Data length: 2 octets

DeviceID (DID)
This parameter contains one of the assigned DeviceIDs (see B.1.9)

Data length: 3 octets

FunctionID (FID)
This parameter contains one of the assigned FunctionIDs (see B.1.10).

Data length: 2 octets

Result (+):
This selection parameter indicates that the service has been executed successfully.

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
STATE_CONFLICT (service unavailable within current state)
PARAMETER_CONFLICT (consistency of parameter set violated)

9.3.2.5 SM_GetDeviceIdent

The SM_GetDeviceIdent service is used to read the Device identification parameter from the System Management. The parameters of the service primitives are listed in Table 92.

Table 92 – SM_GetDeviceIdent

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

Argument
The service-specific parameters are transmitted in the argument.

Result (+):
This selection parameter indicates that the service has been executed successfully.

ParameterList
This parameter contains the configured identification parameters of the Device.

Parameter type: Record

Record Elements:

VendorID (VID)
This parameter contains the actual VendorID of the Device (see B.1.8)

Data length: 2 octets

DeviceID (DID)
This parameter contains the actual DeviceID of the Device (see B.1.9)

Data length: 3 octets

FunctionID (FID)
This parameter contains the actual FunctionID of the Device (see B.1.10).

Data length: 2 octets

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
STATE_CONFLICT (service unavailable within current state)

9.3.2.6 SM_SetDeviceMode

The SM_SetDeviceMode service is used to set the Device into a defined operational state during initialization. The parameters of the service primitives are listed in Table 93.

Table 93 – SM_SetDeviceMode

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

Argument
The service-specific parameters are transmitted in the argument.

Mode

Permitted values:
IDLE (Device changes to waiting for configuration)
SIO (Device changes to the mode defined in service "SM_SetDeviceCom")

Result (+):
This selection parameter indicates that the service has been executed successfully.

Result (-):
This selection parameter indicates that the service failed.

ErrorInfo
This parameter contains error information.

Permitted values:
STATE_CONFLICT (service unavailable within current state)

9.3.2.7 SM_DeviceMode

The SM_DeviceMode service is used to indicate changes of communication states to the Device application. The parameters of the service primitives are listed in Table 94.

Table 94 – SM_DeviceMode

Parameter name	.ind
Argument	M
Mode	M

Argument
The service-specific parameters are transmitted in the argument.

Mode
Permitted values:
IDLE (Device changed to waiting for configuration)
SIO (Device changed to the mode defined in service "SM_SetDeviceCom")
ESTABCOM (Device changed to the SM mode "SM_ComEstablish")
COM1 (Device changed to the COM1 mode)
COM2 (Device changed to the COM2 mode)
COM3 (Device changed to the COM3 mode)
STARTUP (Device changed to the STARTUP mode)
IDENT_STARTUP (Device changed to the SM mode "SM_IdentStartup")
IDENT_CHANGE (Device changed to the SM mode "SM_IdentCheck")
PREOPERATE (Device changed to the PREOPERATE mode)
OPERATE (Device changed to the OPERATE mode)

9.3.3 SM Device protocol

9.3.3.1 Overview

The behaviour of the Device is mainly driven by Master messages.

9.3.3.2 SM Device state machine

Figure 81 shows the SM line handler state machine of the Device. It is triggered by the DL_Mode handler and the Device application. It evaluates the different communication phases during startup and controls the line state of the Device.

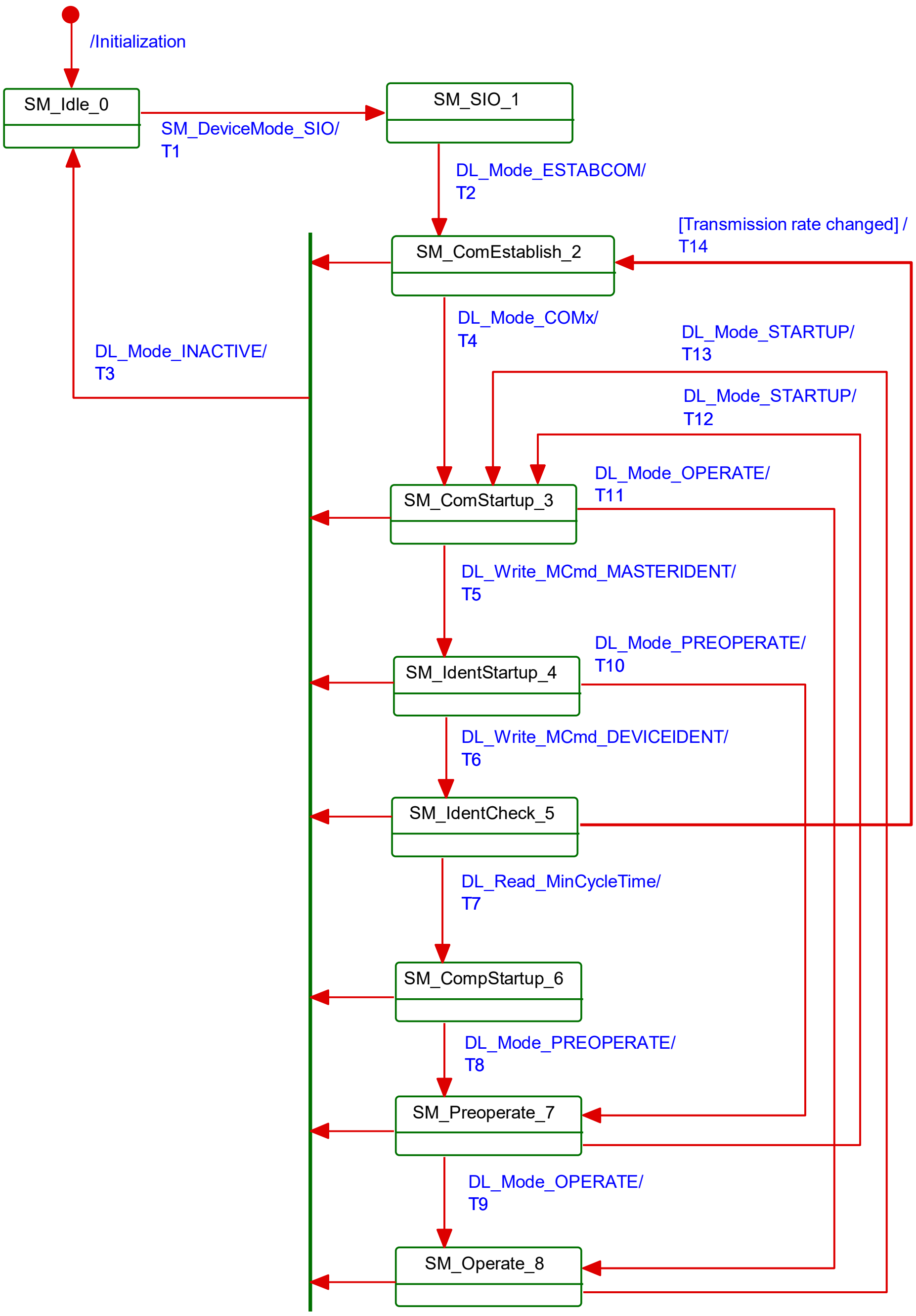


Figure 81 – State machine of the Device System Management

Table 95 specifies the individual states and the actions within the transitions.

Table 95 – State transition tables of the Device System Management

STATE NAME	STATE DESCRIPTION
SM_Idle_0	In SM_Idle the SM is waiting for configuration by the Device application and to be set to SIO mode. The state is left on receiving a SM_SetDeviceMode(SIO) request from the Device application The following sequence of services shall be executed between Device application and SM. Invoke SM_SetDeviceCom(initial parameter list) Invoke SM_SetDeviceIdent(VID, initial DID, FID)

STATE NAME	STATE DESCRIPTION
SM_SIO_1	In SM_SIO the SM Line Handler is remaining in the default SIO mode. The Physical Layer is set to the SIO mode characteristics defined by the Device application via the SetDeviceMode service. The state is left on receiving a DL_Mode(ESTABCOM) indication.
SM_ComEstablish_2	In SM_ComEstablish the SM is waiting for the communication to be established in the Data Link Layer. The state is left on receiving a DL_Mode(INACTIVE) or a DL_Mode(COMx) indication, where COMx may be any of COM1, COM2 or COM3.
SM_ComStartup_3	In SM_ComStartup the communication parameter (Direct Parameter page 1, addresses 0x02 to 0x06) are read by the Master SM via DL_Read requests. The state is left upon reception of a DL_Mode(INACTIVE), a DL_Mode(OPERATE) indication (legacy Master only), or a DL_Write(MCmd_MASTERIDENT) request (Master in accordance with this standard).
SM_IdentStartup_4	In SM_IdentStartup the identification data (VID, DID, FID) are read and verified by the Master. In case of incompatibilities the Master SM writes the supported SDCI Revision (RID) and configured DeviceID (DID) to the Device. The state is left upon reception of a DL_Mode(INACTIVE), a DL_Mode(PREOPERATE) indication (compatibility check passed), or a DL_Write(MCmd_DEVICEIDENT) request (new compatibility requested).
SM_IdentCheck_5	In SM_IdentCheck the SM waits for new initialization of communication and identification parameters. The state is left on receiving a DL_Mode(INACTIVE) indication, a DL_Read(Direct Parameter page 1, addresses 0x02 = "MinCycleTime") request, or the SM requires a switch of the transmission rate. Within this state the Device application shall check the RID and DID parameters from the SM and set these data to the supported values. Therefore the following sequence of services shall be executed between Device application and SM. Invoke SM_GetDeviceCom(configured RID, parameter list) Invoke SM_GetDeviceIdent(configured DID, parameter list) Invoke Device application checks and provides compatibility function and parameters Invoke SM_SetDeviceCom(new supported RID, new parameter list) Invoke SM_SetDeviceIdent(new supported DID, parameter list)
SM_CompStartup_6	In SM_CompatStartup the communication and identification data are reread and verified by the Master SM. The state is left on receiving a DL_Mode(INACTIVE) or a DL_Mode(PREOPERATE) indication.
SM_Preoperate_7	During SM_Preoperate the SerialNumber can be read and verified by the Master SM, as well as Data Storage and Device parameterization may be executed. The state is left on receiving a DL_Mode(INACTIVE), a DL_Mode(STARTUP) or a DL_Mode(OPERATE) indication.
SM_Operate_8	During SM_Operate the cyclic Process Data exchange and acyclic On-request Data transfer are active. The state is left on receiving a DL_Mode(INACTIVE) or a DL_Mode(STARTUP) indication.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	The Device is switched to the configured SIO mode by receiving the trigger SM_SetDeviceMode.req(SIO). Invoke PL_SetMode(DI|DO|INACTIVE) Invoke SM_DeviceMode(SIO)
T2	1	2	The Device is switched to the communication mode by receiving the trigger DL_Mode.ind(ESTABCOM). Invoke PL_SetMode(COMx) Invoke SM_DeviceMode(ESTABCOM)
T3	2,3,4,5,6, 7,8	0	The Device is switched to SM_Idle mode by receiving the trigger DL_Mode.ind(INACTIVE) . Invoke PL_SetMode(INACTIVE) Invoke SM_DeviceMode(IDLE)
T4	2	3	The Device application receives an indication on the baudrate with which the communication has been established in the DL triggered by DL_Mode.ind(COMx). Invoke SM_DeviceMode(COMx)
T5	3	4	The Device identification phase is entered by receiving the trigger DL_Write.ind(MCmd_MASTERIDENT). Invoke SM_DeviceMode(IDENTSTARTUP)
T6	4	5	The Device identity check phase is entered by receiving the trigger DL_Write.ind(MCmd_DEVICEIDENT). Invoke SM_DeviceMode(IDENTCHANGE)

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T7	5	6	The Device compatibility startup phase is entered by receiving the trigger DL_Read.ind(Direct Parameter page 1, address 0x02 = "MinCycleTime").
T8	6	7	The Device's preoperate phase is entered by receiving the trigger DL_Mode.ind(PREOPERATE). Invoke SM_DeviceMode(PREOPERATE)
T9	7	8	The Device's operate phase is entered by receiving the trigger DL_Mode.ind(OPERATE). Invoke SM_DeviceMode(OPERATE)
T10	4	7	The Device's preoperate phase is entered by receiving the trigger DL_Mode.ind(PREOPERATE). Invoke SM_DeviceMode(PREOPERATE)
T11	3	8	The Device's operate phase is entered by receiving the trigger DL_Mode.ind(OPERATE). Invoke SM_DeviceMode(OPERATE)
T12	7	3	The Device's communication startup phase is entered by receiving the trigger DL_Mode.ind(STARTUP). Invoke SM_DeviceMode(STARTUP)
T13	8	3	The Device's communication startup phase is entered by receiving the trigger DL_Mode.ind(STARTUP). Invoke SM_DeviceMode(STARTUP)
T14	5	2	The requested Device identification requires a change of the transmission rate. Stop communication by changing the current transmission rate. Invoke PL_SetMode(COMx) Invoke SM_DeviceMode(ESTABCOM)

INTERNAL ITEMS	TYPE	DEFINITION
COMx	Variable	Any of COM1, COM2, or COM3 transmission rates
DL_Write_MCmd_xxx	Service	DL Service writes MasterCommands (xxx = values out of Table B.2)

Figure 82 shows a typical sequence chart for the SM communication startup of a Device matching the Master port configuration settings (regular startup).

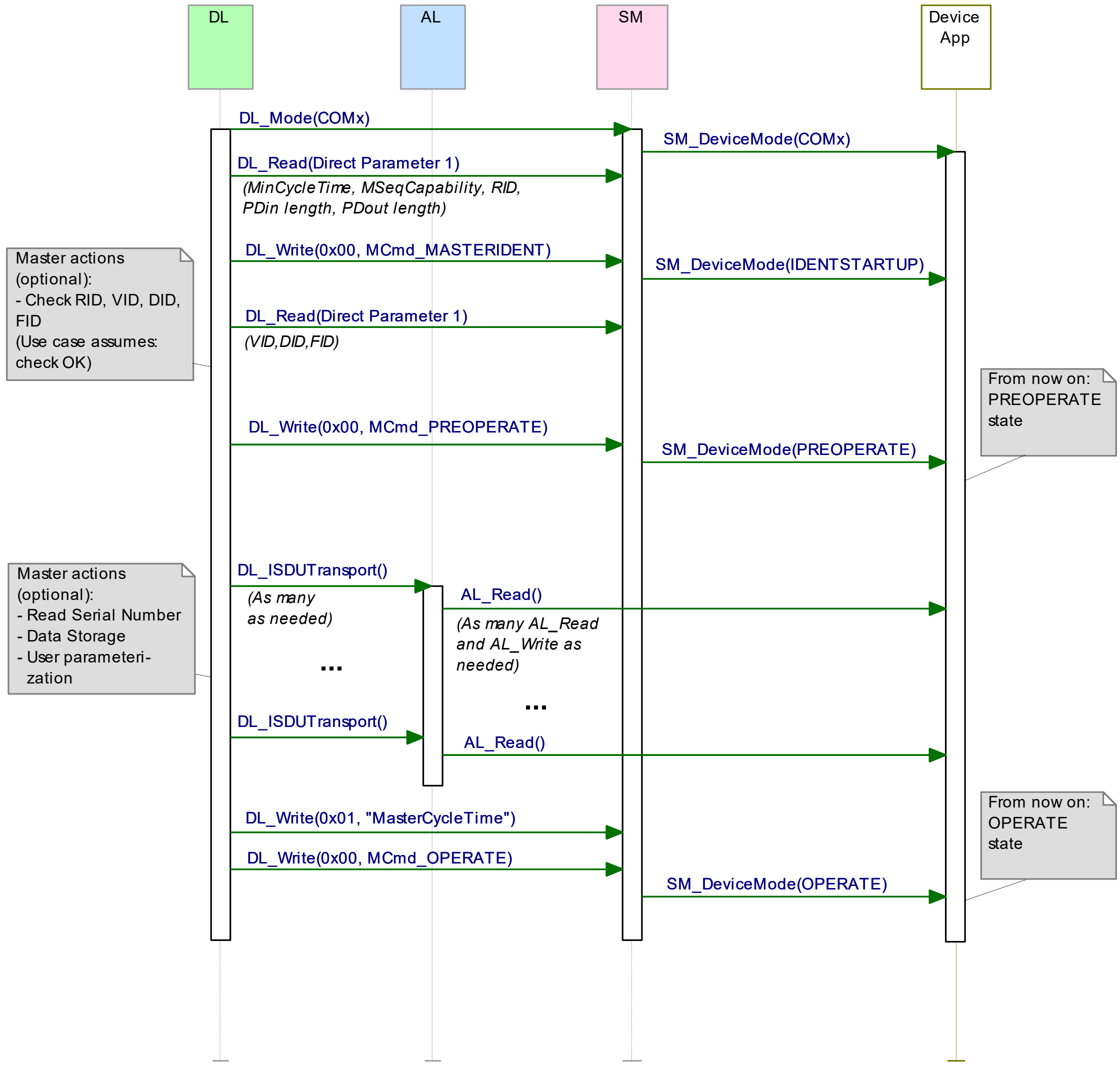


Figure 82 – Sequence chart of a regular Device startup

Figure 83 shows a typical sequence chart for the SM communication startup of a Device not matching the Master port configuration settings (compatibility mode). In this mode, the Master tries to overwrite the Device's communication and identification parameters to achieve a compatible and a workable mode.

The sequence chart in Figure 83 shows only the actions until the PREOPERATE state. The remaining actions until the OPERATE state can be taken from Figure 82.

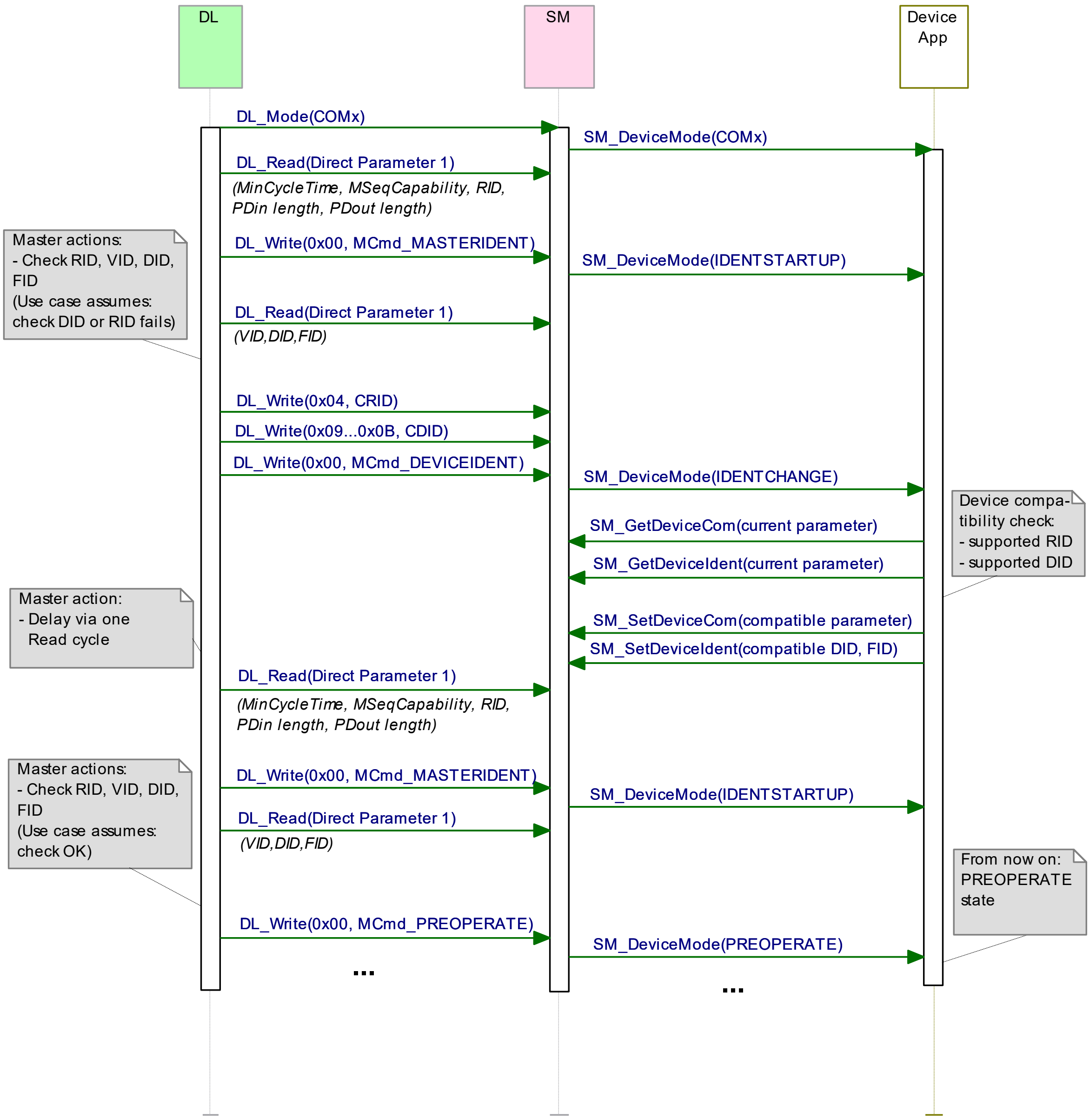
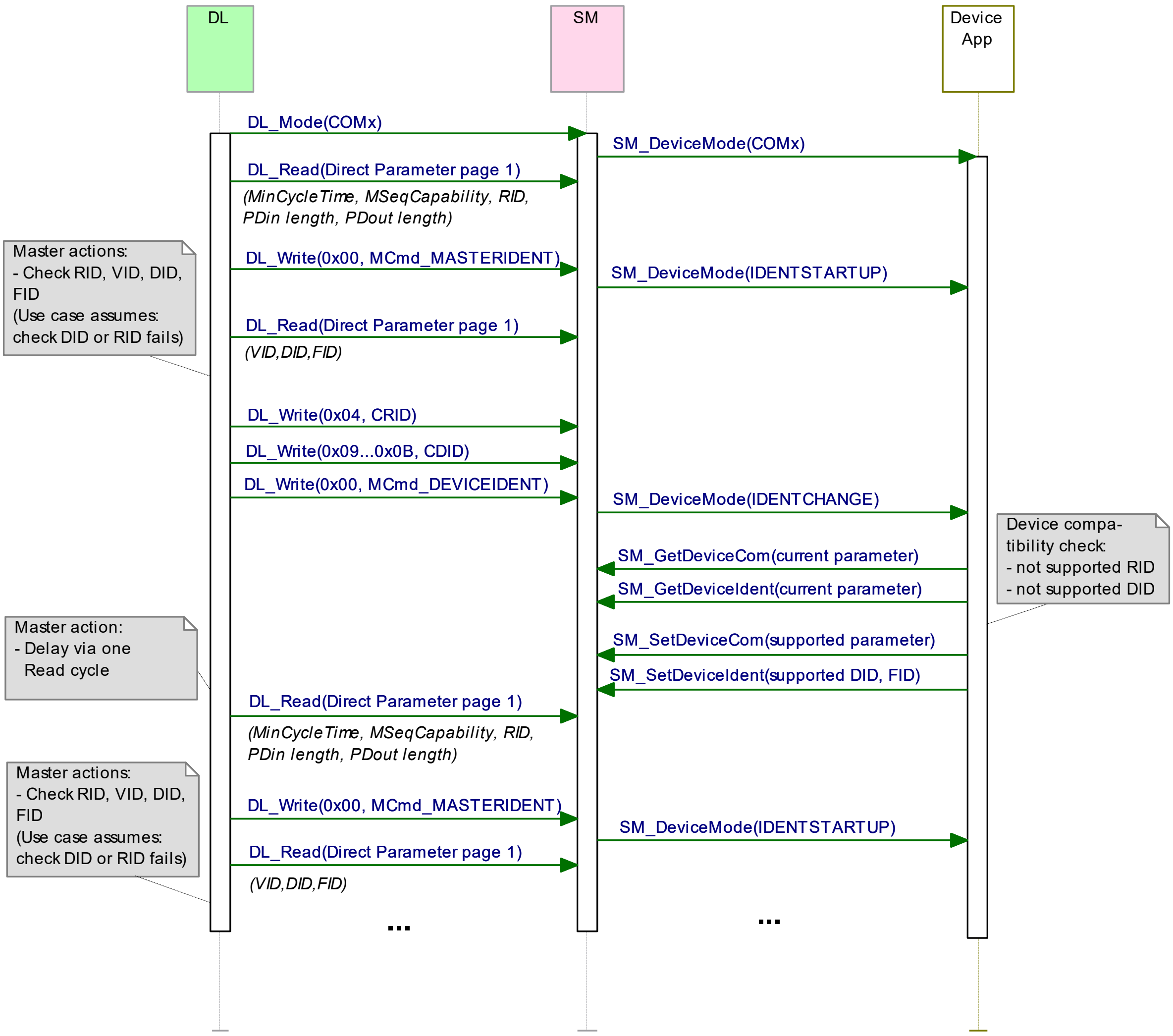


Figure 83 – Sequence chart of a Device startup in compatibility mode

Figure 84 shows a typical sequence chart for the SM communication startup of a Device not matching the Master port configuration settings. The System Management of the Master tries to reconfigure the Device with alternative Device communication and identification parameters (compatibility mode). In this use case, the alternative parameters are assumed to be incompatible.



2735

2736 **Figure 84 – Sequence chart of a Device startup when compatibility fails**

2737

2738

10 Device

10.1 Overview

Figure 85 provides an overview of the complete structure and services of a Device.

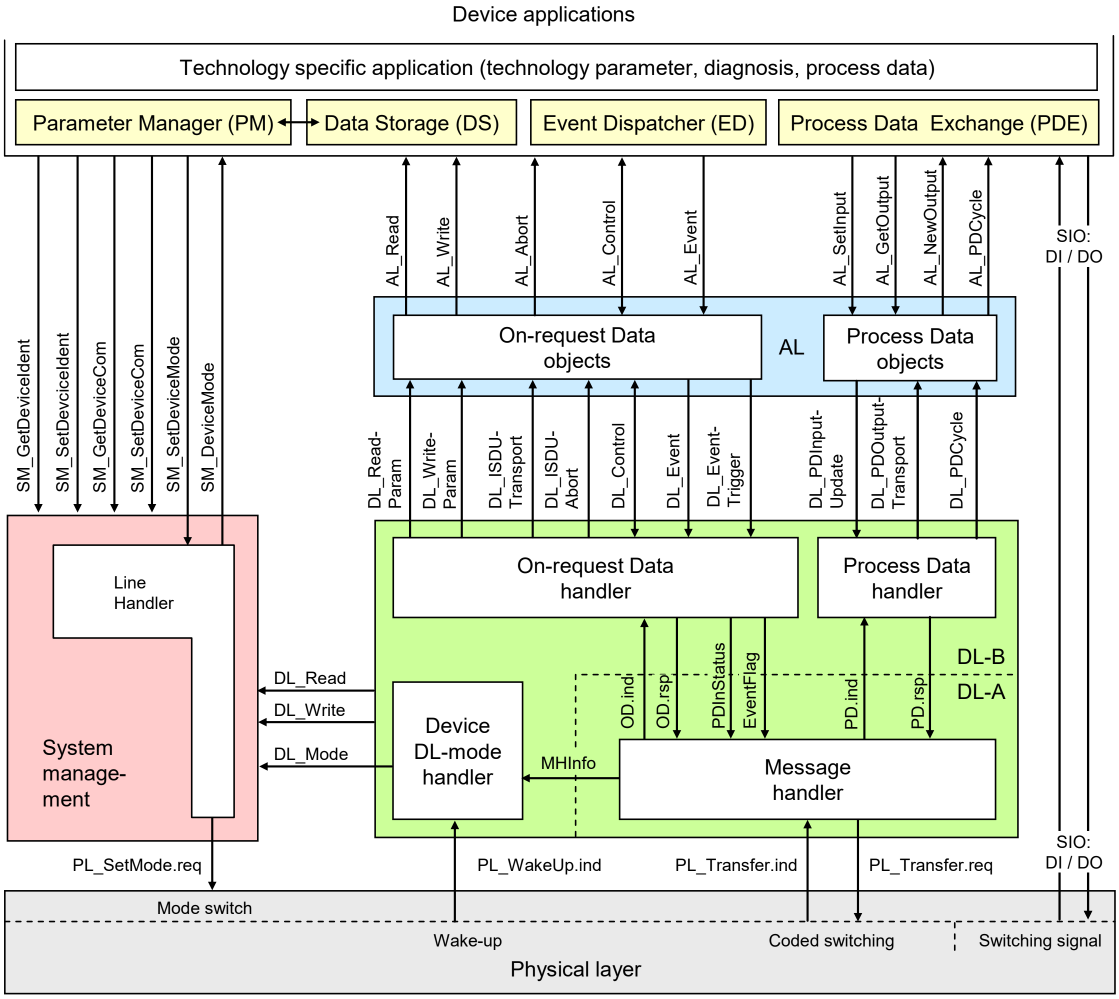


Figure 85 – Structure and services of a Device

The Device applications comprise first the technology specific application consisting of the transducer with its technology parameters, its diagnosis information, and its Process Data. The common Device applications comprise:

- Parameter Manager (PM), dealing with compatibility and correctness checking of complete sets of technology (vendor) specific and common system parameters (see 10.3);
- Data Storage (DS) mechanism, which optionally uploads or downloads parameters to the Master (see 10.4);
- Event Dispatcher (ED), supervising states and conveying diagnosis information such as notifications, warnings, errors, and Device requests as peripheral initiatives (see 10.5);
- Process Data Exchange (PDE) unit, conditioning the data structures for transmission in case of a sensor or preparing the received data structures for signal generation. It also controls the operational states to ensure the validity of Process Data (see 10.2).

These Device applications provide standard methods/functions and parameters common to all Devices, and Device specific functions and parameters, all specified within Clause 10.

10.2 Process Data Exchange (PDE)

The Process Data Exchange unit cyclically transmits and receives Process Data without interference from the On-request Data (parameters, commands, and Events).

An actuator (output Process Data) shall observe the cyclic transmission and enter a default appropriate state, for example keep last value, stop, or de-energize, whenever the data transmission is interrupted (see 7.3.3.5 and 10.8.3). The actuator shall wait on the MasterCommand "ProcessDataOutputOperate" (see Table B.2, output Process Data "valid") prior to regular operation after restart in case of an interruption.

Within cyclic data exchange, an actuator (output Process Data) receives a Master-Command "DeviceOperate", whenever the output Process Data are invalid and a Master-Command "ProcessDataOutputOperate", whenever they become valid again (see Table B.2).

There is no need for a sensor Device (input Process Data) to monitor the cyclic data exchange. However, if the Device is not able to guarantee valid Process Data, the PD status "Process Data invalid" (see A.1.5) shall be signaled to the Master application.

10.3 Parameter Manager (PM)

10.3.1 General

A Device can be parameterized via two basic methods using the Direct Parameters or the Index memory space accessible with the help of ISDUs (see Figure 6).

Mandatory for all Devices are the so-called Direct Parameters in page 1. This page 1 contains common communication and identification parameters (see B.1).

Direct Parameter page 2 optionally offers space for a maximum of 16 octets of technology (vendor) specific parameters for Devices requiring not more than this limited number and with small system footprint (ISDU communication not implemented, easier fieldbus handling possible but with less comfort). Access to the Direct Parameter page 2 is performed via AL_Read and AL_Write (see 10.8.5).

The transmission of parameters to and from the spacious Index memory can be performed in two ways: single parameter by single parameter or as a block of parameters. Single parameter transmission as specified in 10.3.4 is secured via several checks and confirmation of the transmitted parameter. A negative acknowledgment contains an appropriate error description and the parameter is not activated. Block Parameter transmission as specified in 10.3.5 defers parameter consistency checking and activation until after the complete transmission. The Device performs the checks upon reception of a special command and returns a confirmation or a negative acknowledgment with an appropriate error description. In this case the transmitted parameters shall be rejected and a roll back to the previous parameter set shall be performed to ensure proper functionality of the Device.

10.3.2 Parameter manager state machine

The Device can be parameterized using ISDU mechanisms whenever the PM is active. The main functions of the PM are the transmission of parameters to the Master ("Upload"), to the Device ("Download"), and the consistency and validity checking within the Device ("ValidityCheck") as demonstrated in Figure 86.

The PM is driven by command messages of the Master (see Table B.9). For example, the guard [UploadStart] corresponds to the reception of the SystemCommand "ParamUploadStart" and [UploadEnd] to the reception of the SystemCommand "ParamUploadEnd".

NOTE 1 Following a communication interruption, the Master System Management uses the service SM_DeviceMode with the variable "INACTIVE" to stop the upload process and to return to the "IDLE" state.

Any new "ParamUploadStart" or "ParamDownloadStart" while another sequence is pending, for example due to an unexpected shut-down of a vendor parameterization tool, will abort the pending sequence. The corresponding parameter changes will be discarded.

NOTE 2 A PLC user program and a parameterization tool can conflict (multiple access), for example if during commissioning, the user did not disable accesses from the PLC program while changing parameters via the tool.

The parameter manager mechanism in a Device is always active and the DS_ParUpload.req in transition T4 is used to trigger the Data Storage (DS) mechanism in 10.4.2.

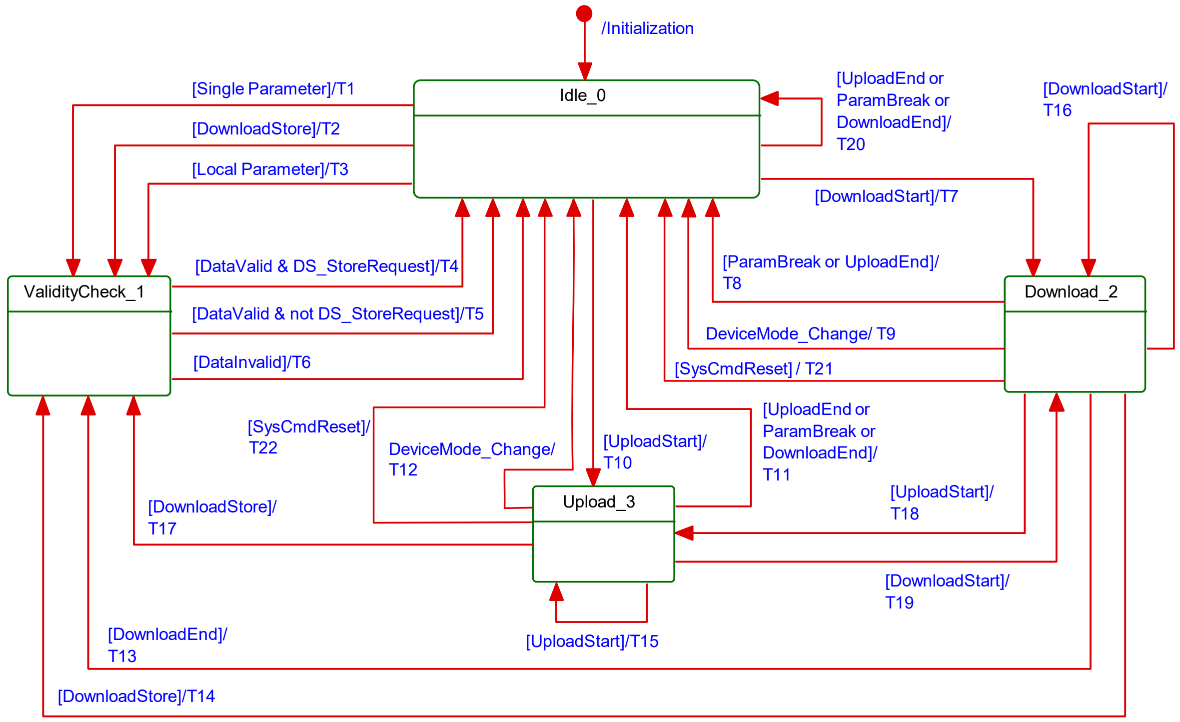


Figure 86 – The Parameter Manager (PM) state machine

Table 96 shows the state transition tables of the Device Parameter Manager (PM) state machine.

Table 96 – State transition tables of the PM state machine

STATE NAME	STATE DESCRIPTION
Idle_0	Waiting on parameter transmission
ValidityCheck_1	Check of consistency and validity of current parameter set.
Download_2	Parameter download active; local parameterization locked (e.g. teach-in). All Read services to Indices other than 3 (DataStorageIndex) shall be rejected (ISDU ErrorType 0x8022 – "Service temporarily not available – Device control") regardless of the result from specific parameter checks (see Table 97)
Upload_3	Parameter upload active; parameterization globally locked. All write accesses for parameter changes not covered in the state machine shall be rejected (ISDU ErrorType 0x8022 – "Service temporarily not available – Device control") regardless of the result from specific parameter checks (see Table 97)

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	-
T2	0	1	Set "StoreRequest" (= TRUE)
T3	0	1	Set "StoreRequest" (= TRUE)
T4	1	0	Mark parameter set as valid; invoke DS_ParUpload.req to DS; enable positive acknowledge of transmission; reset "StoreRequest" (= FALSE)
T5	1	0	Mark parameter set as valid; enable positive acknowledge of transmission
T6	1	0	Mark parameter set as invalid; enable negative acknowledgment of transmission; reset "StoreRequest" (= FALSE); discard parameter buffer
T7	0	2	Lock local parameter access
T8	2	0	Unlock local parameter access; discard parameter buffer
T9	2	0	Unlock local parameter access; discard parameter buffer
T10	0	3	Lock local parameter access

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T11	3	0	Unlock local parameter access
T12	3	0	Unlock local parameter access
T13	2	1	Unlock local parameter access
T14	2	1	Unlock local parameter access; set "StoreRequest" (= TRUE)
T15	3	3	Lock local parameter access
T16	2	2	Discard parameter buffer, so that a possible second start will not be blocked.
T17	3	1	Unlock local parameter access; set "StoreRequest" (= TRUE)
T18	2	3	Discard parameter buffer, so that a possible second start will not be blocked.
T19	3	2	–
T20	0	0	Return ErrorType 0x8036 – *Function temporarily unavailable* if Block Parameterization supported or ErrorType 0x8035 – *Function not available* if Block Parameterization is not supported.
T21	2	0	Unlock local parameter access; discard parameter buffer
T22	3	0	Unlock local parameter access

INTERNAL ITEMS	TYPE	DEFINITION
DownloadStore	Bool	SystemCommand "ParamDownloadStore" received, see Table B.9
DataValid	Bool	Positive result of conformity and validity checking
DataInvalid	Bool	Negative result of conformity and validity checking
DownloadStart	Bool	SystemCommand "ParamDownloadStart" received, see Table B.9
DownloadBreak	Bool	SystemCommand "ParamBreak" or "ParamUploadStart" received
DownloadEnd	Bool	SystemCommand "ParamDownloadEnd" received, see Table B.9
DS_StoreRequest	Bool	Flag for a requested Data Storage sequence, i.e. SystemCommand "ParamDownloadStore" received (= TRUE)
ParamBreak	Bool	SystemCommand "ParamBreak" received, see Table B.9
SysCmdReset	Bool	One of the parameter reset SystemCommands received, see Table 101
DeviceMode_Change	Bool	Reception of SM_DeviceMode with IDLE or STARTUP
UploadStart	Bool	SystemCommand "ParamUploadStart" received, see Table B.9
UploadEnd	Bool	SystemCommand "ParamUploadEnd" received, see Table B.9
Single Parameter	Bool	In case of "single parameter" as specified in 10.3.4
Local Parameter	Bool	In case of "local parameter" as specified in 10.3.3
NOTE "Parameter access locking" shall not be confused with "Device access locking" in Table B.12		

The Parameter Manager (PM) supports handling of "single parameter" (Index and Subindex) transfers as well as "Block Parameter" transmission (entire parameter set).

10.3.3 Dynamic parameter

Parameters accessible through SDCI read or write services may also be changed via on-board control elements (for example teach-in button) or the human machine interface of a Device. These changes shall undergo the same validity checks as a single parameter access. Thus, in case of a positive result "DataValid" in Figure 86, the "StoreRequest" flag shall be applied in order to achieve Data Storage consistency. In case of a negative result "InvalidData", the previous values of the corresponding parameters shall be restored ("roll back"). In addition, a Device specific indication on the human machine interface is recommended as a positive or negative feedback to the user.

It is recommended to avoid concurrent access to a parameter via local control elements and SDCI write services at the same point in time.

10.3.4 Single parameter

Sample sequence charts for valid and invalid single parameter changes are specified in Figure 87.

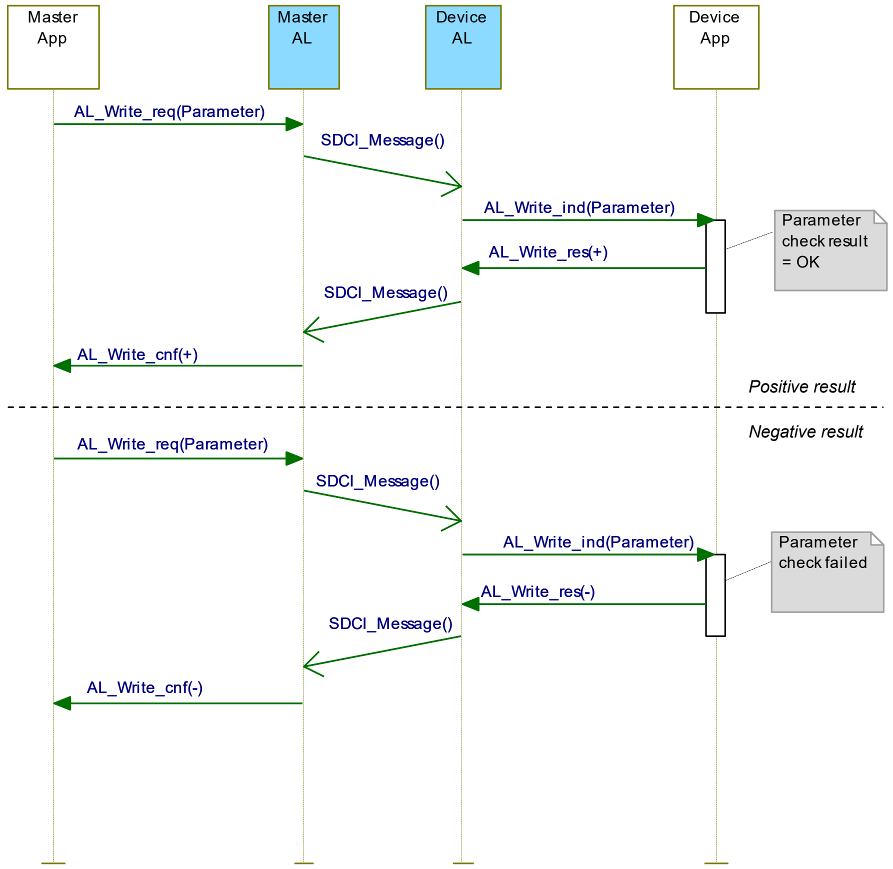


Figure 87 – Positive and negative parameter checking result

If single parameterization is performed via ISDU objects, the Device shall check the access, structure, validity and consistency (see Table 97) of the transmitted data within the context of the entire parameter set and return the result in the confirmation. Via positive conformation, the Device indicates that parameter contents

- passed all checks of Table 97 in the specified order 1 to 4,
- are stored in non-volatile memory in case of non-volatile parameters, and
- are activated in the Device specific technology if applicable.

The negative confirmation carries one of the ErrorTypes of Table C.2 in Annex C.

Table 97 – Sequence of parameter checks

Step	Parameter check	Definition	Error indication
1	Access	Check for valid access rights for this Index / Subindex, independent from data content (Index / Subindex permanent or temporarily unavailable; write/read access on read/write only Index)	See C.2.3 to C.2.8
2	Structure	Check for valid data structure like data size, only complete data structures can be written, for example 2 octets to an UInteger16 data type	See C.2.12 and C.2.13

Step	Parameter check	Definition	Error indication
3	Validity	Check for valid data content of single parameters, testing for data limits	See C.2.9 to C.2.11, C.2.14, C.2.15
4	Consistency	Check for valid data content of the entire parameter set, testing for interference or correlations between parameters	See C.2.16 and C.2.17
NOTE These checks are valid for single and Block Parameters (see 10.3.5)			

10.3.5 Block Parameter

User applications such as function blocks within PLCs and parameterization tool software can use start and end commands to indicate the begin and end of a Block Parameter transmission. For the duration of the Block Parameter transmission the Device application shall inhibit all changes to Read/Write parameters originating from other sources, for example local parameterization, teach-in, etc. In case parameter access is locked, any user application shall unlock “Parameter (write) access” (see Table B.12) prior to downloading a parameter set.

A sample sequence chart for valid Block Parameter changes with an optional Data Storage request is demonstrated in Figure 88.

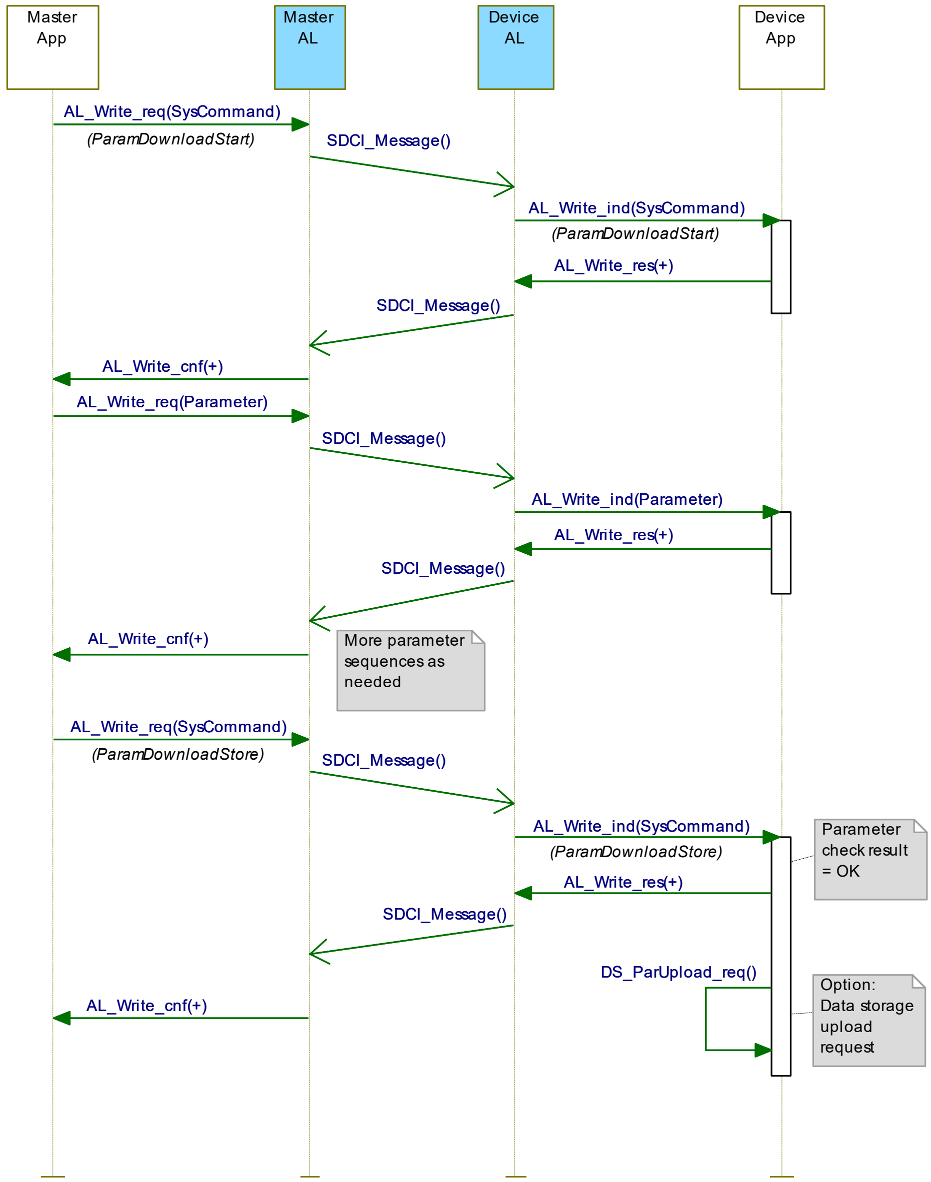
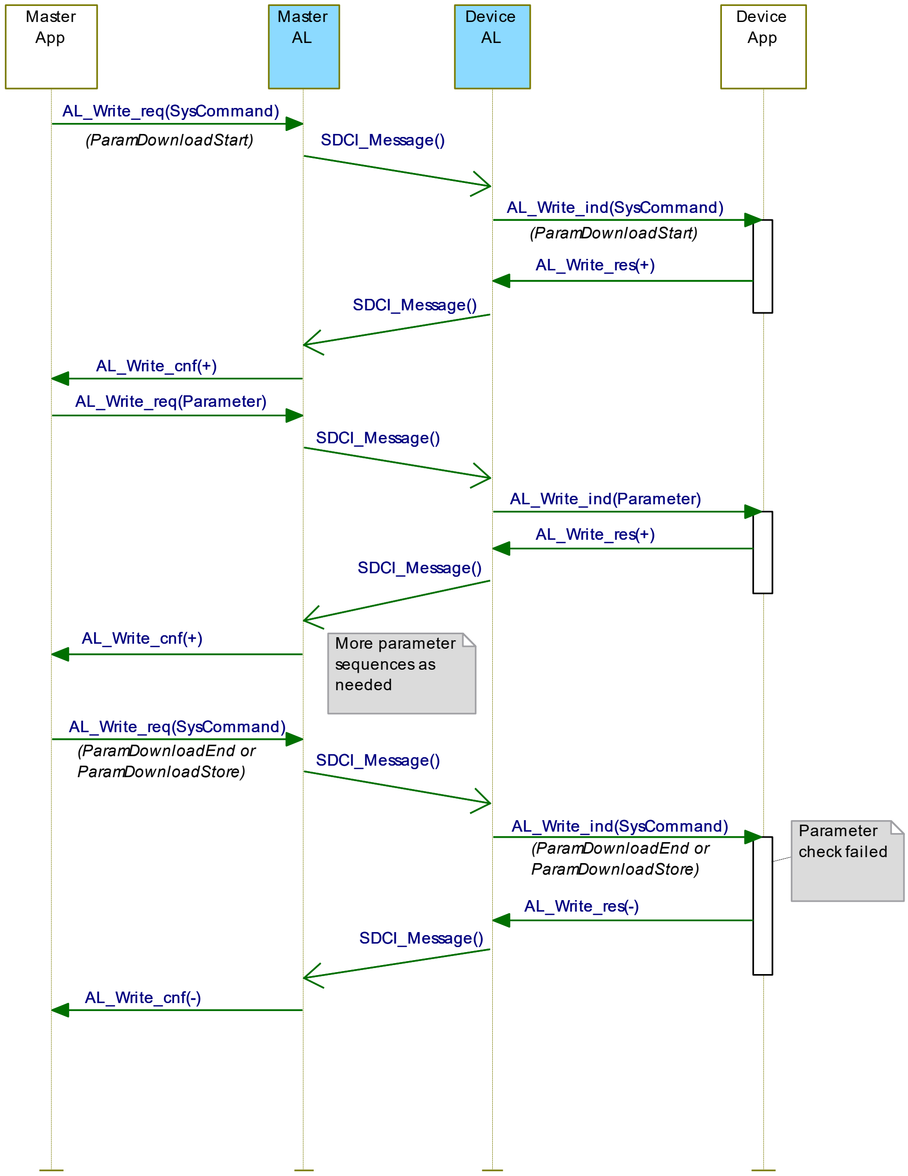


Figure 88 – Positive Block Parameter download with Data Storage request

A sample sequence chart for invalid Block Parameter changes is demonstrated in Figure 89.

The "ParamDownloadStart" command (see Table B.9) indicates the beginning of the Block Parameter transmission in download direction (from user application to the Device). The SystemCommand "ParamDownloadEnd" or "ParamDownloadStore" terminates this sequence. Both functions are similar. However, in addition the SystemCommand "ParamDownloadStore" causes the Data Storage (DS) mechanism to upload the parameter set through the DS_UPLOAD_REQ Event (see 10.4.2).

2864 

2865 **Figure 89 – Negative Block Parameter download**

2866 The checking steps and rules in Table 98 apply.

2867 **Table 98 – Steps and rules for Block Parameter checking**

Rule	Action
1	At first, access and structure checks shall always be performed for each parameter (see Table 97).
2	Then, optionally, validity checks can be performed for each parameter.
3	At this time, consistency checking for transferred parameters shall be disabled and the single parameters shall not be activated.
4	Parameter manager shall not exit from block transfer mode in case of invalid write accesses, structure violations, or validity faults. In case of a ParamDownload the parameter set shall be treated as invalid if one of these checks failed.
5	With command "ParamDownloadEnd" or "ParamDownloadStore", the Device checks validity of each parameter if not already performed and consistency of the entire parameter set. The parameter set shall be treated as invalid if one of these checks failed. The result of the check is indicated to the originator of the Block Parameter transmission within the ISDU acknowledgment in return to the command.

Rule	Action
6	Via positive confirmation the Device indicates that parameters – passed all checks of Table 97, – are stored in non-volatile memory in case of non-volatile parameters, – are activated in the Device specific technology if applicable.
7	Via negative confirmation, the Device indicates that any of the checks of Table 97 failed and the parameter set is invalid. The previous parameter set shall remain active. A Data Storage upload request shall not be triggered. The corresponding negative confirmation shall contain the ErrorType 0x8041 – Inconsistent parameter set (see C.2.17).

The "ParamUploadStart" command (see Table B.9) indicates the beginning of the Block Parameter transmission in upload direction (from the Device to the user application). The SystemCommand "ParamUploadEnd" terminates this sequence, indicates the end of transmission and shall never be rejected with an ErrorCode caused by failed accesses during the block transmission.

A Block Parameter transmission is aborted if the parameter manager receives a SystemCommand "ParamBreak". In this case the block transmission quits without any changes in parameter settings.

In any case, the response to all "ParamXXX" commands (see Table B.9) shall be transmitted after execution of the requested action.

10.3.6 Concurrent parameterization access

There is no mechanism to secure parameter consistency within the Device in case of concurrent accesses from different user applications above Master level. This shall be ensured or blocked on user level (see 13.2.2).

10.3.7 Command handling

Application commands are conveyed in form of parameters. As ISDU response the appropriate priority level of the list in Table 99 shall be used.

Table 99 – Prioritized ISDU responses on command parameters

Priority	ISDU response	Condition
1	"Index not available", see C.2.3	Command parameter is not supported by the Device
2	"Function not available", see C.2.14	Command is not supported by the Device regardless of the Device state
3	"Function temporarily not available", see C.2.15	Command is supported but the actual state of the Device does not permit the requested command.
4	Write response (+)	Command is supported and accepted in the current state of the Device and action is finished. However, within the context of certain commands, the action is just started. This exception is defined at the certain command.

In any case the ISDU timeout shall be observed (see Table 102).

10.4 Data Storage (DS)

10.4.1 General

The Data Storage (DS) mechanism enables the consistent and up-to-date buffering of the Device parameters on upper levels like PLC programs or fieldbus parameter server. Data Storage between Masters and Devices is specified within this standard, whereas the adjacent upper data storage mechanisms depend on the individual fieldbus or system. The Device holds a standardized set of objects providing information about parameters for Data Storage such as memory size requirements as well as control and state information of the Data Storage mechanism (see Table B.10). Revisions of Data Storage parameter sets are identified via a Parameter Checksum.

During Data Storage the Device shall apply the same checking rules as specified for the Block Parameter transfer in 10.3.5.

The implementation of the DS mechanism specified in this standard is highly recommended for Devices. If this mechanism is not supported, it is the responsibility of the Device vendor to describe how parameterization of a Device after replacement can be ensured in a system conform manner without tools.

10.4.2 Data Storage state machine

Any changed set of valid parameters leads to a new Data Storage upload. The upload is initiated by the Device by raising a "DS_UPLOAD_REQ" Event (see Table D.1). The Device shall store the internal state "Data Storage Upload" in non-volatile memory (see Table B.10, State Property), until it receives a Data Storage command "DS_UploadEnd" or "DS_DownloadEnd".

The Device shall generate an Event "DS_UPLOAD_REQ" (see Table D.1) only if the parameter set is valid and

- parameters assigned for Data Storage have been changed locally on the Device (for example teach-in, human machine interface, etc.), or
- parameters assigned for Data Storage have been changed by the SystemCommand Application Reset, or
- the Device receives a SystemCommand "ParamDownloadStore"

With this Event information the Data Storage mechanism of the Master is triggered and initiates a Data Storage upload or download sequence depending on port configuration. The state machine in Figure 90 specifies the Device Data Storage mechanism.

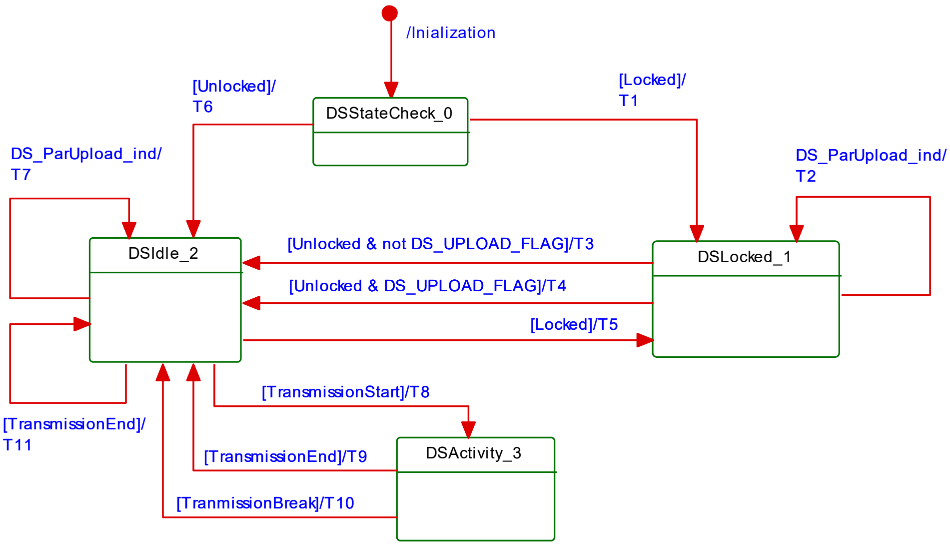


Figure 90 – The Data Storage (DS) state machine

Table 100 shows the state transition tables of the Device Data Storage (DS) state machine. See Table B.10 for details on DataStorageIndex assignments.

Table 100 – State transition table of the Data Storage state machine

STATE NAME	STATE DESCRIPTION
DSStateCheck_0	Check activation state after initialization.
DSLocked_1	Waiting on Data Storage state machine to become unlocked. This state will become obsolete in future releases since Device access lock "Data Storage" shall not be used anymore (see Table B.12). Any DS_Command shall be rejected with the ErrorType "0x8023 Access denied"

STATE NAME	STATE DESCRIPTION
DSIdle_2	Waiting on Data Storage activities. Any unhandled DS-Command shall be rejected with the ErrorType "0x8036 Function temporarily not available"
DSActivity_3	Provide parameter set; local parameterization locked.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	Set State_Property = "Data Storage access locked"
T2	1	1	Set DS_UPLOAD_FLAG = TRUE
T3	1	2	Set State_Property = "Inactive"
T4	1	2	Invoke AL_EVENT.req (EventCode: DS_UPLOAD_REQ), Set State_Property = "Inactive"
T5	2	1	Set State_Property = "Data Storage access locked"
T6	0	2	Set State_Property = "Inactive"
T7	2	2	Set DS_UPLOAD_FLAG = TRUE, invoke AL_EVENT.req (EventCode: DS_UPLOAD_REQ)
T8	2	3	Lock local parameter access, set State_Property = "Upload" or "Download"
T9	3	2	Set DS_UPLOAD_FLAG = FALSE, unlock local parameter access, Set State_Property = "Inactive"
T10	3	2	Unlock local parameter access. Set State_Property = "Inactive"
T11	2	2	Set DS_UPLOAD_FLAG = FALSE

INTERNAL ITEMS	TYPE	DEFINITION
Unlocked	Bool	Data Storage unlocked, see B.2.4
Locked	Bool	Data Storage locked, see B.2.4
DS_ParUpload.ind	Service	Device internal service between PM and DS (see Figure 86)
TransmissionStart	Bool	DS_Command "DS_UploadStart" or "DS_DownloadStart" has been invoked
TransmissionEnd	Bool	DS_Command "DS_UploadEnd" or "DS_DownloadEnd" has been invoked
TransmissionBreak	Bool	DL_Mode.ind(INACTIVE) or DS_Command "DS_Break" received
NOTE "Parameter access locking" shall not be confused with "Device access locking" in Table B.12		

The truncated sequence chart in Figure 91 demonstrates the important communication sequences after the parameterization.

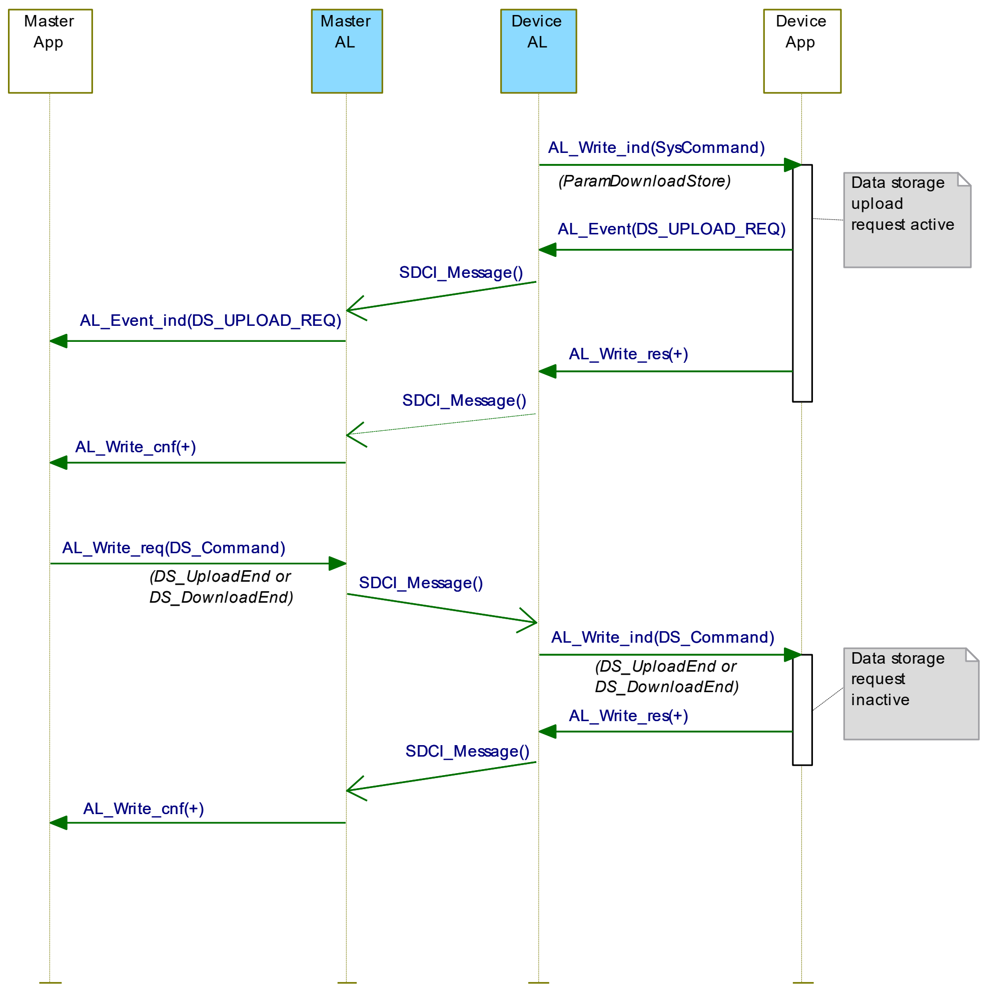


Figure 91 – Data Storage request message sequence

10.4.3 DS configuration

The Data Storage mechanism inside the Device may be disabled via the Master, for example by a tool or a PLC program. See B.2.4 for further details. This is recommended during commissioning or system tests to avoid intensive communication.

NOTE This functionality will be removed in future releases and the Data Storage mechanism will then only be controlled via port configuration in the master.

10.4.4 DS memory space

To handle the requested data amount for Data Storage under any circumstances, the requested amount of indices to be saved and the required total memory space are given in the Data Storage Size parameter, see Table B.10. The required total memory space (including the structural information shall not exceed 2 048 octets (see Annex G). The Data Storage mechanism of the Master shall be able to support this amount of memory per port.

10.4.5 DS Index_List

The Device is the "owner" of the DS Index_List (see Table B.10). Its purpose is to provide all the necessary information for a Device replacement. The DS Index_List shall be fixed for any specific DeviceID. Otherwise the data integrity between Master and Device cannot be guaranteed. The Index List shall contain the termination marker (see Table B.10), if the Device does not support Data Storage (see 10.4.1). The required storage size shall be 0 in this case.

10.4.6 DS parameter availability

All indices listed in the Index List shall be readable and writeable between the SystemCommands "DS_UploadStart" or "DS_DownloadStart" and "DS_UploadEnd" or "DS_DownloadEnd" (see Table B.10). If one of the Indices is rejected by the Device, the Data

Storage Master will abort the up- or download with a SystemCommand "DS_Break". In this case no retries of the Data Storage sequence will be performed.

10.4.7 DS without ISDU

The support of ISDU transmission in a Device is a precondition for the Data Storage of parameters. Parameters in Direct Parameter page 2 cannot be saved and restored by the Data Storage mechanism.

10.4.8 DS parameter change indication

The Parameter_Checksum specified in Table B.10 is used as an indicator for changes in a parameter set. This standard does not require a specific mechanism for detecting parameter changes. A set of recommended methods is provided in the informative Annex K.

10.5 Event Dispatcher (ED)

Any of the Device applications can generate predefined system status information when SDCI operations fail or technology specific information (diagnosis) as a result from technology specific diagnostic methods occur. The Event Dispatcher turns this information into an Event according to the definitions in A.6. The Event consists of an EventQualifier indicating the properties of an incident and an EventCode ID representing a description of this incident together with possible remedial measures. Table D.1 comprises a list of predefined IDs and descriptions for application-oriented incidents. Ranges of IDs are reserved for profile specific and vendor specific incidents. Table D.2 comprises a list of predefined IDs for SDCI specific incidents.

Events are classified in "Errors", "Warnings", and "Notifications". See 10.10.2 for these classifications and see 11.6 for how the Master is controlling and processing these Events.

All Events provided at one point in time are acknowledged with one single command. Therefore, the Event acknowledgment may be delayed by the slowest acknowledgment from upper system levels.

10.6 Device features

10.6.1 General

The following Device features are defined to a certain degree in order to achieve a common behavior. They are accessible via standardized or Device specific methods or parameters. The availability of these features is defined in the IODD of a Device.

10.6.2 Device backward compatibility

This feature enables a Device to play the role of a previous Device revision. In the start-up phase the Master System Management overwrites the Device's inherent DeviceID (DID) with the requested former DeviceID. The Device's technology application shall switch to the former functional sets or subsets assigned to this DeviceID. Device backward compatibility support is optional for a Device.

As a Device can provide backward compatibility to previous DeviceIDs (DID), these compatible Devices shall support all parameters and communication capabilities of the previous DeviceID. Thus, the Device is permitted to change any communication or identification parameter in this case.

10.6.3 Protocol revision compatibility

This feature enables a Device to adjust its protocol layers to a previous SDCI protocol version such as for example to the legacy protocol version of a legacy Master or in the future from version V(x) to version V(x-n). In the start-up phase the Master System Management can overwrite the Device's inherent protocol RevisionID (RID) in case of discrepancy with the RevisionID supported by the Master. A legacy Master does not write the MasterCommand "MasterIdent" (see Table B.2) and thus the Device can adjust to the legacy protocol (V1.0). Revision compatibility support is optional for a Device.

Devices supporting both V1.0 and V1.1 mode are permitted

- to use the same predefined parameters, Events, and ErrorTypes in both modes;
- to support Block Parameterization with full functionality including the Event "DS_UPLOAD_REQ". A legacy Master propagates such an Event without any further action.

10.6.4 Visual SDCI indication

This feature indicates the operational state of the Device's SDCI interface. The indication of the SDCI mode is specified in 10.10.3. Indication of the SIO mode is vendor specific and not covered by this definition. The function is triggered by the indication of the System Management (within all states except SM_Idle and SM_SIO in Figure 81). SDCI indication is optional for a Device.

10.6.5 Parameter access locking

This feature enables a Device to globally lock or unlock write access to all writeable Device parameters accessible via the SDCI interface (see B.2.4). The locking is triggered by the reception of a system parameter "Device Access Locks" (see Table B.8). The support for these functions is optional for a Device.

NOTE It is highly recommended not to implement this feature since it will be omitted in future releases.

10.6.6 Data Storage locking

Setting this lock will cause the "State_Property" in Table B.10 to switch to "Data Storage locked" and the Device not to send a DS_UPLOAD_REQ Event. Support of this function is optional for a Device if the Data Storage mechanism is implemented.

NOTE It is highly recommended not to implement this feature since it will be omitted in future releases.

10.6.7 Locking of local parameter entries

Setting this lock shall have the effect of read only or write protection for local entries at the Device (Bit 2 in Table B.12). Support of this function is optional for a Device, see B.2.4.

10.6.8 Locking of local user interface

Setting this lock shall have the effect of complete disabling of controls and displays, for example shut-down of on-board human machine interface such as keypads on a Device (Bit 3 in Table B.12). Support of this function is optional for a Device.

10.6.9 Offset time

The OffsetTime t_{offset} is a parameter to be configured by the user (see B.2.25). It determines the beginning of the Device's technology data processing in respect to the start of the M-sequence cycle, that means the beginning of the Master (port) message. The offset enables

- Data processing of a Device to be synchronized with the Master (port) cycle within certain limits;
- Data processing of multiple Devices on different Master ports to be synchronized with one another;
- Data processing of multiple Devices on different Master ports to run with a defined offset.

Figure 92 demonstrates the timing of messages in respect to the data processing in Devices.

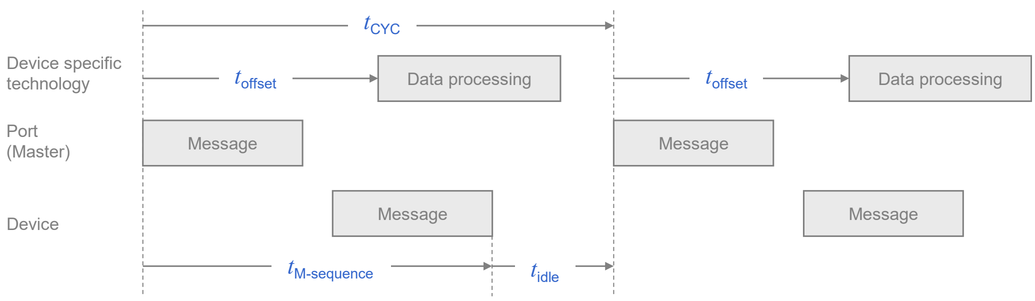


Figure 92 – Cycle timing

The OffsetTime defines a trigger relative to the start of an M-sequence cycle. The support for this function is optional for a Device.

10.6.10 Data Storage concept

The Data Storage mechanism in a Device allows to automatically save parameters in the Data Storage server of the Master and to restore them upon Event notification. Data consistency is checked in either direction within the Master and Device. Data Storage mainly focuses on configuration parameters of a Device set up during commissioning (see 10.4 and 11.4).

10.6.11 Block Parameter

The Block Parameter transmission feature in a Device allows transfer of parameter sets from a PLC program without checking the consistency single data object by single data object. The validity and consistency check are performed at the end of the Block Parameter transmission for the entire parameter set. This function mainly focuses on exchange of parameters of a Device to be set up at runtime (see 10.3). The support of this function is optional for a Device.

10.7 Device reset options

10.7.1 Overview

There are five possibilities for the user to put a Device into a certain defined condition by using either

- Power supply off/on (PowerCycle), or
- SystemCommand "Device reset" (128), or
- SystemCommand "Application reset" (129), or
- SystemCommand "Restore factory settings" (130), or
- SystemCommand "Back to box" (131).

Table B.9 defines which of these SystemCommands are mandatory, highly recommended or optional.

Table 101 provides an overview on impacted items when performing one of these options.

Table 101 – Overview on reset options and their impact on Devices

Impacted item a)	Power-Cycle	Device reset	Application reset	Restore factory settings	Back-to-box
Diagnosis and status	"0"	"0"	No	Clear	"0"
History recorder	No	No	No	No	No
Technology specific parameters (adjustable, teachable)	No	No	Default	Default	Default
Identification/tags	No	No	No	Default	Default
Data Storage behavior	No	No	Upload required DS_UPLOAD_ REQ =1, DS Event	Delete upload request DS_UPLOAD_ REQ =0	Delete upload request DS_UPLOAD_ REQ =0
RevisionID	Default	Default	No	Default	Default
DeviceID	No	No	No	Default	Default
COM behavior	Restart via Master	Restart triggered by Device	No	Restart triggered by Device if necessary, see 10.7.4	Device stops and disables com-munication until next PowerCycle
Access locks	No	No	Default	Default	Default
Block Parameter transfer	–	Discard	Discard	Discard	Discard

Keys

a)	see 10.7.6 for explanation on impacted items
"0"	The numerical parameter or list of parameters contain a zero
PowerCycle	Device power on → off → on
Initial	Set to initial values according to power up state
COM	Communication
No	Not affected
Clear	Set to "0" in case of no COM restart. All active Events will be sent with "Disappear" to clear DeviceStatus. After a performed "Restore factory settings", pending Events can be resent.
Default	Reset to initial value of state of delivery to customer
Event	Trigger upload via DS_UPLOAD_REQ flag
Discard	Transferred parameters not activated

10.7.2 Device reset

This feature enables a Device to perform a "warm start". It is especially useful, whenever a Device needs to be reset to an initial state such as power-on, which means communication will be interrupted.

This feature is triggered upon reception of SystemCommand "Device reset" (see Table B.9). The ISDU response to this SystemCommand shall be transmitted to the Master after successful execution of the requested action. The Device shall wait at least 3 MasterCycle times after the last ISDU Response prior to the communication stop.

The SystemCommand "Device reset" is optional for a Device.

10.7.3 Application reset

This feature enables a Device to reset the technology specific application. It is especially useful, whenever a technology specific application needs to be set to a predefined operational state without communication interruption and a shut-down cycle. Contrary to "Restore factory settings" only the application specific parameters are reset to "Default". Each and every communication and identification parameter remains unchanged.

This feature is triggered upon reception of a SystemCommand "Application reset" (see Table B.9). In any case, the ISDU response to this SystemCommand shall be transmitted to the Master after successful execution of the requested action.

The SystemCommand "Application reset" is highly recommended for a Device.

10.7.4 Restore factory settings

This feature enables a Device to restore parameters to the original delivery status. It is triggered upon reception of the SystemCommand "Restore factory settings" (see Table B.9). The DS_UPLOAD_FLAG (see Table B.10) and other dynamic parameters such as "ErrorCount" (see B.2.18), "DeviceStatus" (see B.2.21), and "DetailedDeviceStatus" (see B.2.22) shall be reset when this feature is applied. This does not include vendor specific parameters such as for example counters of operating hours.

NOTE In this case an existing stored parameter set within the Master will be automatically downloaded into the Device after the next communication restart. This can be avoided by using the “Back to box” SystemCommand (see 10.7.5).

It is the Device vendor's responsibility to guarantee the correct function under any circumstances. If any parameter of the Direct Parameter page 1 (see Direct Parameter page 1 in Table B.1) is changed during this restore, the communication shall be stopped by the Device to trigger a new communication start using the updated communication and identification parameters. The ISDU response to this SystemCommand shall be transmitted to the Master after successful execution of the requested action. The Device shall wait at least 3 MasterCycle times after the last ISDU Response prior to the communication stop.

The SystemCommand "Restore factory settings" is optional for a Device.

10.7.5 Back-to-box

This feature enables a Device to restore parameters to the original delivery values without any interaction with upper level mechanisms such as Data Storage or PLC based parameterization. It is especially useful, whenever a Device is removed from an already parameterized installation and reactivated for example as a spare part. If the Device remains in an automation application beyond the next PowerCycle, all parametrization will be overwritten just as if it were a replacement.

It is triggered upon reception of the SystemCommand "Back-to-box" (see Table B.9), i.e. the Device shall stop and disable communication until next PowerCycle. The ISDU response to this SystemCommand shall be transmitted to the Master after successful execution of the requested action. The Device shall wait at least 3 MasterCycle times after the last ISDU Response prior to the communication stop. All digital signal output drivers shall be disabled and optionally the Device can visually signal the completion of the action.

The SystemCommand "Back-to-box" is conditional on the provision of minimum one user changeable non-volatile parameter.

10.7.6 Explanation on impacted items

The list of impacted items in Table 101 comprises several different parameter types. To explain different categories some standardized parameters are assigned.

- Diagnosis and Status: Comprising the parameters containing the internal Device status like DeviceStatus and DetailledDeviceStatus
- History recorder: Comprising the parameters containing the information regarding the life cycle of the Device like Operating hours counter or minimum or maximum ambient temperature
- Technology specific parameter: Comprising the user settings regarding the Device functionality like AccessLocks or profiled functional parameters like setpoints
- Identification/tags: Comprising the parameters which allow the customer to identify the specific Device by unique identifier like ApplicationSpecificTag, FunctionTag, and LocationTag

10.8 Device design rules and constraints

10.8.1 General

In addition to the protocol definitions in form of state, sequence, activity, and timing diagrams some more rules and constraints are required to define the behavior of the Devices. An overview of the major protocol variables scattered all over the standard is concentrated in Table 102 with associated references.

10.8.2 Process Data

The process communication channel transmits the cyclic Process Data without any interference of the On-request Data communication channels. Process Data exchange starts automatically whenever the Device is switched into the OPERATE state via message from the Master.

The format of the transmitted data is Device specific and varies from no data octets up to 32 octets in each communication direction.

Recommendations:

- Data structures should be suitable for use by PLC applications.
- It is highly recommended to comply with the rules in F.3.3 and in [6].

See 10.2 for details on the indication of valid or invalid Process Data.

10.8.3 Communication loss

It is the responsibility of the Device designer to define the appropriate behaviour of the Device in case communication with the Master is lost (transition T10 in Figure 44 handles detection of the communication loss, while 10.2 defines resulting Device actions).

NOTE This is especially important for actuators such as valves or motor management.

10.8.4 Direct Parameter

The Direct Parameter page communication provides no handshake mechanism to ensure proper reception or validity of the transmitted parameters. The Direct Parameter page can only be accessed single octet by single octet (Subindex) or as a whole (16 octets). The consistency of parameters larger than 1 octet cannot be guaranteed.

The parameters from the Direct Parameter page cannot be saved and restored via the Data Storage mechanism.

10.8.5 ISDU communication channel

The ISDU communication channel provides a powerful means for the transmission of parameters and commands (see Clause B.2).

The following rules shall be considered when using this channel (see Figure 7).

- Index 0 is not accessible via the ISDU communication channel. The access is redirected by the Master to the Direct Parameter page 1 using the page communication channel.
- Index 1 is not accessible via the ISDU communication channel. The access is redirected by the Master to the Direct Parameter page 2 using the page communication channel.
- Index 3 cannot be accessed by a PLC application program. The access is limited to the Master application only (Data Storage).
- After reception of an ISDU request from the Master the Device shall respond within 5 000 ms (see Table 102). Any violation causes the Master to abandon the current task.
- Parameters with attribute write-only (W) shall be treated like a SystemCommand. Only basic data types are permitted.

10.8.6 DeviceID rules related to Device variants

Devices with a certain DeviceID and VendorID shall not deviate in communication and functional behavior. This applies for sensors and actuators. Those Devices may vary for example in

- cable lengths,
- housing materials,
- mounting mechanisms,
- other features, and environmental conditions.

10.8.7 Protocol constants

Table 102 gives an overview of the major protocol constants for Devices.

Table 102 – Overview of the protocol constants for Devices

System variable	References	Values	Definition
ISDU acknowledgment time, for example after a SystemCommand	B.2.2	5 000 ms	Time from reception of an ISDU for example SystemCommand and the beginning of the response message of the Device (see Figure 63)
Maximum number of entries in Index List	B.2.3	70	Each entry comprises an Index and a Subindex. 70 entries results in a total of 210 octets.
Preset values for unused or reserved parameters, for example FunctionID	Annex B	0 (if numbers) 0x00 (if characters)	Engineering shall set all unused parameters to the preset values.
Wake-up procedure	7.3.2.2	See Table 42 and Table 43	Minimum and maximum timings and number of retries
MaxRetry	7.3.3.3	2, see Table 46	Maximum number of retries after communication errors
MinCycleTime	A.3.7 and B.1.3	See Table A.11 and Table B.3	Device defines its minimum cycle time to aquire input or process output data. For constraints of MasterCycleTime see 7.3.3.3
Usable Index range	B.2	See Table B.8	This version of the standard reserves some areas within the total range of 65535 Indices.
Errors and warnings	10.10.2	50 ms	An Event with MODE "Event appears" shall stay at least for the duration of this time.
EventCount	8.2.2.11	1	Constraint for AL_Event.req

10.9 IO Device description (IODD)

An IODD (I/O Device Description) is a file that provides all the necessary properties to establish communication and the necessary parameters and their boundaries to establish the desired function of a sensor or actuator.

An IODD (I/O Device Description) is a file that formally describes a Device.

An IODD file shall be provided for each Device and shall include all information necessary to support this standard.

The IODD can be used by engineering tools for PLCs and/or Masters for the purpose of identification, configuration, definition of data structures for Process Data exchange, parameterization, and diagnosis decoding of a particular Device.

NOTE Details of the IODD language to describe a Device can be found in [6].

10.10 Device diagnosis

10.10.1 Concepts

This standard provides only most common EventCodes in D.2. It is the purpose of these common diagnosis informations to enable an operator or maintenance person to take fast remedial measures without deep knowledge of the Device's technology. Thus, the text associated with a particular EventCode shall always contain a corrective instruction together with the diagnosis information.

Fieldbus-Master-Gateways tend to only map few EventCodes to the upper system level. Usually, vendor specific EventCodes defined via the IODD can only be decoded into readable instructions via a Port and Device Configuration Tool (PDCT) or specific vendor tool using the IODD.

Condensed information of the Device's "state of health" can be retrieved from the parameter "DeviceStatus" (see B.2.21). Whenever an Event appears, the DetailedDeviceStatus contains this Event until it disappears, see B.2.22. Table 103 provides an overview of the various possibilities for Devices and shows examples of consumers for this information.

If implemented, it is also possible to read the number of faults since power-on or reset via the parameter "ErrorCount" (see B.2.18) and more information in case of profile Devices via the parameter "DetailedDeviceStatus" (see B.2.22).

NOTE Profile specific values for the "DetailedDeviceStatus" are given in [7].

A Device may provide additional "deep" technology specific diagnosis information in the form of Device specific parameters (see Table B.8) that can be retrieved via port and Device configuration tools for Masters or via vendor specific tools. Usually, only experts or service personnel of the vendor are able to draw conclusions from this information.

Table 103 – Classification of Device diagnosis incidents

Diagnosis incident	Appear/ disappear	Single shot	Parameter	Destination	Consumer
Error (fast remedy; standard EventCodes)	yes	-	-	PLC or HMI (fieldbus mapping)	Maintenance and repair personnel
Error (IODD: vendor specific EventCodes; see Table D.1)	yes	-	-	PDCT or vendor tool	Vendor service personnel
Error (via Device specific parameters)	-	-	See Table B.8	PDCT or vendor tool	Vendor service personnel
Warning (fast remedy; standard EventCodes)	yes	-	-	PLC or HMI	Maintenance and repair personnel
Warning (IODD: vendor specific EventCodes; see Table D.1)	yes	-		PDCT or vendor tool	Vendor service personnel
Warning (via Device specific parameters)	-	-	See Table B.8		
Notification (Standard EventCodes)	-	yes		PDCT	Commissioning personnel
Detailed Device status	-	-		PDCT or vendor tool	Commissioning personnel and vendor service personnel
Number of faults via parameter "ErrorCount"	-	-	See B.2.20		
Device "health" via parameter "DeviceStatus"	-	-	See B.2.21, Table B.13	HMI, Tools such as "Asset Management"	Operator

10.10.2 Events

MODE values shall be assigned as follows (see A.6.4):

- Events of TYPE "Error" shall use the MODEs "Event appears / disappears"
- Events of TYPE "Warning" shall use the MODEs "Event appears / disappears"
- Events of TYPE "Notification" shall use the MODE "Event single shot"

The following requirements apply:

- All Events already placed in the Event queue are discarded by the Event Dispatcher when communication is interrupted or cancelled. Once communication resumed, the technology specific application is responsible for proper reporting of the current Event causes.
- It is the responsibility of the Event Dispatcher to control the "Event appears" and "Event disappears" flow. Once the Event Dispatcher has sent an Event with MODE "Event appears" for a given EventCode, it shall not send it again for the same EventCode before it has sent an Event with MODE "Event disappears" for this same EventCode.
- Each Event shall use static mode, type, and instance attributes.
- Each vendor specific EventCode shall be uniquely assigned to one of the TYPEs (Error, Warning, or Notification).
- Each appearing Event ("Warning" or "Error") shall change the DeviceStatus from "0: Device is operating properly" to any other valid value.

In order to prevent the diagnosis communication channel (see Figure 7) from being flooded, the following requirements apply:

- The same diagnosis information shall not be reported at less than 1 s intervals. This means that the Event Dispatcher shall not invoke the AL_Event service with the same EventCode and EventQualifier more often than once per second. This measure avoids frequent repetitions of Events.
- The Event Dispatcher shall not issue an "Event disappears" less than 50 ms after the corresponding "Event appears".
- Subsequent incidents of errors or warnings with the same root cause shall be disregarded, that means one root cause shall lead to a single error or warning.
- The Event Dispatcher shall invoke the AL_Event service with an EventCount equal one.
- Errors are prioritized over Warnings.

Figure 93 shows how two successive errors are processed, and the corresponding flow of "Event appears" / "Event disappears" Events for each error.

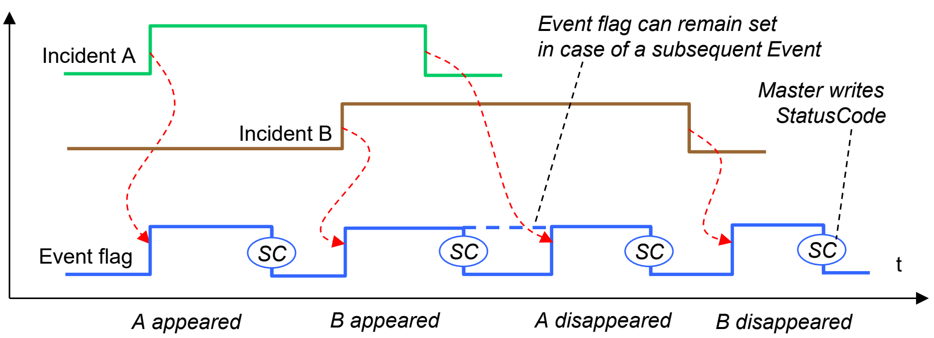


Figure 93 – Event flow in case of successive errors

10.10.3 Visual indicators

The indication of SDCI communication on the Device is optional. The SDCI indication shall use a green indicator. The indication follows the timing and specification shown in Figure 94.

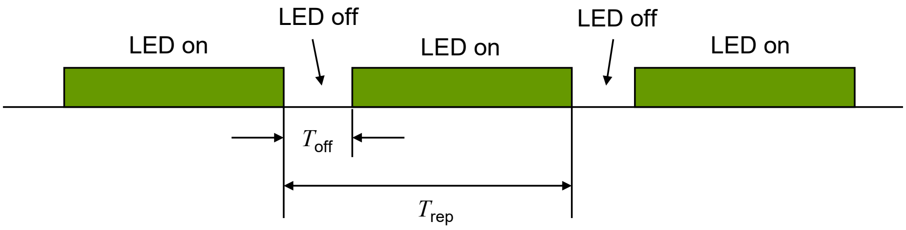


Figure 94 – Device LED indicator timing

Table 104 defines the timing for the LED indicator of Devices.

Table 104 – Timing for LED indicators

Timing	Minimum	Typical	Maximum	Unit
T_{rep}	750	1 000	1 250	ms
T_{off}	75	100	150	ms
T_{off}/T_{rep}	7,5	10	12,5	%

NOTE Timings above are defined such that the general perception would be "power is on".

A short periodical interruption indicates that the Device is in COMx communication state. In order to avoid flickering, the indication cycle shall start with a "LED off" state and shall always be completed (see Table 104).

10.11 Device connectivity

See 5.5 for the different possibilities of connecting Devices to Master ports and the corresponding cable types as well as the color coding.

NOTE For compatibility reasons, this standard does not prevent SDCI devices from providing additional wires for connection to functions outside the scope of this standard (for example to transfer analog output signals).

11 Master

11.1 Overview

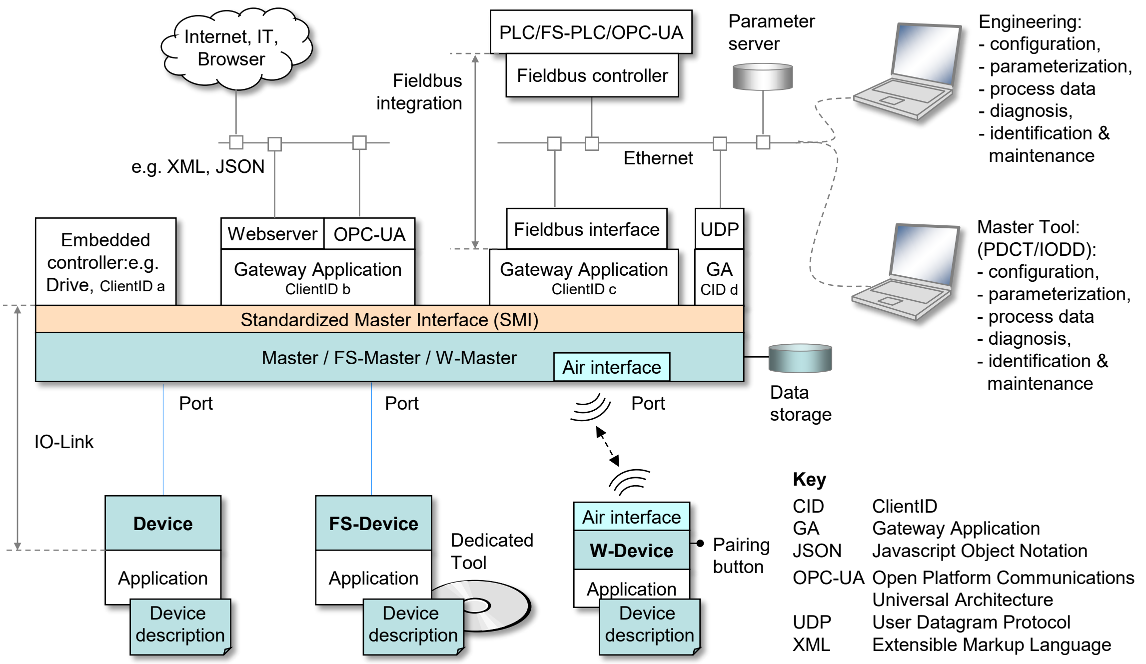
11.1.1 Positioning of Master and Gateway Applications

In 0 the domain of the SDCI technology within the automation hierarchy is already illustrated. Figure 95 shows the recommended relationship between the SDCI technology and a fieldbus technology. Even though this may be the major use case in practice, this does not automatically imply that the SDCI technology depends on the integration into fieldbus systems. It can also be directly integrated into PLC systems, industrial PC, or other automation systems without fieldbus communication in between.

For the sake of preferably uniform behavior of Masters, Figure 95 shows a Standardized Master Interface (SMI) as layer in between the Master and the Gateway Applications or embedded systems on top. This Standardized Master Interface is intended to serve also the safety system extensions as well as the wireless system extensions. In case of FS-Masters, attention shall be payed to the fact, that this SMI in some aspects requires implementation according to safety standards.

The Standardized Master Interface is specified in this clause via services and data objects similar to the other layers (PL, DL, and AL) in this document. It is designed using few uniform base structures that both upper layer fieldbus and upper layer IT systems can use in an efficient manner: push ("write"), pull ("read"), push/pull ("write/read"), and indication ("Event").

The specification of Gateway Applications is not subject of this document. Designers shall observe the realtime requirements of control functions and safety functions in case of concurrent Gateway Applications (see 13.2).



NOTE Blue and orange shaded areas indicate features specified in this standard except those for functional safety (FS) and wireless (W)

Figure 95 – Generic relationship of SDCI and automation technology

11.1.2 Structure, applications, and services of a Master

Figure 96 provides an overview of the complete structure and the services of a Master.

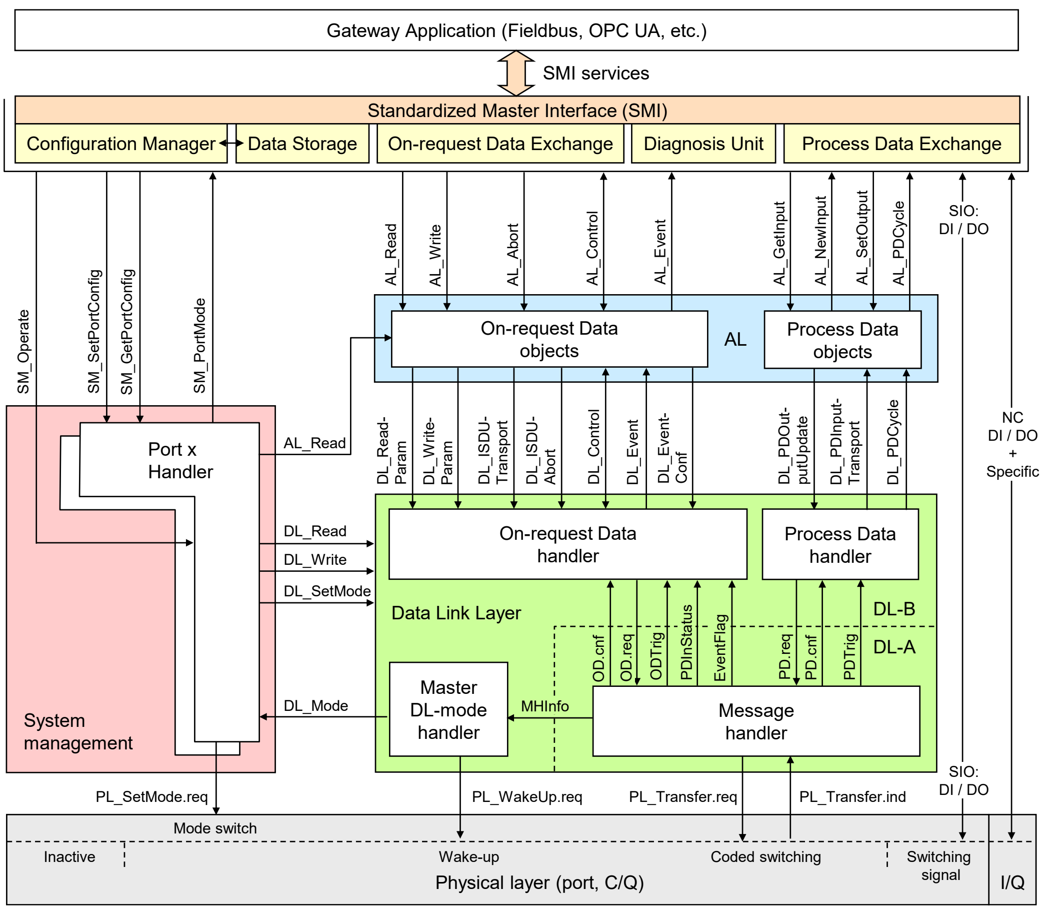


Figure 96 – Structure, applications, and services of a Master

The Master applications are located on top of the Master structure and consist of:

- Configuration Manager (CM), which transforms the user configuration assignments into port set-ups;
- On-request Data Exchange (ODE), which provides for example acyclic parameter access;
- Data Storage (DS) mechanism, which can be used to save and restore the Device parameters;
- Diagnosis Unit (DU), which routes Events from the AL to the Data Storage unit or the gateway application;
- Process Data Exchange (PDE), building the bridge to upper level automation instruments.

They are accessible by the gateway applications (and others) via the Standardized Master Interface (SMI) and its services/methods.

These services and corresponding functions are specified in an abstract manner within clauses 11.2.2 to 11.2.22 and Annex E.

Master applications are described in detail in clauses 11.3 to 11.7. The Configuration Manager (CM) and the Data Storage mechanism (DS) require special coordination with respect to On-request Data.

11.1.3 Object view of a Master and its ports

Figure 97 illustrates the data object model of Master and ports from an SMI point of view.

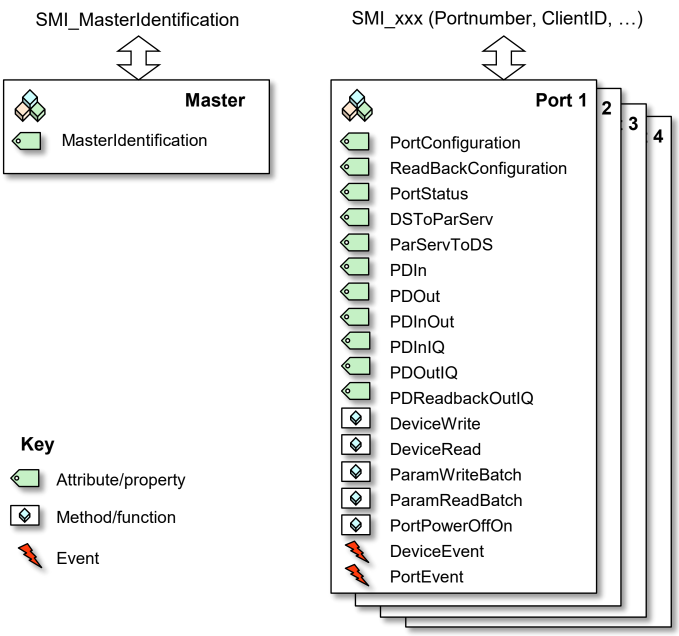


Figure 97 – Object model of Master and Ports

Each object comes with attributes and methods that can be accessed by SMI services. Both, SMI services and attributes/methods/events are specified in the following clause 11.2.

11.2 Services of the Standardized Master Interface (SMI)

11.2.1 Overview

Figure 98 illustrates the individual SMI services available for example to gateway applications.

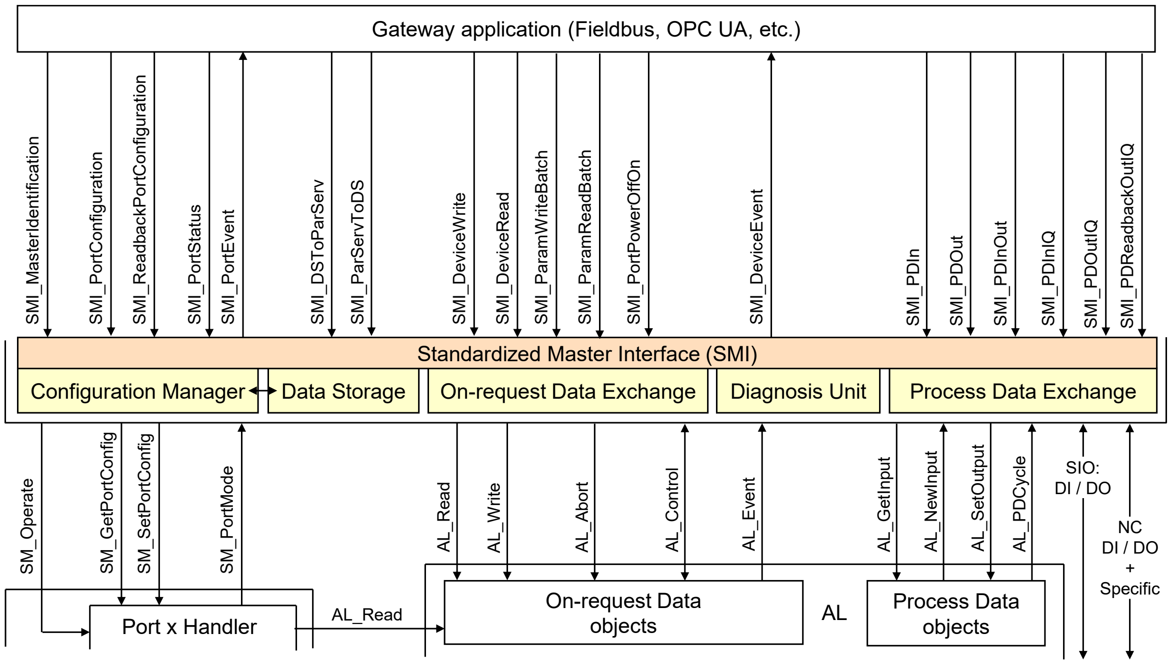


Figure 98 – SMI services

Communication interfaces such as Fieldbus, OPC UA, JSON, UDP or alike are responsible to provide access to the SMI services. It is mandatory for upper level communication systems to refer to the SMI definitions in their adaptations. Functionality behind SMI is mandatory unless it is specifically declared as optional.

Table 105 lists the SMI services available to gateway applications or other clients.

Table 105 – SMI services

Service name	Master	M/O/C	Purpose
SMI_MasterIdentification	R	M	Universal service to identify any Master
SMI_PortConfiguration	R	M	Setting up port configuration
SMI_ReadbackPortConfiguration	R	M	Retrieve current port configuration
SMI_PortStatus	R	M	Retrieve port status
SMI_DSToParServ	R	M	Transfer Data Storage to parameter server
SMI_ParServToDS	R	M	Transfer Parameter server to Data Storage
SMI_DeviceWrite	R	M	ISDU transport to Device
SMI_DeviceRead	R	M	ISDU transport from Device
SMI_ParamWriteBatch	R	O	Batch ISDU transport of parameters (write)
SMI_ParamReadBatch	R	O	Batch ISDU transport of parameters (read)
SMI_PortPowerOffOn	R	O	PortPowerOffOn
SMI_DeviceEvent	I	M	Universal "Push" service for Device Events
SMI_PortEvent	I	M	Universal "Push" service for port Events
SMI_PDIn	R	M	Retrieve PD from InBuffer
SMI_PDOut	R	M	Set PD in OutBuffer
SMI_PDInOut	R	M	Retrieve In- and OutBuffer
SMI_PDInIQ	R	C	Process data in at I/Q (Pin 2 on M12)
SMI_PDOutIQ	R	C	Process data out at I/Q (Pin 2 on M12)
SMI_PDReadbackOutIQ	R	C	Retrieve process data out at I/Q (Pin 2 on M12)

Service name	Master	M/O/C	Purpose
Key I Initiator of service R Receiver (Responder) of service M Mandatory O Optional C Conditional			

11.2.2 Structure of SMI service arguments

The SMI service arguments contain a fixed structure of standard elements, which are characterized in the following.

ClientID
Gateway Applications may use the SMI services concurrently as clients of the SMI (see 11.2.3). Thus, SMI services will assign a unique ClientID to each individual client. It is the responsibility of the Gateway Application(s) to coordinate these SMI service activities and to route responses to the calling client. The maximum number of concurrent clients is Master specific.

Data type: Unsigned8

Permitted values: 1 to vendor specific maximum number of concurrent clients. "0" is solely used for broadcast purposes in case of indications, see 11.2.15 and 11.2.16.

PortNumber
Each SMI service contains the port number in case of an addressed port object (job) or in case of a triggered port object (event).

Data type: Unsigned8

Permitted values: 1 to MaxNumberOfPorts. "0" is solely used to address the entire Master (see 11.2.4).

ExpArgBlockID
This element specifies the expected ArgBlockID to carry the response data of a service request. The IDs are defined in Table E.1.

Data type: Unsigned16

Permitted values: 1 to to 65535

RefArgBlockID
Within results, this element specifies the ID of the Argblock sent by the service request. The IDs are defined in Table E.1.

Data type: Unsigned16

Permitted values: 1 to to 65535

ArgBlockLength
This element specifies the total length of the subsequent ArgBlock. Vendor specific extensions are not permitted.

Data type: Unsigned16

Permitted values: 2 to to 65535

ArgBlock
All SMI services contain an ArgBlock characterized by an ArgBlockID and its description. Service results provide the ArgBlock associated to the ExpArgBlockID, which is part of this ArgBlock. The possibly variable length of the ArgBlock is predefined through definition in this document.

Pairs of ExpArgBlock/RefArgBlock and ArgBlockID within one SMI structure shall be unique. Detailed coding of the ArgBlocks is specified in Annex E. ArgBlock types and their ArgBlockIDs are defined in Table E.1. Service errors are listed at each individual service and in C.4.

11.2.3 Concurrency and prioritization of SMI services

The following rules apply for concurrency of SMI services when accessing attributes:

- All SMI services with different PortNumber access different port objects (disjoint operations);
- Different SMI services using the same PortNumber access different attributes/methods of a port object (concurrent operations);
- Identical SMI services using the same PortNumber and different ClientIDs access identical attributes concurrently (consistency).

The following rules apply for SMI services when accessing methods:

- SMI services for methods using different PortNumbers access different port objects (disjoint operations);
- SMI services for methods using the same PortNumber and different ClientIDs create job instances and will be processed in the order of their arrival (n Client concurrency);
- SMI_ParamWriteBatch (ArgBlock "DeviceBatch") shall be treated as a job instance that shall not be interrupted by any SMI_DeviceWrite or SMI_DeviceRead service.

Prioritization of SMI services within the Standardized Master Interface is not performed. All services accessing methods will be processed in the order of their arrival (first come, first serve).

11.2.4 SMI_MasterIdentification

So far, an explicit identification of a Master did not have priority in SDCI since gateway applications usually provided hard-coded identification and maintenance information as required by the fieldbus system. Due to the requirement "one Master Tool (PCDT) fits different Master brands", corresponding new Master Tools shall be able to connect to Masters providing an SMI. For that purpose, the SMI_MasterIdentification service has been created. It allows Master Tools to adjust to individual Master brands and types, if a particular fieldbus gateway provides the SMI services in a uniform accessible coding (see clause 13). A class of Masters with a certain MasterID and VendorID shall not deviate in communication and functional behavior (Master type identification). Table 106 shows the service SMI_MasterIdentification.

Table 106 – SMI_MasterIdentification

Parameter name	.req	.cnf
Argument	M	
ClientID	M	
PortNumber (0x00)	M	
ExpArgBlockID (e.g. 0x0001)	M	
ArgBlockLength	M	
ArgBlock (VoidBlock: 0xFFF0)	M	
Result (+)		S
ClientID		M
PortNumber (0x00)		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber (0x00)		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber
This parameter contains a virtual Port addressing the entire Master unit (0x00)

ExpArgBlockID

This parameter contains an ArgBlockID of the MasterIdent family, e.g. 0x0001 (see Table E.1)

ArgBlockLength
This parameter contains the length of the "VoidBlock" ArgBlock

ArgBlock
This parameter contains the ArgBlock "VoidBlock" (0xFFF0, see Annex E.17)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Table E.2)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18

Permitted values in prioritized order (see Table C.3):
ARGBLOCK_NOT_SUPPORTED (ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID (incorrect ArgBlock length)

11.2.5 SMI_PortConfiguration

With the help of this service, an SMI client such as a gateway application launches the indicated Master port and the connected Device using the elements in parameter PortConfigList. The service shall be accepted immediately and performed without delay. Content of Data Storage for that port will be deleted at each relevant change of port configuration via "DS_Delete" (see Figure 99). Table 107 shows the structure of the service. The ArgBlock usually is different in SDCI Extensions such as safety and wireless and specified there (see [10] and [11]).

Table 107 – SMI_PortConfiguration

Parameter name	.req	.cnf
Argument	M	
ClientID	M	
PortNumber	M	
ExpArgBlockID (VoidBlock: 0xFFF0)	M	
ArgBlockLength	M	
ArgBlock (e.g. 0x8000)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0x8000)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M

Parameter name		.req	.cnf
PortNumber			M
RefArgBlockID	(ID of request ArgBlock 0x8000)		M
ArgBlockLength			M
ArgBlock	(JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID "VoidBlock" (0xFFF0, see Annex E.17)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock to be "pushed"

ArgBlock
This parameter contains an ArgBlock of the PortConfigList family, e.g. 0x8000 (see Table E.1)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x8000)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (0xFFF0)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x8000)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:

PORT_NUM_INVALID	(incorrect Port number)
ARGBLOCK_NOT_SUPPORTED	(ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID	(incorrect ArgBlock length)
ARGBLOCK_INCONSISTENT	(incorrect ArgBlock content type)
ARGBLOCK_VALOUTOFRANGE	(incorrect ArgBlock content)

11.2.6 SMI_ReadbackPortConfiguration

This service allows for retrieval of the effective configuration of the indicated Master port. Table 108 shows the structure of the service. This service usually is different in SDCI Extensions such as safety and wireless (see [10] and [11]).

Table 108 – SMI_ReadbackPortConfiguration

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (e.g. 0x8000)	M	
ArgBlockLength	M	
ArgBlock (VoidBlock: 0xFFF0)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains an ArgBlockID of the PortConfigList family, e.g. 0x8000 (see Table E.1)

ArgBlockLength
This parameter contains the length of the "VoidBlock" ArgBlock

ArgBlock
This parameter contains the ArgBlock "VoidBlock" (0xFFF0, see Annex E.17)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Annex E.3)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:

PORT_NUM_INVALID	(incorrect Port number)
ARGBLOCK_NOT_SUPPORTED	(ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID	(incorrect ArgBlock length)

11.2.7 SMI_PortStatus

This service allows for retrieval of the effective status of the indicated Master port. Table 109 shows the structure of the service. This service usually is different in SDCI Extensions such as safety and wireless (see [10] and [11]).

Table 109 – SMI_PortStatus

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (e.g. 0x9000)	M	
ArgBlockLength	M	
ArgBlock (VoidBlock: 0xFFF0)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains an ArgBlockID of the PortStatusList family, e.g. 0x9000 (see Table E.1)

ArgBlockLength
This parameter contains the length of the "VoidBlock" ArgBlock

ArgBlock
This parameter contains the ArgBlock "VoidBlock" (0xFFF0, see Annex E.17)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Annex E.4)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:
PORT_NUM_INVALID (incorrect Port number)
ARGBLOCK_NOT_SUPPORTED (ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID (incorrect ArgBlock length)

11.2.8 SMI_DSToParServ

With the help of this service, an SMI client such as a gateway application is able to retrieve the technology parameter set of a Device from Data Storage and back it up within an upper level parameter server (see Figure 95, clauses 11.4, and 13.4.2). Table 110 shows the structure of the service.

In case of DI or DO on this Port, content of Data Storage is cleared. The same applies if Data Storage is not enabled for this Port.

Table 110 – SMI_DSToParServ

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (0x7000)	M	
ArgBlockLength	M	
ArgBlock (VoidBlock: 0xFFF0)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID 0x7000 (see Table E.1)

ArgBlockLength
This parameter contains the length of the "VoidBlock" ArgBlock

ArgBlock
This parameter contains the ArgBlock "VoidBlock" (0xFFF0, see Annex E.17)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Annex E.6)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:

PORT_NUM_INVALID	(incorrect Port number)
ARGBLOCK_NOT_SUPPORTED	(ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID	(incorrect ArgBlock length)

11.2.9 SMI_ParServToDS

With the help of this service, an SMI client such as a gateway application is able to restore the technology parameter set of a Device within Data Storage from an upper level parameter server (see Figure 95, clauses 11.4, and 13.4.2).

Table 111 shows the structure of the service.

In case Data Storage is not supported or not activated on this Port, the service will be replied with Result(-) INCONSISTENT_DS_DATA. The same applies if Data Storage is not consistent with Port configuration, e.g. VendorID does not match.

Table 111 – SMI_ParServToDS

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (VoidBlock: 0xFFF0)	M	
ArgBlockLength	M	
ArgBlock (0x7000)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0x7000)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0x7000)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID "VoidBlock" (0xFFF0, see Annex E.17)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock to be "pushed"

ArgBlock
This parameter contains the ArgBlock DS_Data (0x7000, see Table E.1)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x7000)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x7000)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18

Permitted values in prioritized order:
PORT_NUM_INVALID (incorrect Port number)
ARGBLOCK_NOT_SUPPORTED (ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID (incorrect ArgBlock length)
ARGBLOCK_INCONSISTENT (incorrect ArgBlock content type),
INCONSISTENT_DS_DATA (inconsistent Data Storage data).

11.2.10 SMI_DeviceWrite

This service allows for writing On-request Data (OD) for propagation to the Device. Table 112 shows the structure of the service.

Table 112 – SMI_DeviceWrite

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (VoidBlock: 0xFFF0)	M	
ArgBlockLength	M	
ArgBlock (0x3000)	M	

Parameter name		.req	.cnf
Result (+)			S
ClientID			M
PortNumber			M
RefArgBlockID	(ID of request ArgBlock 0x3000)		M
ArgBlockLength			M
ArgBlock	(associated to the ExpArgBlockID)		M
Result (-)			S
ClientID			M
PortNumber			M
RefArgBlockID	(ID of request ArgBlock 0x3000)		M
ArgBlockLength			M
ArgBlock	(JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID "VoidBlock" (0xFFF0, see Annex E.17)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock to be "pushed

ArgBlock
This parameter contains the ArgBlock "On-requestData" (0x3000, see Table E.1)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x3000)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x3000)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18

Permitted values in prioritized order:

PORT_NUM_INVALID	(incorrect Port number)
ARGBLOCK_NOT_SUPPORTED	(ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID	(incorrect ArgBlock length)
ARGBLOCK_INCONSISTENT	(incorrect ArgBlock content type)
SERVICE_TEMP_UNAVAILABLE	(Master busy)
DEVICE_NOT_ACCESSIBLE	(Device not communicating)
Device ErrorType	(See Annex C.2 and 0)

11.2.11 SMI_DeviceRead

This service allows for reading On-request Data (OD) from the Device via the Master. Table 113 shows the structure of the service.

Table 113 – SMI_DeviceRead

Parameter name		.req	.cnf
Argument			
ClientID		M	
PortNumber		M	
ExpArgBlockID	(0x3000)	M	
ArgBlockLength		M	
ArgBlock	("On-request Data/Index": 0x3001)	M	
Result (+)			S
ClientID			M
PortNumber			M
RefArgBlockID	(ID of request ArgBlock 0x3001)		M
ArgBlockLength			M
ArgBlock	(associated to ExpArgBlockID)		M
Result (-)			S
ClientID			M
PortNumber			M
RefArgBlockID	(ID of request ArgBlock 0x3001)		M
ArgBlockLength			M
ArgBlock	(JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID of "On-requestData" (0x3000, see Table E.1)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock "On-requestData/Index" (0x3001, see Annex E.5)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x3001)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Table E.5)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x3001)

ArgBlockLength

This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18

Permitted values in prioritized order:

PORT_NUM_INVALID	(incorrect Port number)
ARGBLOCK_NOT_SUPPORTED	(ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID	(incorrect ArgBlock length)
ARGBLOCK_INCONSISTENT	(incorrect ArgBlock content type)
SERVICE_TEMP_UNAVAILABLE	(Master busy)
DEVICE_NOT_ACCESSIBLE	(Device not communicating)
Device ErrorType	(See Annex C.2 and 0)

11.2.12 SMI_ParamWriteBatch

This service allows for the "push" transfer of a large number of consistent Device objects via multiple ISDUs. Table 114 shows the structure of the service. The following rules apply:

- The service transfers the ArgBlock "DeviceParBatch" to the Master that conveys the content object by object to the Device via AL_Write (ISDU).
- The same ArgBlock structure is returned as Result (+). However, a value "0x0000" indicates success of a particular AL_Write or an ISDU ErrorType of a failed AL_Write instead of a parameter record.
- Result (-) is only returned in case of a failing service via "JobError".

NOTE1 This service supposes use of Block Parameterization and sufficient buffer ressources

NOTE2 This service may have unexpected duration

This service is optional. Availability is indicated via Master identification (see Table E.2)

Table 114 – SMI_ParamWriteBatch

Parameter name		.req	.cnf
Argument			
ClientID		M	
PortNumber		M	
ExpArgBlockID	DeviceParBatch: 0x7001)	M	
ArgBlockLength		M	
ArgBlock	("DeviceParBatch": 0x7001)	M	
Result (+)			S
ClientID			M
PortNumber			M
RefArgBlockID	(ID of request ArgBlock 0x7001)		M
ArgBlockLength			M
ArgBlock	(associated to the ExpArgBlockID)		M
Result (-)			S
ClientID			M
PortNumber			M
RefArgBlockID	(ID of request ArgBlock 0x7001)		M
ArgBlockLength			M
ArgBlock	(JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID "DeviceParBatch" (0x7001, see Annex E.7)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock to be "pushed"

ArgBlock

This parameter contains the ArgBlock "DeviceParBatch" (0x7001, see Table E.1)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x7001)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Table E.7)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x7001)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:

SERVICE_NOT_SUPPORTED	(Service unknown)
PORT_NUM_INVALID	(incorrect Port number)
ARGBLOCK_NOT_SUPPORTED	(ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID	(incorrect ArgBlock length)
ARGBLOCK_INCONSISTENT	(incorrect ArgBlock content type)
ARGBLOCK_VALOUTOFRANGE	(incorrect ArgBlock content)
MEMORY_OVERRUN	(insufficient memory)
SERVICE_TEMP_UNAVAILABLE	(Master busy)
DEVICE_NOT_ACCESSIBLE	(Device not communicating)

11.2.13 SMI_ParamReadBatch

This service allows for the "pull" transfer of a large number of consistent Device parameters via multiple ISDUs. Table 114 shows the structure of the service. The following rules apply:

- The service transfers the ArgBlock "IndexList" to the Master that transforms the content entry by entry into AL_Read (ISDU) to the Device.
- The corresponding ArgBlock "DeviceParBatch is returned as Result (+). In case of a successful AL_Read of an object, the corresponding parameter record or an ISDU ErrorType of a failed AL_Read instead of a parameter record is returned.
- Result (-) is only returned in case of a failing service via "JobError".

NOTE1 This service supposes use of Block Parameterization and sufficient buffer ressources

NOTE2 This service may have unexpected duration

This service is optional. Availability is indicated via Master identification (see Table E.2)

Table 115 – SMI_ParamReadBatch

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	

Parameter name	.req	.cnf
ExpArgBlockID ("DeviceParBatch": 0x7001)	M	
ArgBlockLength	M	
ArgBlock ("IndexList": 0x7002)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0x7002)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0x7002)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID of "DeviceParBatch" (0x7001, see Table E.1)

ArgBlockLength
This parameter contains the length of the ArgBlock "IndexList"

ArgBlock
This parameter contains the ArgBlock "IndexList" (0x7002, see Table E.1)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x7002)

ArgBlockLength
This parameter contains the conditional length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Table E.7)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x7002)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:
SERVICE_NOT_SUPPORTED (Service unknown)
PORT_NUM_INVALID (incorrect Port number)
ARGBLOCK_NOT_SUPPORTED (ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID (incorrect ArgBlock length)

ARGBLOCK_INCONSISTENT	(incorrect ArgBlock content type)
ARGBLOCK_VALOUTOFRANGE	(incorrect ArgBlock content)
MEMORY_OVERRUN	(insufficient memory)
SERVICE_TEMP_UNAVAILABLE	(Master busy)
DEVICE_NOT_ACCESSIBLE	(Device not communicating)

11.2.14 SMI_PortPowerOffOn

This service allows for switching Power 1 of a particular port off and on (see 5.4.1). It returns upon elapsed time provided within the ArgBlock. Table 116 shows the structure of the service.

Table 116 – SMI_PortPowerOffOn

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (VoidBlock: 0xFFF0)	M	
ArgBlockLength	M	
ArgBlock ("PortPowerOffOn": 0x7003)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0x7003)		M
ArgBlockLength		M
ArgBlock (associated to the ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber		M
ExpArgBlockID (ID of request ArgBlock 0x7003)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID "VoidBlock" (0xFFF0, see Annex E.17)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock to be "pushed"

ArgBlock
This parameter contains the ArgBlock "PortPowerOffOn" (0x7003, see Table E.1)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x7003)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (0xFFF0)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x7003)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:

PORT_NUM_INVALID	(incorrect Port number)
ARGBLOCK_NOT_SUPPORTED	(ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID	(incorrect ArgBlock length)
ARGBLOCK_INCONSISTENT	(incorrect ArgBlock content type)
ARGBLOCK_VALOUTOFRANGE	(incorrect ArgBlock content)
SERVICE_TEMP_UNAVAILABLE	(Master busy)

11.2.15 SMI_DeviceEvent

This service allows for signaling a Master Event created by the Device. Table 117 shows the structure of the service.

Table 117 – SMI_DeviceEvent

Parameter name	.ind	.rsp
Argument		
ClientID (= "0" → Broadcast)	M	
PortNumber	M	
ExpArgBlockID (VoidBlock: 0xFFF0)	M	
ArgBlockLength	M	
ArgBlock ("DeviceEvent": 0xA000)	M	
Acknowledgment		S
ClientID (= "0")		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xA000)		M
ArgBlockLength		M
ArgBlock (VoidBlock: 0xFFF0)		M

Argument
The specific parameters of this indication are transmitted in the argument (see 11.2.2).

ClientID
For this indication, the ClientID shall be "0" ("broadcast" to upper level system)

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID "VoidBlock" (0xFFF0, see Annex E.17)

ArgBlockLength
This parameter contains the length of the reported ArgBlock 0xA000

ArgBlock
This parameter contains the ArgBlock "DeviceEvent" (0xA000, see Table E.1)

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

ClientID
The ClientID shall be "0"

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xA000)

ArgBlockLength

This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (0xFFF0)

11.2.16 SMI_PortEvent

This service allows for signaling a Master Event created by the Port. Table 118 shows the structure of the service.

Table 118 – SMI_PortEvent

Parameter name	.ind	.rsp
Argument		
ClientID (= "0" → Broadcast)	M	
PortNumber	M	
ExpArgBlockID (VoidBlock: 0xFFF0)	M	
ArgBlockLength	M	
ArgBlock (PortEvent: 0xA001)	M	
Acknowledgment		S
ClientID (= "0")		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xA001)		M
ArgBlockLength		M
ArgBlock (VoidBlock: 0xFFF0)		M

Argument
The specific parameters of this indication are transmitted in the argument (see 11.2.2).

ClientID
For this indication, the ClientID shall be "0" ("broadcast" to upper level system)

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID "VoidBlock" (0xFFF0, see Annex E.17)

ArgBlockLength
This parameter contains the length of the reported ArgBlock 0xA001

ArgBlock
This parameter contains the ArgBlock "PortEvent" (0xA001, see Table E.1)

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

ClientID
The ClientID shall be "0"

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xA001)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (0xFFF0)

11.2.17 SMI_PDIn

This service allows for cyclically reading input Process Data from an InBuffer (see 11.7.2.1). Table 119 shows the structure of the service. This service usually has companion services in SDCI Extensions such as safety and wireless (see [10] and [11]).

Table 119 – SMI_PDIn

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (e.g. 0x1001)	M	
ArgBlockLength	M	
ArgBlock (VoidBlock: 0xFFF0)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains an ArgBlockID of the Process Data family, e.g. 0x1001 (see Table E.1)

ArgBlockLength
This parameter contains the length of the "VoidBlock" ArgBlock

ArgBlock
This parameter contains the ArgBlock "VoidBlock" (0xFFF0, see Annex E.17)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock: PDIn
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Annex E.10)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock

This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:
PORT_NUM_INVALID (incorrect Port number)
ARGBLOCK_NOT_SUPPORTED (ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID (incorrect ArgBlock length)
DEVICE_NOT_IN_OPERATE (Process Data not accessible)

11.2.18 SMI_PDOut

This service allows for cyclically writing output Process Data to an OutBuffer (see 11.7.3.1). Table 120 shows the structure of the service. This service usually has companion services in SDCI Extensions such as safety and wireless (see [10] and [11]).

Table 120 – SMI_PDOut

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (VoidBlock: 0xFFF0)	M	
ArgBlockLength	M	
ArgBlock (e.g. 0x1002)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0x1002)		M
ArgBlockLength		M
ArgBlock (VoidBlock: 0xFFF0)		M
Result (-)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0x1002)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID "VoidBlock" (0xFFF0, see Annex E.17)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock to be "pushed"

ArgBlock
This parameter contains ArgBlock of the Process Data family, e.g. 0x1002 (see Table E.1)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x1002)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (0xFFF0)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x1002)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:
PORT_NUM_INVALID (incorrect Port number)
ARGBLOCK_NOT_SUPPORTED (ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID (incorrect ArgBlock length)
ARGBLOCK_INCONSISTENT (incorrect ArgBlock content type)
ARGBLOCK_VALOUTOFRANGE (incorrect ArgBlock content)
DEVICE_NOT_IN_OPERATE (Process Data not accessible)

11.2.19 SMI_PDInOut

This service allows for periodically reading input from an InBuffer (see 11.7.2.1) and periodically reading output Process Data from an OutBuffer (see 11.7.3.1). Table 121 shows the structure of the service. This service usually has companion services in SDCI Extensions such as safety and wireless (see [10] and [11]).

Table 121 – SMI_PDInOut

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (e.g. 0x1003)	M	
ArgBlockLength	M	
ArgBlock (VoidBlock: 0xFFF0)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains an ArgBlockID of the "Process Data" family, e.g. 0x1003 (see Table E.1)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock "VoidBlock" (0xFFF0, see Annex E.17)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Annex E.12)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:
PORT_NUM_INVALID (incorrect Port number)
ARGBLOCK_NOT_SUPPORTED (ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID (incorrect ArgBlock length)
DEVICE_NOT_IN_OPERATE (Process Data not accessible)

11.2.20 SMI_PDInIQ

This service allows for cyclically reading input Process Data from an InBuffer (see 11.7.2.1) containing the value of the input "I" signal (Pin 2 at M12). Table 122 shows the structure of the service.

Table 122 – SMI_PDInIQ

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (e.g. 0x1FFE)	M	
ArgBlockLength	M	
ArgBlock (VoidBlock: 0xFFF0)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains an ArgBlockID of the "Process Data" family, e.g. 0x1FFE (see Table E.1)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock "VoidBlock" (0xFFF0, see Annex E.17)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Annex E.13)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:
SERVICE_NOT_SUPPORTED (Service unknown)
PORT_NUM_INVALID (incorrect Port number)
ARGBLOCK_NOT_SUPPORTED (ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID (incorrect ArgBlock length)

11.2.21 SMI_PDOutIQ

This service allows for cyclically writing output Process Data to an OutBuffer (see 11.7.3.1) containing the value of the output "Q" signal (Pin 2 at M12). Table 123 shows the structure of the service.

Table 123 – SMI_PDOutIQ

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (VoidBlock: 0xFFF0)	M	
ArgBlockLength	M	
ArgBlock (e.g. 0x1FFF)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0x1FFF)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M

Parameter name	.req	.cnf
Result (-)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0x1FFF)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains the ArgBlockID "VoidBlock" (0xFFF0, see Annex E.17)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock to be "pushed"

ArgBlock
This parameter contains an ArgBlock of the "Process Data" family, e.g. 0x1FFF (see Table E.1)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x1FFF)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock associated to the ExpArgBlockID (0xFFF0)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0x1FFF

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock
This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:

SERVICE_NOT_SUPPORTED	(Service unknown)
PORT_NUM_INVALID	(incorrect Port number)
ARGBLOCK_NOT_SUPPORTED	(ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID	(incorrect ArgBlock length)
ARGBLOCK_INCONSISTENT	(incorrect ArgBock content type)
ARGBLOCK_VALOUTOFRANGE	(incorrect ArgBlock content)

11.2.22 SMI_PDReadbackOutIQ

This service allows for cyclically reading back input Process Data from an OutBuffer (see 11.7.3.1) containing the value of the output "Q" signal (Pin 2 at M12). Table 124 shows the structure of the service.

Table 124 – SMI_PDReadbackOutIQ

Parameter name	.req	.cnf
Argument		
ClientID	M	
PortNumber	M	
ExpArgBlockID (e.g. 0x1FFF)	M	
ArgBlockLength	M	
ArgBlock (VoidBlock: 0xFFF0)	M	
Result (+)		S
ClientID		M
PortNumber		M
RefArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (associated to ExpArgBlockID)		M
Result (-)		S
ClientID		M
PortNumber		M
ExpArgBlockID (ID of request ArgBlock 0xFFF0)		M
ArgBlockLength		M
ArgBlock (JobError: 0xFFFF)		M

Argument
The specific parameters of the service request are transmitted in the argument (see 11.2.2).

ClientID

PortNumber

ExpArgBlockID
This parameter contains an ArgBlockID of the "Process Data" family, e.g. 0x1FFF (see Table E.1)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock
This parameter contains the ArgBlock "VoidBlock" (0xFFF0, see Annex E.17)

Result (+):
This selection parameter indicates that the service request has been executed successfully.

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the subsequent ArgBlock

ArgBlock: PDOutIQ
This parameter contains the ArgBlock associated to the ExpArgBlockID (see Annex E.14)

Result (-):
This selection parameter indicates that the service request failed

ClientID

PortNumber

RefArgBlockID
This parameter contains as reference the ID of the ArgBlock sent by the request (0xFFF0)

ArgBlockLength
This parameter contains the length of the "JobError" ArgBlock

ArgBlock

This parameter contains the ArgBlock "JobError" (0xFFFF, see Annex E.18)

Permitted values in prioritized order:

SERVICE_NOT_SUPPORTED	(Service unknown)
PORT_NUM_INVALID	(incorrect Port number)
ARGBLOCK_NOT_SUPPORTED	(ArgBlock unknown)
ARGBLOCK_LENGTH_INVALID	(incorrect ArgBlock length)

11.3 Configuration Manager (CM)

11.3.1 Coordination of Master applications

Figure 99 illustrates the coordination between Master applications. Main responsibility is assigned to the Configuration Manager (CM), who initializes port start-ups and who starts or stops the other Master applications depending on a respective port state.

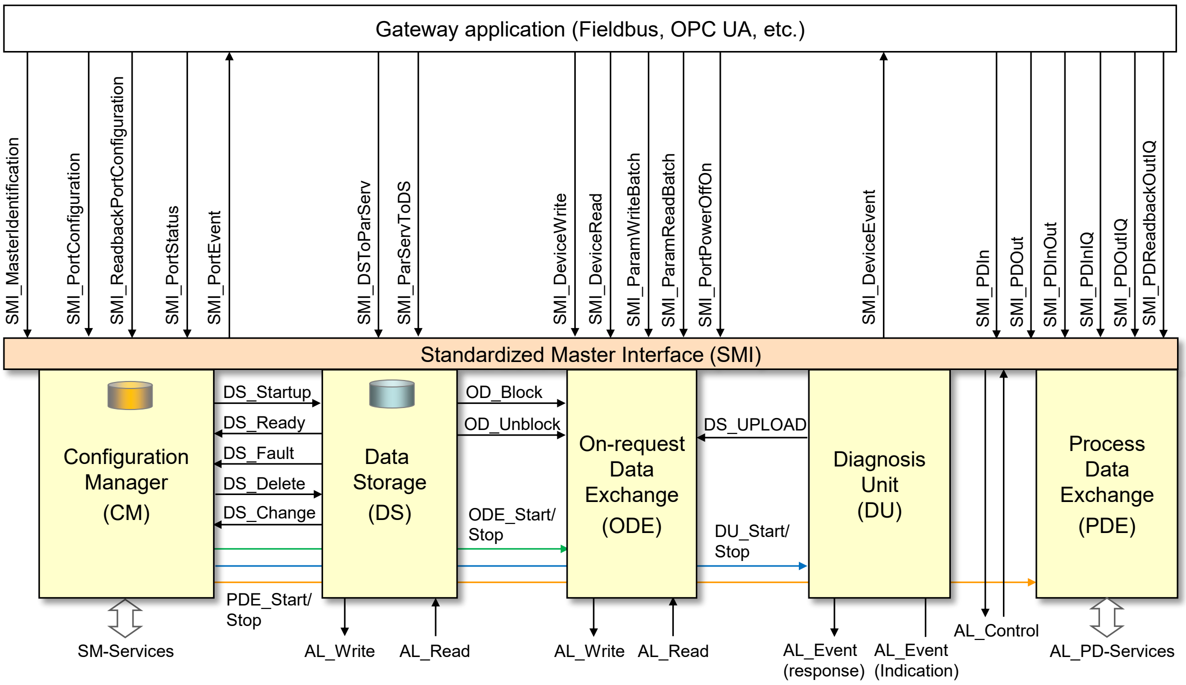


Figure 99 – Coordination of Master applications

Internal variables and Events controlling Master applications are listed in Table 125.

Table 125 – Internal variables and Events controlling Master applications

Internal Variable	Definition
DS_Startup	This variable triggers the Data Storage (DS) state machine causing an Upload or Download of Device parameters if required (see 11.4).
DS_Ready	This variable indicates the Data Storage has been accomplished successfully; operating mode is CFGCOM or AUTOCOM (see 9.2.2.2)
DS_Fault	This variable indicates the Data Storage has been aborted due to a fault.
DS_Delete	Any relevant change of port configuration leads to a deletion of the stored data set in the Data Storage.
DS_Change	This variable indicates a content change of Data Storage triggered by service SMI_ParServToDS.
DS_Upload	This variable triggers the Data Storage state machine in the Master due to the special Event "DS_UPLOAD_REQ" from the Device.
OD_Start	This variable enables On-request Data access via AL_Read and AL_Write.

Internal Variable	Definition
OD_Stop	This variable indicates that On-request Data access via AL_Read and AL_Write is acknowledged with a negative response to the gateway application.
OD_Block	Data Storage upload and download actions disable the On-request Data access through AL_Read or AL_Write. Access by the gateway application is denied.
OD_Unblock	This variable enables On-request Data access via AL_Read or AL_Write.
DU_Start	This variable enables the Diagnosis Unit to propagate remote (Device) Events to the gateway application.
DU_Stop	This variable indicates that the Device Events are not propagated to the gateway application and not acknowledged. Available Events are blocked until the DU is enabled again.
PD_Start	This variable enables the Process Data exchange with the gateway application.
PD_Stop	This variable disables the Process Data exchange with the gateway application.

Restart of a port is basically driven by two activities:

- SMI_PortConfiguration service (Port parameter setting and start-up or changes and restart of a port)
- SMI_ParServToDS service (Download of Data Storage data if Data Storage is activated)

The Configuration Manager (CM) is launched upon reception of a "SMI_PortConfiguration" service. The elements of parameter "PortConfigList" are stored in non-volatile memory within the Master. The service "SMI_ReadbackPortConfiguration" allows for checking correct storage.

CM uses the values of ArgBlock "PortConfigList", initializes the port start-up in case of value changes and conditionally empties the Data Storage via "DS_Delete" or checks emptiness (see Figure 99).

A gateway application can poll the actual port state via "SMI_PortStatus" to check whether the expected port state is reached. In case of fault this service provides corresponding information.

After successfully setting up the port, CM starts the Data Storage mechanism and returns via parameter element "PortStatusInfo" either "OPERATE" or "PORT_FAULT" to the gateway application.

In case of "OPERATE", CM activates the state machines of the associated Master applications Diagnosis Unit (DU), On-request Data Exchange (ODE), and Process Data Exchange (PDE).

In case of a fault in SM_PortMode such as COMP_FAULT, REVISION_FAULT, or SERNUM_FAULT according to 9.2.3, CM activates the state machines of the associated Master applications Diagnosis Unit (DU) and On-request Data Exchange (ODE).

Figure 100 illustrates the start-up of a port via SMI_ PortConfiguration service in a sequence diagram.

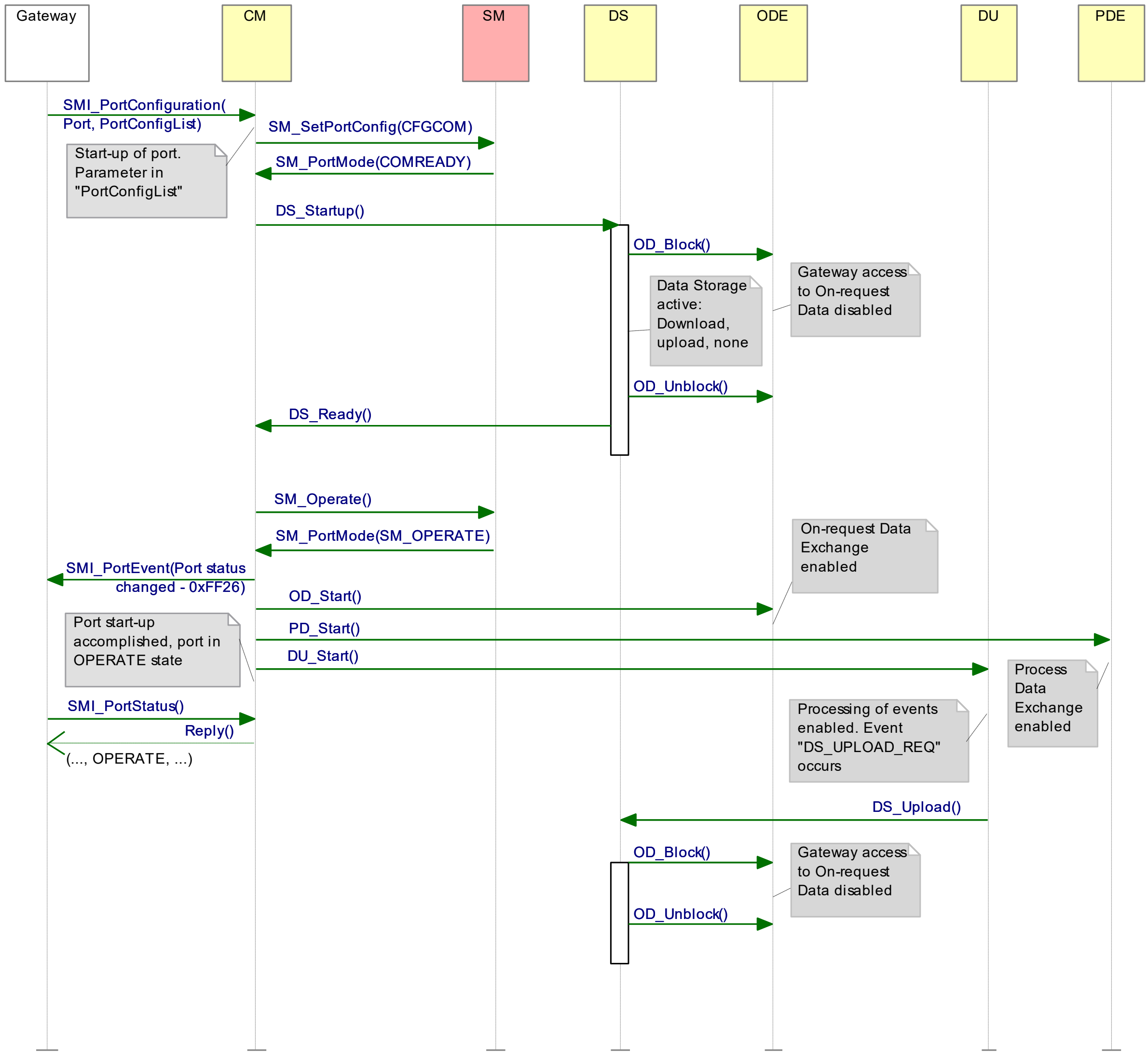


Figure 100 – Sequence diagram of start-up via Configuration Manager

11.3.2 State machine of the Configuration Manager

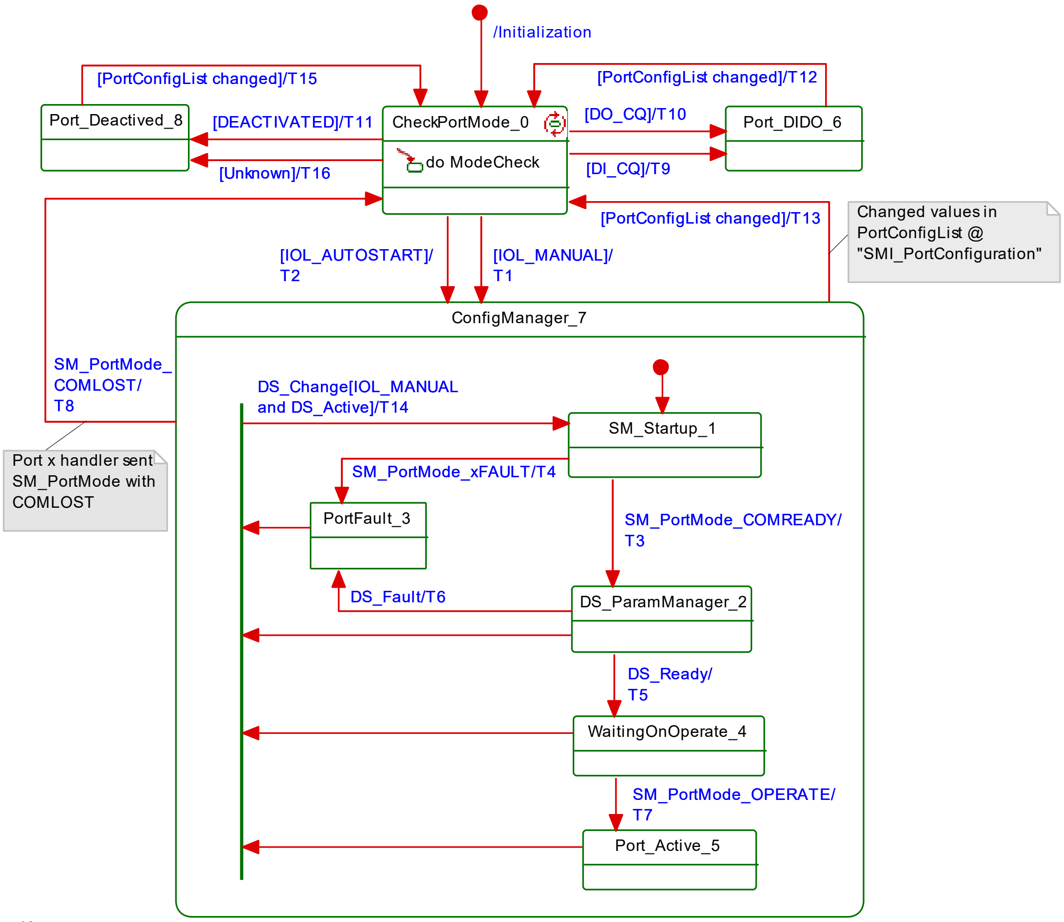
Figure 101 shows the state machine of the Configuration Manager. In general, states and transitions correspond to those of the message handler: STARTUP, PREOPERATE (fault or Data Storage), and at the end OPERATE. Dedicated "SM_PortMode" services are driving the transitions (see 9.2.2.4). A special state is related to SIO mode DI or DO.

Configuration Manager can receive the information COMLOST from Port x Handler through "SM_PortMode" at any time. It also can receive a service "SMI_PortConfiguration" from the gateway application with changed values in "PortConfigList" also at any time (see 11.2.5).

It can also receive a Data Storage object with a changed parameter set via service "SMI_ParServToDS" from the gateway application triggering action in the Configuration Manager if Data Storage is activated.

Port x is started/restarted in all cases.

Figure 101 together with Table 126 also shows transitions leading to corresponding changes in "PortStatusInfo" of ArgBlock "PortStatusList" (see Table E.4). Based on these transitions, Events are triggered via SMI_PortEvent. For details see Clause D.3.



Key
xFAULT: REV_FAULT or COMP_FAULT or SERNUM_FAULT or CYCTIME_FAULT

Figure 101 – State machine of the Configuration Manager

Table 126 shows the state transition tables of the Configuration Manager.

Table 126 – State transition tables of the Configuration Manager

STATE NAME	STATE DESCRIPTION
CheckPortMode_0	Check "Port Mode" element in parameter "PortConfigList" (see 11.2.5)
SM_Startup_1	Waiting on an established communication or loss of communication or any of the faults REVISION_FAULT, COMP_FAULT, or SERNUM_FAULT (see Table 85)
DS_ParamManager_2	Waiting on accomplished Data Storage startup. Parameter are downloaded into the Device or uploaded from the Device.
PortFault_3	Device in state PREOPERATE (communicating). However, one of the three faults REVISION_FAULT, COMP_FAULT, SERNUM_FAULT, or DS_Fault, or PORT_DIAG occurred.
WaitingOnOperate_4	Waiting on SM to switch to OPERATE.
Port_Active_5	Port is in OPERATE mode. The gateway application is exchanging Process Data and ready to send or receive On-request Data.
Port_DIDO_6	Port is in DI or DO mode. The gateway application is exchanging Process Data (DI or DO).
ConfigManager_7	This superstate handles Port communication operations and allows all states inside to react on COMLOST via SM_PortMode service. A Port restart is managed inside the superstate triggered by the DS_Change signal (see Table 125).
Port_Deactivated_8	Port is in DEACTIVATED mode.

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	7	Invoke DS-Delete if identification (VendorID, DeviceID) within DS is different to configured port identification. SM_SetPortConfig_CFGCOM
T2	0	7	Invoke DS-Delete. SM_SetPortConfig_AUTOCOM
T3	1	2	DS_Startup: The DS state machine is triggered. Update parameter elements of "PortStatusList": - PortStatusInfo = NOT_AVAILABLE - RevisionID = (real) RRID - Transmission rate = COMx - VendorID = (real) RVID - DeviceID = (real) RDID - MasterCycleTime = value - Port QualityInfo = invalid
T4	1	3	Update parameter elements of "PortStatusList": - PortStatusInfo = PORT_DIAG - RevisionID = (real) RRID - Transmission rate = COMx - VendorID = (real) RVID - DeviceID = (real) RDID - Port QualityInfo = invalid
T5	2	4	SM_Operate
T6	2	3	Data Storage failed. Rollback to previous parameter set. Update parameter elements of "PortStatusList": - PortStatusInfo = PORT_DIAG - RevisionID = (real) RRID - Transmission rate = COMx - VendorID = (real) RVID - DeviceID = (real) RDID - Port QualityInfo = invalid
T7	4	5	Update parameter elements of "PortStatusList": - PortStatusInfo = OPERATE - RevisionID = (real) RRID - Transmission rate = COMx - VendorID = (real) RVID - DeviceID = (real) RDID - Port QualityInfo = x
T8	1,2,3,4,5	0	Update parameter elements of "PortStatusList": - PortStatusInfo = NO_DEVICE - RevisionID = 0 - Transmission rate = 0 - VendorID = 0 - DeviceID = 0 - Port QualityInfo = Invalid Delete DiagEntries (SOURCE = DEVICE) in PortStatusList (see Table E.4)
T9	0	6	Invoke DS-Delete. SM_SetPortConfig_DI. Update parameter elements of "PortStatusList": - PortStatusInfo = DI_C/Q - RevisionID = 0 - Transmission rate = 0 - VendorID = 0 - DeviceID = 0 - Port QualityInfo = invalid Delete DiagEntries (SOURCE = DEVICE) in PortStatusList (see Table E.4)
T10	0	6	Invoke DS-Delete. SM_SetPortConfig_DO. Update parameter elements of "PortStatusList": - PortStatusInfo = DO_C/Q - RevisionID = 0 - Transmission rate = 0 - VendorID = 0 - DeviceID = 0 - Port QualityInfo = invalid Delete DiagEntries (SOURCE = DEVICE) in PortStatusList (see Table E.4)

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T11	0	8	Invoke DS-Delete. SM_SetPortConfig_INACTIVE. Update parameter elements of "PortStatusList": - PortStatusInfo = DEACTIVATED - RevisionID = 0 - Transmission rate = 0 - VendorID = 0 - DeviceID = 0 - Port QualityInfo = invalid Delete DiagEntries (SOURCE = DEVICE) in PortStatusList (see Table E.4)
T12	6	0	Update parameter elements of "PortStatusList": - PortStatusInfo = NOT_AVAILABLE - RevisionID = 0 - Transmission rate = 0 - VendorID = 0 - DeviceID = 0 - Port QualityInfo = Invalid Delete DiagEntries (SOURCE = DEVICE) in PortStatusList (see Table E.4)
T13	1,2,3,4,5	0	Update parameter elements of "PortStatusList": - PortStatusInfo = NOT_AVAILABLE - RevisionID = 0 - Transmission rate = 0 - VendorID = 0 - DeviceID = 0 - Port QualityInfo = Invalid Delete DiagEntries (SOURCE = DEVICE) in PortStatusList (see Table E.4)
T14	1,2,3,4,5	1	SM_SetPortConfig_CFGCOM Update parameter elements of "PortStatusList": - PortStatusInfo = NOT_AVAILABLE - RevisionID = 0 - Transmission rate = 0 - VendorID = 0 - DeviceID = 0 - Port QualityInfo = Invalid Delete DiagEntries (SOURCE = DEVICE) in PortStatusList (see Table E.4)
T15	8	0	Update parameter elements of "PortStatusList": - PortStatusInfo = NOT_AVAILABLE - RevisionID = 0 - Transmission rate = 0 - VendorID = 0 - DeviceID = 0 - Port QualityInfo = Invalid Delete DiagEntries (SOURCE = DEVICE) in PortStatusList (see Table E.4)
T16	0	8	Invoke DS-Delete. SM_SetPortConfig_INACTIVE. Update parameter elements of "PortStatusList": - PortStatusInfo = DEACTIVATED - RevisionID = 0 - Transmission rate = 0 - VendorID = 0 - DeviceID = 0 - Port QualityInfo = invalid Delete DiagEntries (SOURCE = DEVICE) in PortStatusList (see Table E.4)

INTERNAL ITEMS	TYPE	DEFINITION
PortConfigList changed	Guard	Values of "PortConfigList" have changed
DS_Ready	Signal	Data Storage sequence (upload, download) accomplished; see Table 125.
DS_Fault	Signal	See Table 125
DEACTIVATED	Guard	See Table E.3
IOL_MANUAL	Guard	See Table E.3
IOL_AUTOSTART	Guard	See Table E.3
DI_C/Q	Guard	See Table E.3
DO_C/Q	Guard	See Table E.3

INTERNAL ITEMS	TYPE	DEFINITION
DS_Change	Signal	See Table 125
DS_Active	Guard	Port configured to "Backup + Restore" (3) or "Restore" (4); see Table E.3

State "CheckPortMode_0" contains an activity with complex logic for checking the Port mode within a received Port configuration (see Table E.3). Figure 102 shows this activity within the context of the state machine in Figure 101.

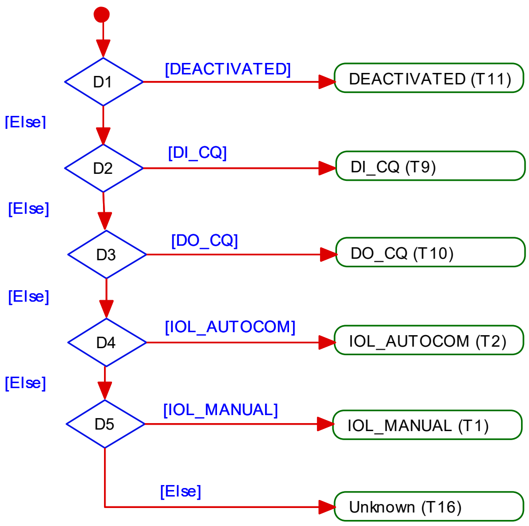


Figure 102 – Activity for state "CheckPortMode_0"

11.4 Data Storage (DS)

11.4.1 Overview

Data Storage between Master and Device is specified within this standard, whereas the adjacent upper Data Storage mechanisms depend on the individual fieldbus or system. The Device holds a standardized set of objects providing parameters for Data Storage, memory size requirements, control and state information of the Data Storage mechanism. Changes of Data Storage parameter sets are detectable via the "Parameter Checksum" (see 10.4.8).

11.4.2 DS data object

The structure of a Data Storage data object is specified in Table G.1.

The Master shall always hold the header information (Parameter Checksum, VendorID, and DeviceID) for the purpose of checking and control. The object information (objects 1… n) will be stored within the non-volatile memory part of the Master (see Annex G). Prior to a download of the Data Storage data object (parameter block), the Master will check the consistency of the header information with the particular Device.

The maximum permitted size of the Data Storage data object is 2048 octets. It is mandatory for Masters to provide at least this memory space per port.

11.4.3 Backup and Restore

Gateways are able to retrieve a port's current Data Storage object out of the Master using the service "SMI_DSToParServ", see 11.2.8.

In return, gateways are also able to write a port's current Data Storage object into the Master using the service "SMI_ParServToDS" (see 11.2.9). This causes under certain conditions an implicit restart of the Device and activation of the parameters within the Device (see 11.3.2).

11.4.4 DS state machine

The Data Storage mechanism is called right after establishing the COMx communication, before entering the OPERATE mode. During this time any other communication with the Device shall be rejected by the gateway.

Figure 103 shows the state machine of the Data Storage mechanism.

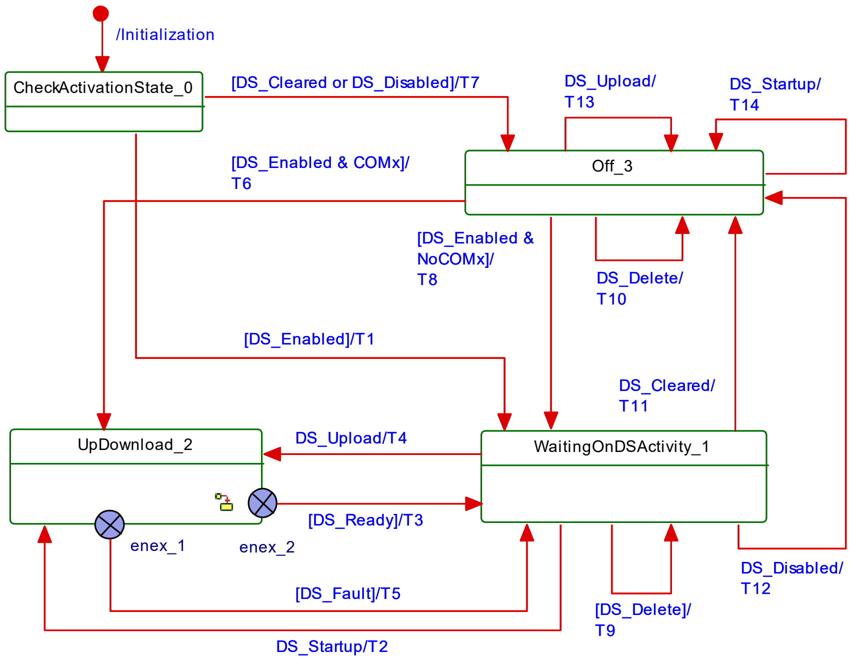


Figure 103 – Main state machine of the Data Storage mechanism

Internal parameter "ActivationState" (DS_Enabled, DS_Disabled, and DS_Cleared) are derived from parameter "Backup behavior" in "SMI_PortConfiguration" service (see 11.2.5 and Table 127 / INTERNAL ITEMS).

Figure 104 shows the submachine of the state "UpDownload_2".

This submachine can be invoked by the Data Storage mechanism or during runtime triggered by a "DS_UPLOAD_REQ" Event.

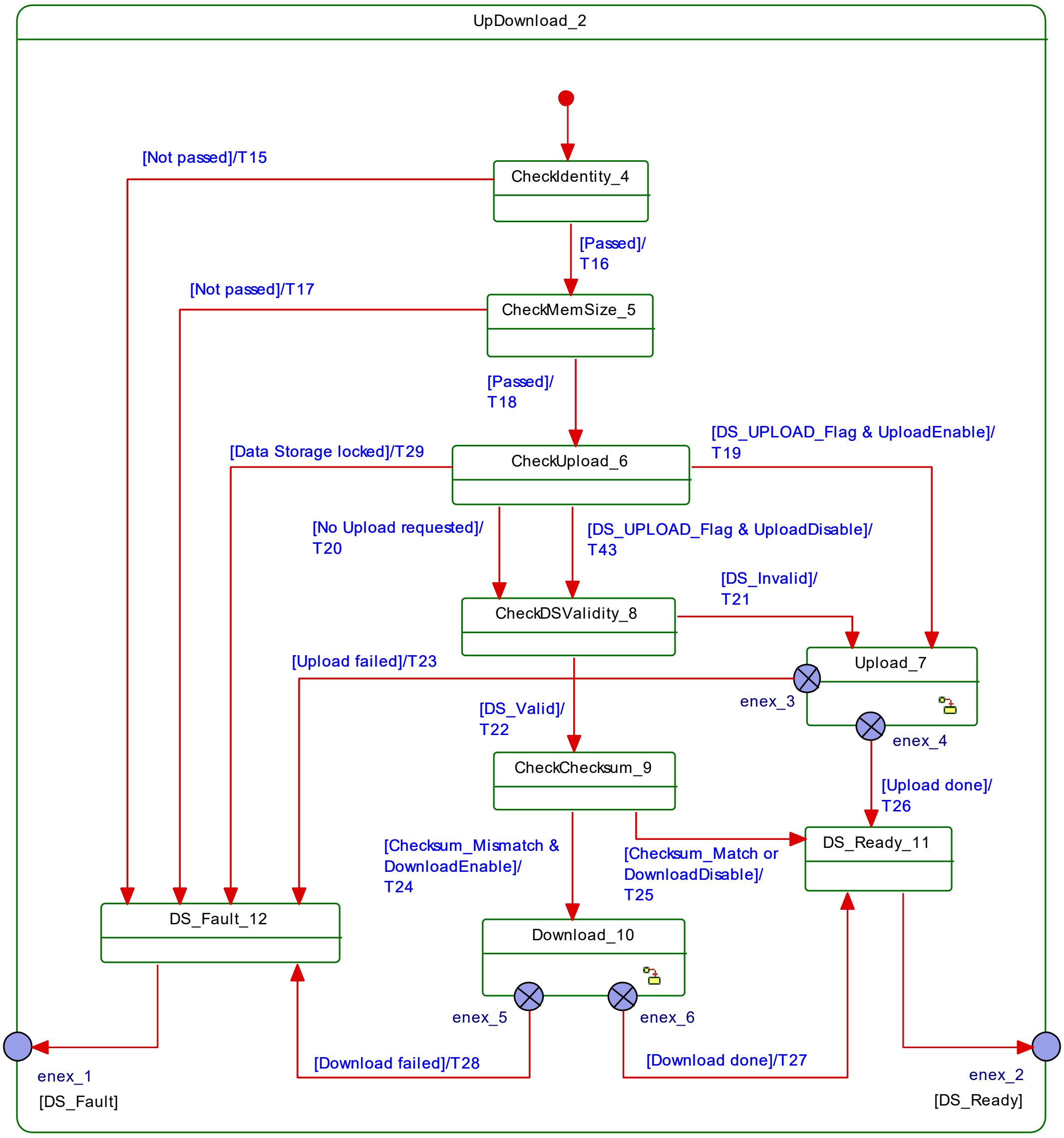


Figure 104 – Submachine "UpDownload_2" of the Data Storage mechanism

Figure 105 shows the submachine of the state "Upload_7".

This state machine can be invoked by the Data Storage mechanism or during runtime triggered by a DS_UPLOAD_REQ Event.

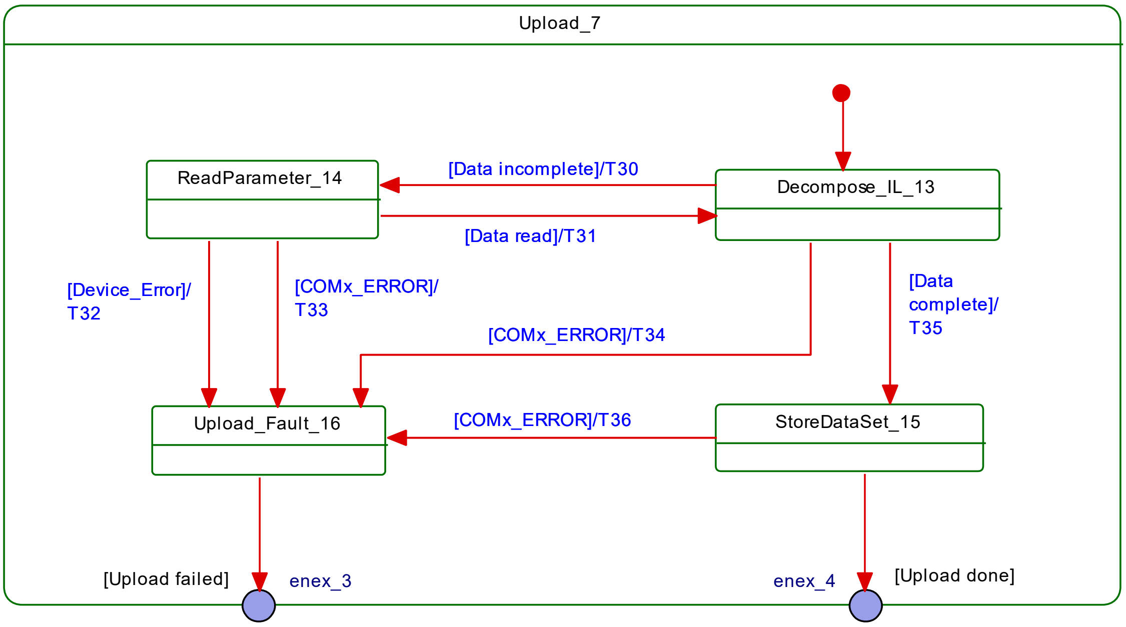


Figure 105 – Data Storage submachine "Upload_7"

Figure 106 demonstrates the Data Storage upload sequence using the DataStorageIndex (DSI) specified in B.2.3 and Table B.10. The structure of Index_List is specified in Table B.11. The DS_UPLOAD_FLAG shall be reset at the end of each sequence (see Table B.10).

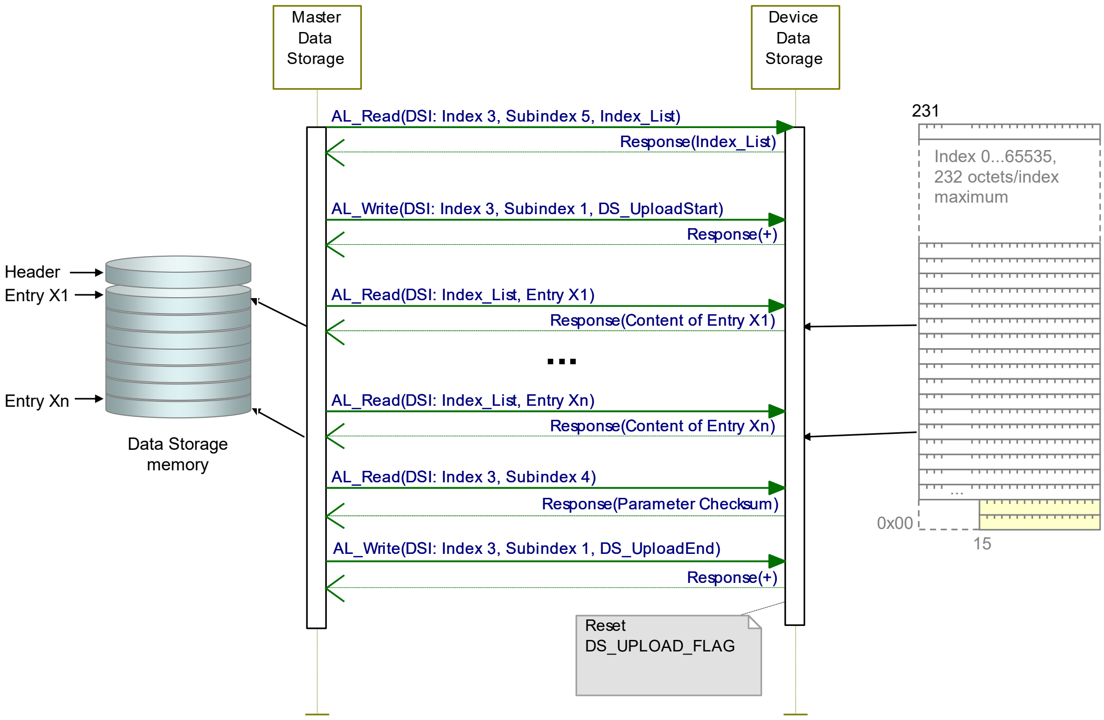


Figure 106 – Data Storage upload sequence diagram

Figure 107 shows the submachine of the state "Download_10".

This state machine can be invoked by the Data Storage mechanism.

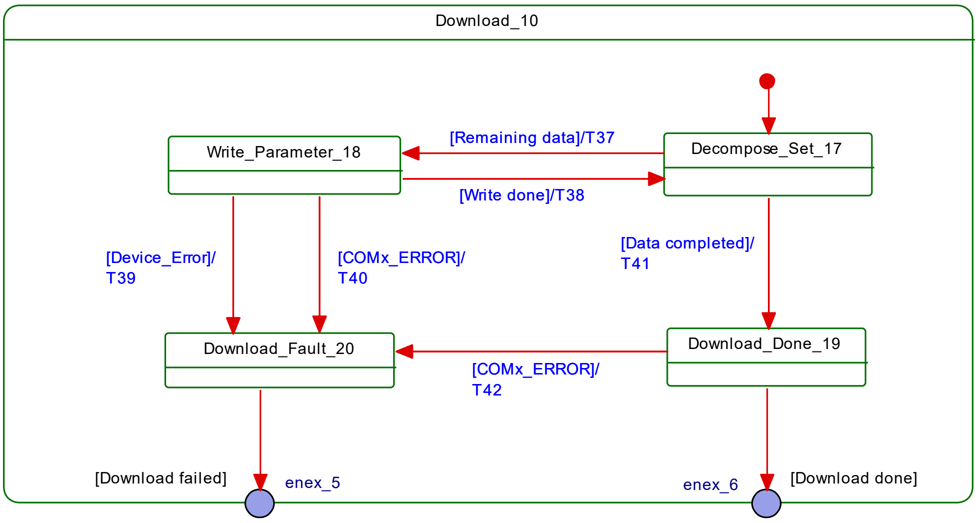


Figure 107 – Data Storage submachine "Download_10"

Figure 108 demonstrates the Data Storage download sequence using the DataStorageIndex (DSI) specified in B.2.3 and Table B.10. The structure of Index_List is specified in Table B.11. The DS_UPLOAD_FLAG shall be reset at the end of each sequence (see Table B.10).

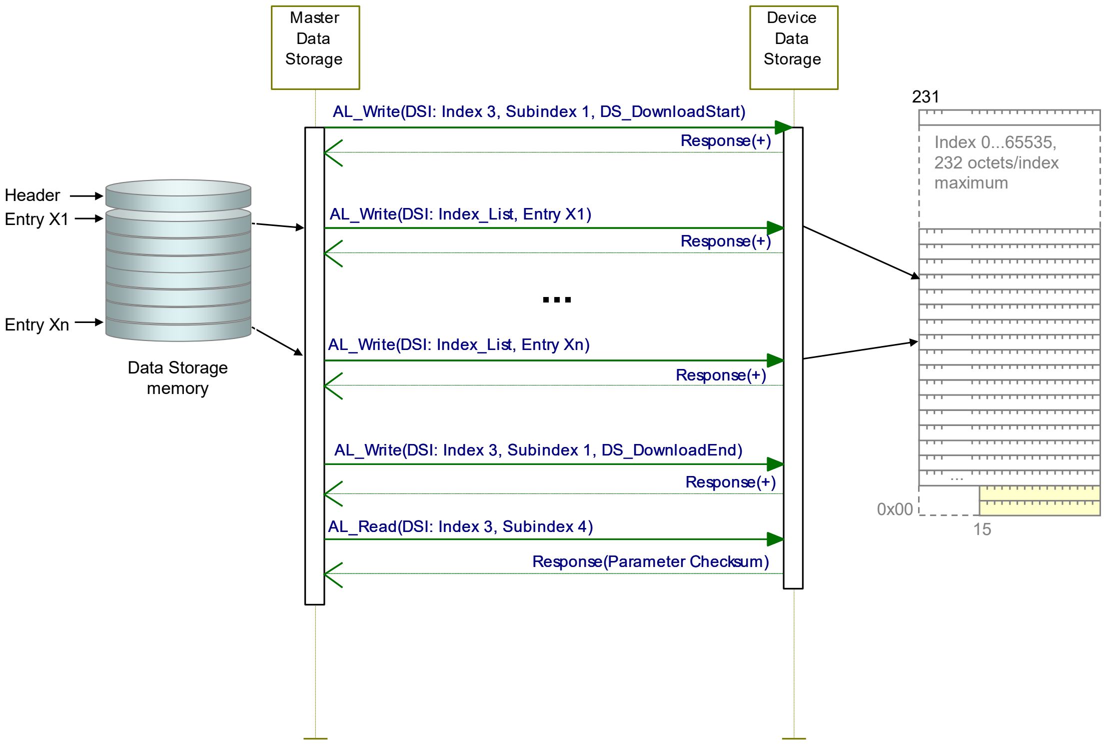


Figure 108 – Data Storage download sequence diagram

Table 127 shows the states and transitions of the Data Storage state machines.

Table 127 – States and transitions of the Data Storage state machines

STATE NAME	STATE DESCRIPTION
CheckActivationState_0	Check current state of the DS configuration: Independently from communication status, DS_Startup from configuration management or an Event DS_UPLOAD_REQ is expected.

STATE NAME	STATE DESCRIPTION
WaitingOnDSActivity_1	Waiting for upload request, Device startup, all changes of activation state independent of the Device communication state.
UpDownload_2	Submachine for up/download actions and checks
Off_3	Data Storage handling switched off or deactivated
SM: CheckIdentity_4	Check Device identification (DeviceID, VendorID) against parameter set within the Data Storage (see Table G.2). Empty content does not lead to a fault.
SM: CheckMemSize_5	Check data set size (Index 3, Subindex 3) against available Master storage size
SM: CheckUpload_6	Check for DS_UPLOAD_FLAG within the DataStorageIndex (see Table B.10)
SM: Upload_7	Submachine for the upload actions
SM: CheckDSValidity_8	Check whether stored data within the Master is valid or invalid. A Master could be replaced between upload and download activities. It is the responsibility of a Master designer to implement a validity mechanism according to the chosen use cases
SM: CheckChecksum_9	Check for differences between the data set content and the Device parameter via the "Parameter Checksum" within the DataStorageIndex (see Table B.10)
SM: Download_10	Submachine for the download actions
SM: DS_Ready_11	Prepare DS_Ready indication to the Configuration Management (CM)
SM: DS_Fault_12	Prepare DS_Fault indication from "Identification_Fault", "SizeCheck_Fault", "Upload_Fault", and "Download_Fault" to the Configuration Management (CM)
SM: Decompose_IL_13	Read Index List within the DataStorageIndex (see Table B.10). Read content entry by entry of the Index List from the Device (see Table B.11).
SM: ReadParameter_14	Wait until read content of one entry of the Index List from the Device is accomplished.
SM: StoreDataSet_15	Task of the gateway application: store entire data set according to Table G.1 and Table G.2
SM: Upload_Fault_16	Prepare Upload_Fault indication from "Device_Error" and "COM_ERROR" as input for the higher-level indication DS_Fault.
SM: Decompose_Set_17	Write parameter by parameter of the data set into the Device according to Table G.1.
SM: Write_Parameter_18	Wait until write of one parameter of the data set into the Device is accomplished.
SM: Download_Done_19	Download completed. Read back "Parameter Checksum" from the DataStorageIndex according to Table B.10. Save this value in the stored data set according to Table G.2.
SM: Download_Fault_20	Prepare Download_Fault indication from "Device_Error" and "COM_ERROR" as input for the higher-level indication DS_Fault.

4342

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	1	–
T2	1	2	–
T3	2	1	OD_Unblock; Indicate DS_Ready to CM
T4	1	2	Confirm Event "DS_Upload" (see INTERNAL ITEMS)
T5	2	1	DS_Break (AL_Write, Index 3, Subindex 1); clear intermediate data (garbage collection); rollback to previous parameter state; DS_Fault (see Figure 98); OD_Unblock.
T6	3	2	–
T7	0	3	–
T8	3	1	–
T9	1	1	Clear saved parameter set (see Table G.1 and Table G.2)
T10	3	3	Clear saved parameter set (see Table G.1 and Table G.2)
T11	1	3	Clear saved parameter set (see Table G.1 and Table G.2)
T12	1	3	–
T13	3	3	Confirm Event "DS_Upload" (see INTERNAL ITEMS); no further action
T14	3	3	DS_Ready to CM

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T15	4	12	Indicate DS_Fault(Identification_Fault) to the gateway application
T16	4	5	Read "Data Storage Size" according to Table B.10, OD_Block
T17	5	12	Indicate DS_Fault(SizeCheck_Fault) to the gateway application
T18	5	6	Read "DS_UPLOAD_FLAG" according to Table B.10
T19	6	7	DataStorageIndex 3, Subindex 1: "DS_UploadStart" (see Table B.10)
T20	6	8	–
T21	8	7	DataStorageIndex 3, Subindex 1: "DS_UploadStart" (see Table B.10)
T22	8	9	–
T23	7	12	DataStorageIndex 3, Subindex 1: "DS_Break" (see Table B.10). Indicate DS_Fault(Upload) to the gateway application
T24	9	10	DataStorageIndex 3, Subindex 1: "DS_DownloadStart" (see Table B.10)
T25	9	11	–
T26	7	11	DataStorageIndex 3, Subindex 1: "DS_UploadEnd"; read Parameter Checksum (see Table B.10)
T27	10	11	–
T28	10	12	DataStorageIndex 3, Subindex 1: "DS_Break" (see Table B.10). Indicate DS_Fault(Download) to the gateway application.
T29	6	12	Indicate DS_Fault(Data Storage locked) to the gateway application
T30	13	14	AL_Read (Index List)
T31	14	13	–
T32	14	16	–
T33	14	16	–
T34	13	16	–
T35	13	15	Read "Parameter Checksum" (see Table B.10).
T36	15	16	–
T37	17	18	Write parameter via AL_Write
T38	18	17	–
T39	18	20	–
T40	18	20	–
T41	17	19	DataStorageIndex 3, Subindex 1: "DS_DownloadEnd" (see Table B.10) Read "Parameter Checksum" (see Table B.10).
T42	19	20	–
T43	6	8	–

4343

INTERNAL ITEMS	TYPE	DEFINITION
DS_Cleared	Bool	Data Storage handling switched off
DS_Disabled	Bool	Data Storage handling deactivated
DS_Enabled	Bool	Data Storage handling activated
COMx_ERROR	Bool	Error in COMx communication detected
Device_Error	Bool	Access to Index denied, AL_Read or AL_Write.cnf(-) with ErrorCode 0x80
DS_Startup	Variable	Trigger from CM state machine, see Figure 99
NoCOMx	Bool	No COMx communication
COMx	Bool	COMx communication working properly
DS_Upload	Variable	Trigger upon DS_UPLOAD_REQ, see Table D.1 and Table B.10
DS_UPLOAD_FLAG	Bool	See Table B.10 ("State property")

INTERNAL ITEMS	TYPE	DEFINITION
UploadEnable	Bool	Data Storage handling configuration
DownloadEnable	Bool	Data Storage handling configuration
DS_Valid	Bool	Valid parameter set available within the Master. See state description "SM: CheckDSValidity_8"
DS_Invalid	Bool	No valid parameter set available within the Master. See state description "SM: CheckDSValidity_8"
Checksum_Mismatch	Bool	Acquired "Parameter Checksum" from Device does not match the checksum within Data Storage (binary comparison)
Checksum_Match	Bool	Acquired "Parameter Checksum" from Device matches the checksum within Data Storage (binary comparison)
Data Storage locked	Bool	See Table B.10 ("State property")

11.4.5 Parameter selection for Data Storage

The Device designer defines the parameters that are part of the Data Storage mechanism.

The IODD marks all parameters not included in Data Storage with the attribute "excludedFromDataStorage". However, the Data Storage mechanism shall not consider the information from the IODD but rather the Parameter List read out from the Device.

11.5 On-request Data exchange (ODE)

Figure 109 shows the state machine of the Master's On-request Data Exchange. This behaveiour is mandatory for a Master.

The gateway application is able to read On-request Data (OD) from the Device via the service "SMI_DeviceRead". This service is directly mapped to service AL_Read with Port, Index, and Subindex (see 8.2.2.1).

The gateway application is able to write On-request Data (OD) to the Device via the service "SMI_DeviceWrite". This service is directly mapped to service AL_Write with Port, Index, and Subindex (see 8.2.2.2).

During an active data transmission of the Data Storage mechanism, all On-request Data requests are blocked.

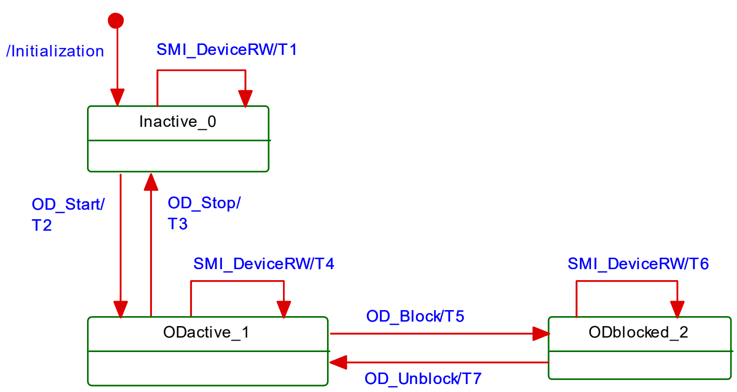


Figure 109 – State machine of the On-request Data Exchange

Table 128 shows the state transition table of the On-request Data Exchange state machine.

Table 128 – State transition table of the ODE state machine

STATE NAME	STATE DESCRIPTION
Inactive_0	Waiting for activation
ODactive_1	On-request Data communication active using AL_Read or AL_Write
ODblocked_2	On-request Data communication blocked

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	0	Access blocked (inactive): indicates "DEVICE_NOT_ACCESSIBLE" to the gateway application
T2	0	1	-
T3	1	0	-
T4	1	1	AL_Read or AL_Write
T5	1	2	-
T6	2	2	Access blocked temporarily: indicates "SERVICE_TEMP_UNAVAILABLE" to the gateway application
T7	2	1	-

INTERNAL ITEMS	TYPE	DEFINITION
SMI_DeviceRW	Variable	On-request Data read or write requested via SMI_DeviceRead, SMI_DeviceWrite, SMI_ParamWriteBatch, or SMI_ParamReadBatch

11.6 Diagnosis Unit (DU)

11.6.1 General

The Diagnosis Unit (DU) routes Device or Port specific Events via the SMI_DeviceEvent and the SMI_PortEvent service to the gateway application (see Figure 99). These Events primarily contain diagnosis information. The structure corresponds to the AL_Event in 8.2.2.11 with Instance, Mode, Type, Origin, and EventCode.

Additionally, the DU generates a Device or port specific diagnosis status that can be retrieved by the SMI_PortStatus service in PortStatusList (see Table E.4 and 11.6.4).

11.6.2 Device specific Events

The SMI_DeviceEvent service provides Device specific Events directly to the gateway application. The special DS_UPLOAD_REQ Event (see 10.4 and Table D.1) of a Device shall be redirectted to the common Master application Data Storage. Those Events are acknowledged by the DU itself and not propagated via SMI_DeviceEvent to the gateway.

Device diagnosis information flooding is avoided by flow control as shown in Figure 110, which allows for only one Event per Device to be propagated via SMI_DeviceEvent to the gateway application at a time.

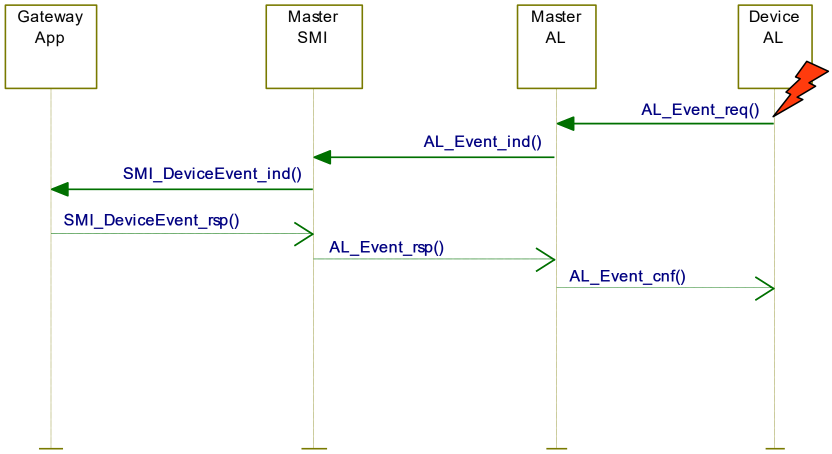


Figure 110 – DeviceEvent flow control

11.6.3 Port specific Events

The SMI_PortEvent service provides also port specific Events directly to the gateway application. Those Events are similarly characterized by Instance = Application, Source = Master, Type = Error or Warning or Notification, and Mode Event appears or disappears or single shot (see A.6.4). Usually, only one port Event at a time is pending as shown in Figure 111.

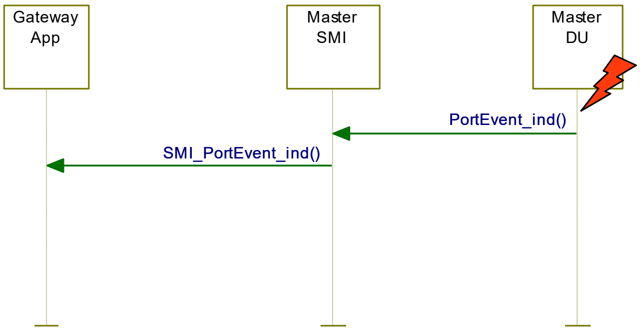


Figure 111 – Port Event flow control

The following rules apply:

- It is not required to send disappearing Port Events in case of Device communication interrupt (communication restart);
- Once communication resumed, the gateway client is responsible for proper reporting of the current Event causes.

Port specific Events are specified in Annex D.3.

11.6.4 Dynamic diagnosis status

DU generates the diagnosis status by collecting all appearing DeviceEvents and PortEvents continuously in a buffer. Any disappearing Event will cause the DU to remove the corresponding Event with the same EventCode from the buffer. Thus, the buffer represents an actual image of the consolidated diagnosis status, which can be taken over as diagnosis entries within the PortStatusList (see Table E.4).

After COMLOST and during Device startup the buffer will be deleted.

11.6.5 Best practice recommendations

Main goal for diagnosis information is to alert an operator in an efficient manner. That means:

- no diagnosis information flooding

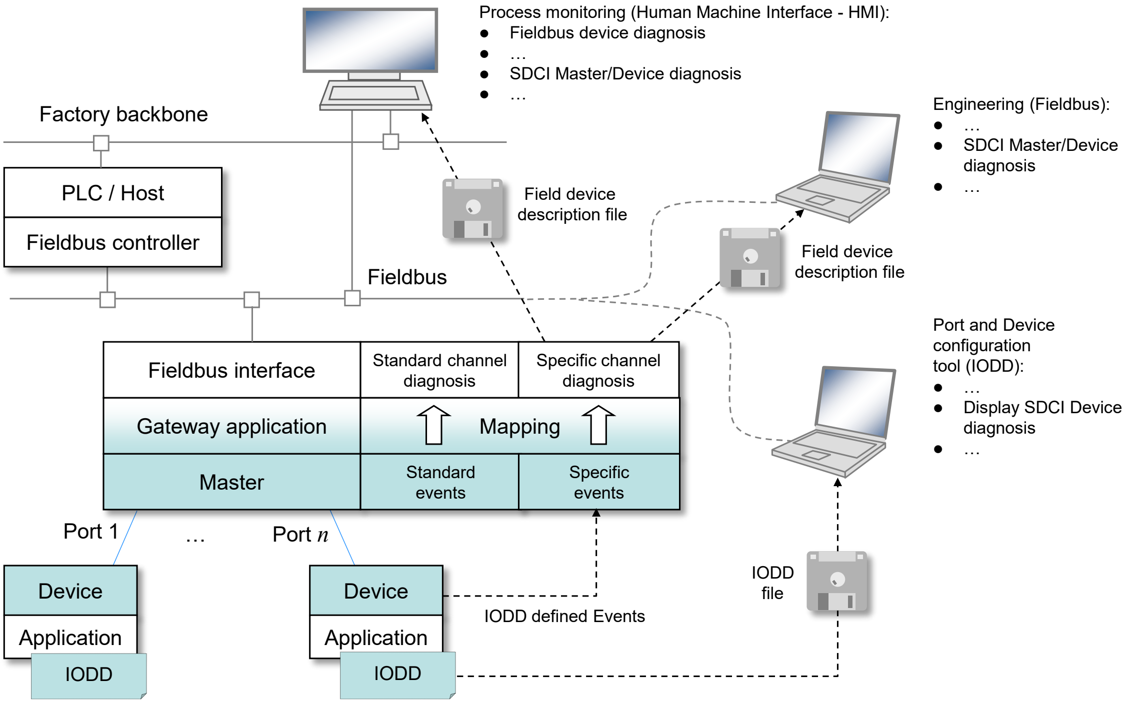
- report of the root cause of an incident within a Device or within the Master and no subsequent correlated faults
- diagnosis information shall provide information on how to maintain or repair the affected component for fast recovery of the automation system.

Figure 112 shows an example of the diagnosis information flow through a complete SDCI/fieldbus system.

NOTE The flow can end at the Master/PDCT or be more integrated depending on the fieldbus capabilities.

Within SDCI, diagnosis information on Devices is conveyed to the Master via Events consisting of EventQualifiers and EventCodes (see A.6). The associated human readable text is available for standardized EventCodes within this standard (see Annex D) and for vendor specific EventCodes within the associated IODD file of a Device.

NOTE The standardized EventCodes can be mapped to semantically identical or closest fieldbus channel diagnosis definitions within the gateway application.



NOTE Blue shaded areas indicate features specified in this standard

Figure 112 – SDCI diagnosis information propagation via Events

11.7 PD Exchange (PDE)

11.7.1 General

The Process Data Exchange provides the transmission of Process Data between the gateway application and the connected Device.

The Standard Master Interface (SMI) comes with the following three services for the gateway application:

- SMI_PDIn allows for reading input Process Data from the InBuffer together with Quality Information (PQI), see 11.2.17
- SMI_PDOut allows for writing output Process Data to the OutBuffer, see 11.2.18
- SMI_PDInOut allows for reading output Process Data from the OutBuffer and reading input Process Data from the InBuffer within one cycle, see 11.2.19

After an established communication and Data Storage, the port is ready for any On-request Data (ODE) transfers. Process Data exchange is enabled whenever the specific port or all ports are switched to the OPERATE mode.

11.7.2 Process Data input mapping

11.7.2.1 Port Modes "IOL_MANUAL" or "IOL_AUTOSTART"

Figure 99 shows how the Master application "Process Data Exchange" (PDE) is related to the other Master applications. It is responsible for the cyclic acquisition of input data using the service "AL_GetInput" (see 8.2.2.4) and of Port Qualifier (PQ) information using the service "AL_Control" (see 8.2.2.12). Both shall be synchronized for consistency.

A gateway application can get access to these data via the service "SMI_PDIn" (see 11.2.17). Figure 113 illustrates the principles of Process Data Input mapping and the content of the ArgBlock of this service (see E.10) consisting of the ArgBlockID, the qualifier PQI, the parameter InputDataLength, and the input Process Data.

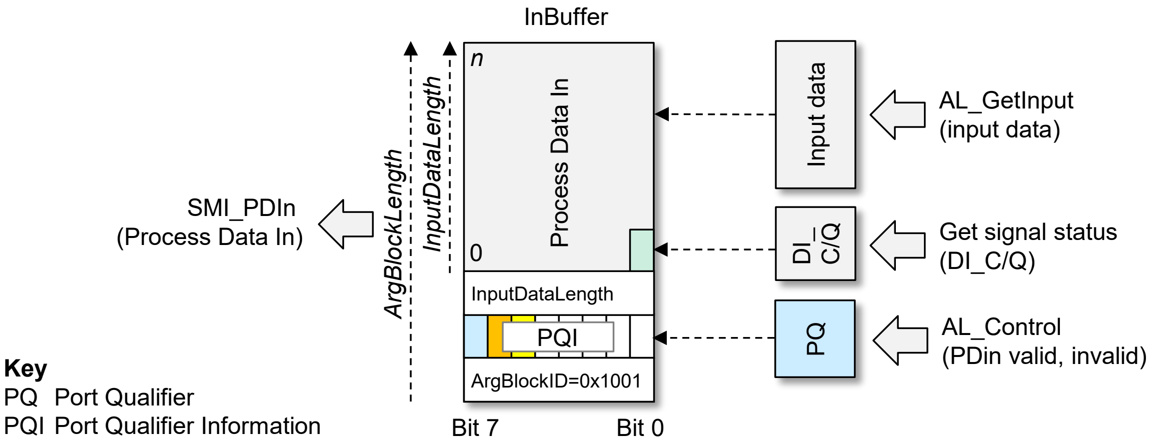


Figure 113 – Principles of Process Data Input mapping

At state OPERATE the input data are cyclicly copied into the InBuffer starting at offset "4".

The InBuffer is expanded by an octet "PQI" at offset "2", whose content shall be updated anytime the input data are read. Figure 114 illustrates the structure of this octet.

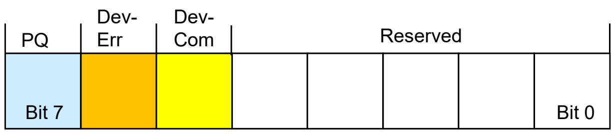


Figure 114 – Port Qualifier Information (PQI)

Bit 0 to 4: Reserved

These bits are reserved for future use.

Bit 5: DevCom

Parameter "PortStatusInfo" of service "SMI_PortStatus" provides the necessary information for this bit.

It will be set if a Device is detected and in OPERATE state. It will be reset if there is no Device available.

Bit 6: DevErr

Parameter "PortStatusInfo" and "DiagEntry x"of service "SMI_PortStatus" provide the necessary information for this bit.

It will be set if an Error or Warning occurred assigned to either Device or port. It will be reset if there is no Error or Warning.

Bit 7: Port Qualifier (PQ)

A value VALID for Process Data in service "AL_CONTROL" will set this bit.

A value INVALID or PortStatusInfo <> "4" (see E.4) will reset this bit.

11.7.2.2 Port Mode "DI_C/Q"

In this Port Mode the signal status of DI_C/Q will be mapped into octet 0, Bit 0 of the InBuffer (see Figure 113).

11.7.2.3 Port Mode "DEACTIVATED"

In this Port Mode the InBuffer will be filled with "0".

11.7.3 Process Data output mapping

11.7.3.1 Port Modes "IOL_MANUAL" or "IOL_AUTOSTART"

Master application "Process Data Exchange" (PDE) is responsible for the cyclic transfer of output data using the services "AL_SetOutput" (see 8.2.2.10) and "AL_Control" (see 8.2.2.12). Both shall be synchronized for consistency.

A gateway application can write data via the service "SMI_PDOut" into the OutBuffer (see 11.2.18). Figure 115 illustrates the principles of Process Data Output mapping and the content of the ArgBlock of this service (see E.11) consisting of the ArgBlockID, the Output Enable bit, the parameter OutputDataLength, and the output Process Data.

An ErrorType 0x4034 – *Incorrect ArgBlock length* will be returned if length does not add up to Process Data Out plus four octets (see C.4.9).

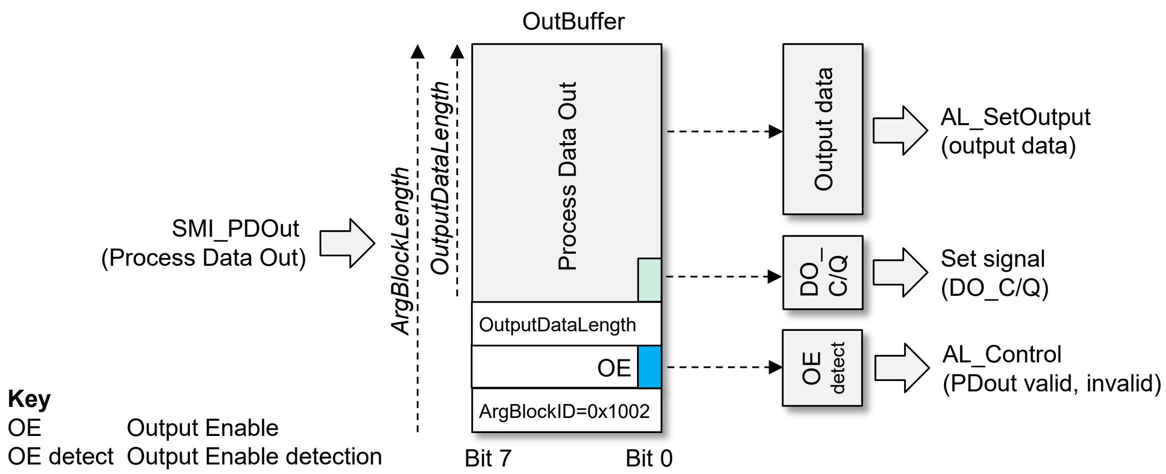


Figure 115 – Principles of Process Data Output mapping

At state OPERATE the Process Data Out are cyclicly copied to output data starting at offset "3".

The OutBuffer is expanded by an octet "OE" (Output Enable) at offset "2". Bit 0 indicates the validity of the Process Data Out. "0" means invalid, "1" means valid data. A change of this Bit from "0" to "1" will launch an AL_Control with "PDout valid". A change of this Bit from "1" to "0" will launch an AL_Control with "PDout invalid". See "OE detect" in Figure 115.

11.7.3.2 Port Mode: "DO_C/Q"

In this Port Mode octet 0, Bit 0 of the Process Data Out in the OutBuffer will be mapped into the signal status of DO_C/Q (see Figure 115).

11.7.4 Process Data invalid/valid qualifier status

A sample transmission of an output PD qualifier status "invalid" from Master AL to Device AL is shown in the upper section of Figure 116.

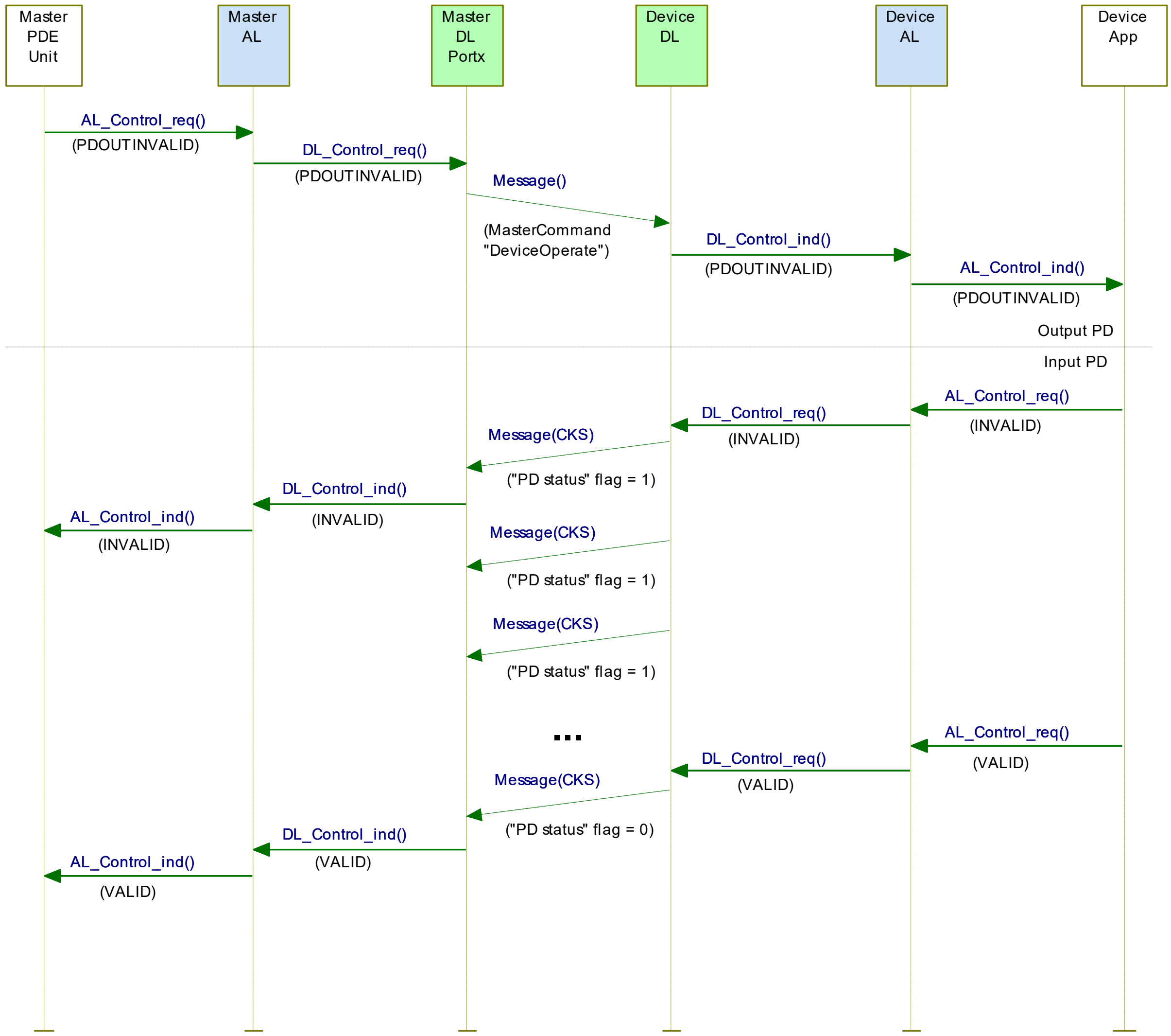


Figure 116 – Propagation of PD qualifier status between Master and Device

The Master informs the Device about the output Process Data qualifier status "valid/invalid" by sending MasterCommands (see Table B.2) to the Direct Parameter page 1 (see 7.3.7.1).

For input Process Data the Device sends the Process Data qualifier status in every single message as "PD status" flag in the Checksum / Status (CKS) octet (see A.1.5) of the Device message. A sample transmission of the input PD qualifier status "valid" from Device AL to Master AL is shown in the lower section of Figure 116.

Any perturbation while in interleave transmission mode leads to an input or output Process Data qualifier status "invalid" indication respectively.

11.8 Port power switching

The optional ability to switch the port power source allows to control the power consumption of the attached Device over time or may force a power down reset of the attached Device.

The Standardized Master Interface (SMI) provides the service SMI_PortPowerOffOn. The associated ArgBlock is defined in E.9, the dynamic behavior is shown in Figure 117.

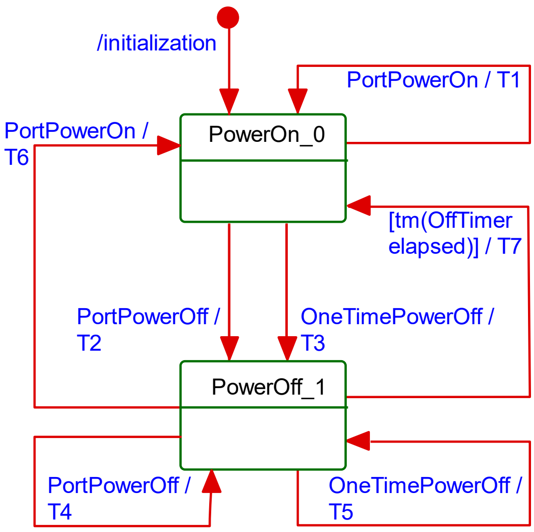


Figure 117 – Port power state machine

Table 129 shows the states and transitions of the Port power state machine.

Table 129 – States and Transitions of the Port power state machine

STATE NAME	STATE DESCRIPTION
PowerOn_0	Port power is switched on
PowerOff_1	Port power is switched off

TRANSITION	SOURCE STATE	TARGET STATE	ACTION
T1	0	0	-
T2	0	1	Switch Port power off
T3	0	1	Switch Port power off, start OffTimer with PowerOffTime
T4	1	1	Stop Timer
T5	1	1	Restart OffTimer with PowerOffTime
T6	1	0	Switch Port Power on, stop OffTimer
T7	1	0	Switch Port power on

INTERNAL ITEMS	TYPE	DEFINITION
PortPowerOn	Call	Received SMI_PowerOnOff with PortPowerMode “SwitchPowerOn”
PortPowerOff	Call	Received SMI_PowerOnOff with PortPowerMode “SwitchPowerOff”
OneTimePowerOff	Call	Received SMI_PowerOnOff with PortPowerMode “OneTimeSwitchOff”
OffTimer	Variable	Timer to schedule the power reactivation

12 Holistic view on Data Storage

12.1 User point of view

In this clause the Data Storage mechanism is described from a holistic user's point of view as best practice pattern. This is in contrast to clause 10.4 and 11.4 where Device and Master are described separately and each with more features then used within the recommended concept herein after.

12.2 Operations and preconditions

12.2.1 Purpose and objectives

Main purpose of the IO-Link Data Storage mechanism is the replacement of obviously defect Devices or Masters by spare parts (new or used) without using configuration, parameterzation, or other tools. The scenarios and associated preconditions are described in the following clauses.

12.2.2 Preconditions for the activation of the Data Storage mechanism

The following preconditions shall be observed prior to the usage of Data Storage:

a) Data Storage is only available for Devices and Masters implemented according to this document (≥ *V1.1*).

b) The Inspection Level of that Master port, the Device is connected to shall be adjusted to "type compatible" (corresponds to "TYPE_COMP" within Table 80)

c) The Backup Level of that Master port, the Device is connected to shall be either "Backup/Restore" or "Restore", which corresponds to DS_Enabled in 11.4.4. See 12.4 within this document for details on Backup Level.

12.2.3 Preconditions for the types of Devices to be replaced

After activation of a Backup Level (Data Storage mechanism) a "faulty" Device can be replaced by a type equivalent or compatible other Device. In some exceptional cases, for example non-calibrated Devices, a user manipulation can be required such as teach-in, to guarantee the same functionality and performance.

Thus, two classes of Devices exist in respect to exchangeability, which shall be described in the user manual of the particular Device:

Data Storage class 1: automatic DS

The configured Device supports Data Storage in such a manner that the replacement Device plays the role of its predecessor fully automatically and with the same performance.

Data Storage class 2: semi-automatic DS

The configured Device supports Data Storage in such a manner that the replacement Device requires user manipulation such as teach-in prior to operation with the same performance.

The Data Storage class shall be described in the user manual of the Device. Device designer is responsible in case of class 2 to prevent from dangerous system restart after Device replacement, at least via descriptions within the user manual.

12.2.4 Preconditions for the parameter sets

Each Device operates with the configured set of active parameters. The associated set of backup parameters stored within the system (Master and upper level system, for example PLC) can be different from the set of active parameters (see Figure 118).

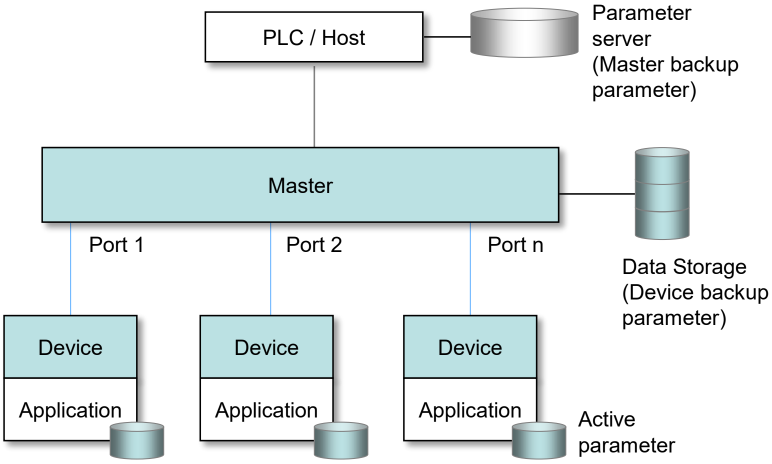


Figure 118 – Active and backup parameter

A replacement of the Device in operation will result in overwriting the active parameter set with the backup parameters in the newly connected Device.

12.3 Commissioning

12.3.1 On-line commissioning

Usually, the Devices are configured and parameterized along with the configuration and parameterization of the fieldbus and PLC system with the help of engineering tools. After the user assigned values to the parameters, they are downloaded into the Device and become active parameters. Upon the system command "ParamDownloadStore", these parameters are uploaded (copied) into the Data Storage within the Master, which in turn will initiate a backup of all its parameters depending on the features of the upper level system.

12.3.2 Off-site commissioning

Another possibility is the configuration and parameterization of Devices with the help of extra tools such as "USB-Masters" and the IODD of the Device away (off-site) from the machine/ facility (see Figure 119).

The USB-Master tool will mark the parameter set after configuration, parameterization, and validation (to become "active") via DS_UPLOAD_FLAG (see Table 131 and Table B.10). After installation into the machine/facility these parameters are uploaded (copied) automatically into the Data Storage within the Master (backup).

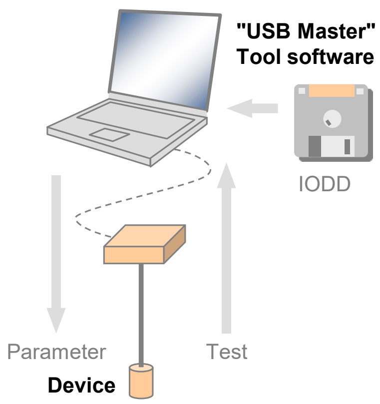


Figure 119 – Off-site commissioning

12.4 Backup Levels

12.4.1 Purpose

Within automation projects including IO-Link usually three situations with different user requirements for backup of parameters via Data Storage can be identified:

- Commissioning ("Disable");
- Production ("Backup/Restore");
- Production ("Restore").

Accordingly, three different "Backup Levels" are defined allowing the user to adjust the system to the particular functionality such as for Device replacement, off-site commissioning, parameter changes at runtime, etc. (see Table 130).

These adjustment possibilities lead for example to drop-down menu entries for "Backup Level".

12.4.2 Overview

Table 130 shows the recommended practice for Data Storage within an IO-Link system. It simplifies the activities and their comprehension since activation of the Data Storage implies transfer of the parameters.

Table 130 – Recommended Data Storage Backup Levels

Backup Level	Data Storage adjustments	Behavior
Commissioning ("Disable")	Master port: Activation state: "DS_Cleared"	Any change of active parameters within the Device will not be copied/saved. Device replacement without automatic/semi-automatic Data Storage.
Production ("Backup/Restore")	Master port: Activation state: "DS_Enabled" Master port: UploadEnable Master port: DownloadEnable	Changes of active parameters within the Device will be copied/saved. Device replacement with automatic/semi-automatic Data Storage supported.
Production ("Restore")	Master port: Activation state: "DS_Enabled" Master port: UploadDisable Master port: DownloadEnable	Any change of active parameters within the Device will not be copied/saved. If the parameter set is marked to be saved, the "frozen" parameters will be restored by the Master. However, Device replacement with automatic/semi-automatic Data Storage of "frozen" parameters is supported.

Legacy rules and presetting:

- For (legacy) Devices according to [8] or Devices according to this document where the Port is preset to Inspection Level "NO_CHECK", only the Backup Level "Commissioning" shall be supported. This should also be the default presetting in this case.
- For Devices according to this document where the Port is preset to Inspection Level "TYPE_COMP" all three Backup Levels shall be supported. Default presetting in this case should be "Backup/Restore".

The following clauses describe the phases in detail.

12.4.3 Commissioning ("Disable")

Data Storage is disabled in Master port configuration, where configurations, parameterizations, and PLC programs are fine-tuned, tested, and verified. This includes the involved IO-Link Masters and Devices. Usually, repeated saving (uploading) of the active Device parameters makes no sense in this phase. As a consequence, the replacement of Master and Devices with automatic/semi-automatic Data Storage is not supported.

12.4.4 Production ("Backup/Restore")

Data Storage in Master port configuration will be enabled. Current active parameters within the Device will be copied/saved as backup parameters. Device replacement with automatic/semi-automatic Data Storage is now supported via download/copy of the backup parameters to the Device and thus turning them into active parameters.

Criteria for the particular copy activities are listed in Table 131. These criteria are the conditions to trigger a copy process of the active parameters to the backup parameters, thus ensuring the consistency of these two sets.

Table 131 – Criteria for backing up parameters ("Backup/Restore")

User action	Operations	Data Storage
Commissioning session (see 12.3.1)	Parameterization of the Device via Master tool (on-line). Transfer of active parameter(s) to the Device will cause backup activity.	Master tool sends ParamDownloadStore; Device sets "DS_UPLOAD_FLAG" and then triggers upload via "DS_UPLOAD_REQ" Event. "DS_UPLOAD_FLAG" is reset as soon as the upload is completed.
Switching from commissioning to production	Restart of Port and Device because Port configuration has been changed	During system startup, the "DS_UPLOAD_FLAG" triggers upload (copy). "DS_UPLOAD_FLAG" is reset as soon as the upload is completed
Local modifications	Changes of the active parameters through teach-in or local parameterzation at the Device (on-line)	Device technology application sets "DS_UPLOAD_FLAG" and then triggers upload via "DS_UPLOAD_REQ" Event. "DS_UPLOAD_FLAG" is reset as soon as the upload is completed.
Off-site commissioning (see 12.3.2)	Phase 1: Device is parameterized off-site via USB-Master tool (see Figure 119). Phase 2: Connection of that Device to a Master port.	Phase 1: USB-Master tool sends ParamDownloadStore; Device sets "DS_UPLOAD_FLAG" (in non-volatile memory) and then triggers upload via "DS_UPLOAD_REQ" Event, which is ignored by the USB-Master. Phase 2: During system startup, the "DS_UPLOAD_FLAG" triggers upload (copy). "DS_UPLOAD_FLAG" is reset as soon as the upload is completed.
Changed port configuration (in case of "Backup-/Restore" or "Restore")	Whenever relevant port configuration has been changed via Master tool (on-line): see 11.4.4.	Change of relevant port configuration triggers "DS_Delete" followed by an upload (copy) to Data Storage (see 13.4.1, 11.3.1 and 11.4.4).
PLC program demand	Parameter change via user program followed by a SystemCommand	User program sends SystemCommand ParamDownloadStore; Device sets "DS_UPLOAD_FLAG" and then triggers upload via "DS_UPLOAD_REQ" Event. "DS_UPLOAD_FLAG" is reset as soon as the upload is completed.
Device reset (see 10.7)	Parameter change using one of the reset options in 10.7	See Table 101
NOTE For details on "DS_UPLOAD_FLAG" see 11.4.4		

12.4.5 Production ("Restore")

Data Storage in Master port configuration is enabled. However, only DS_Download operation is available. This means, unintended overwriting of Data Storage within the Master is prohibited.

Any changes of the active parameters through teach-in, tool based parameterization, or local parameterization will lead to a Data Storage Event, and State Property "DS_UPLOAD_FLAG" will be set in the Device.

In back-up level Production ("Restore") the Master shall ignore this flag and shall issue a DS_Download to overwrite the changed parameters.

Criteria for the particular copy activities are listed in Table 132. These criteria are the conditions to trigger a copy process of the active parameters to the backup parameters, thus ensuring the consistency of these two sets.

Table 132 – Criteria for backing up parameters ("Restore")

User action	Operations	Data Storage
Change port configuration	Change of relevant port configuration via Master tool (on-line): see 11.4.4	Change of relevant port configuration triggers "DS_Delete" followed by an

		upload (copy) to Data Storage (see 13.4.1, 11.3.1 and 11.4.4).

12.5 Use cases

12.5.1 Device replacement (@ "Backup/Restore")

The stored (saved) set of back-up parameters overwrites the active parameters (e.g. factory settings) within the replaced compatible Device of same type. This one operates after a restart with the identical parameters as with its predecessor.

The preconditions for this use case are

a) Devices and Master port adjustments according to 12.2.2;

b) *Backup Level:* "Backup/Restore"

c) The replacement Device shall be re-initiated to "factory settings" in case it is not a new Device out of the box (for "Back-to-box" see 10.7.5)

12.5.2 Device replacement (@ "Restore")

The stored (saved) set of back-up parameters overwrites the active parameters (e.g. factory settings) within the replaced compatible Device of same type. This one operates after a restart with the identical parameters as with its predecessor.

The preconditions for this use case are

a) Devices and Master port adjustments according to 12.2.2;

b) *Backup Level:* "Restore"

12.5.3 Master replacement

12.5.3.1 General

This feature depends heavily on the implementation and integration concept of the Master designer and manufacturer as well as on the features of the upper level system (fieldbus).

12.5.3.2 Without fieldbus support (base level)

Principal approach for a replaced (new) Master using a Master tool:

c) Set port configurations: amongst others the *Backup Level* to "Backup/Restore" or "Restore"

d) Master "reset to factory settings": clear backup parameters of all ports within the Data Storage in case it is not a new Master out of the box

e) Active parameters of all Devices are automatically uploaded (copied) to Data Storage (backup)

12.5.3.3 Fieldbus support (comfort level)

Any kind of fieldbus specific mechanism to back up the Master parameter set including the Data Storage of all Devices is used. Even though these fieldbus mechanisms are similar to the IO-Link approach, they are following their certain paradigm which may conflict with the described paradigm of the IO-Link back up mechanism (see Figure 118).

12.5.3.4 PLC system

The Device and Master parameters are stored within the system specific database of the PLC and downloaded to the Master at system startup after replacement.

This top down concept may conflict with the active parameter setting within the Devices.

12.5.4 Project replication

Following the concept of 12.5.3.3, the storage of complete Master parameter sets within the parameter server of an upper level system can automatically initiate the configuration of Masters and Devices besides any other upper level components and thus support the automatic replication of machines.

Following the concept of 12.5.3.4, after supply of the Master by the PLC, the Master can supply the Devices.

13 Integration

13.1 Generic Master model for system integration

Figure 120 shows the integration relevant excerpt of Figure 95. Basis is the Standardized Master Interface (SMI), which is specified in an abstract manner in 11.2. It transforms SDCI objects into services and objects appropriate for the upper level systems such as embedded controllers, IT systems (JSON), fieldbuses and PLCs, engineering systems, as well as universal Master Tools (PDCT) for Masters of different brands.

It is an objective of this SMI to achieve uniform behavior of Masters of different brands from a user's point of view. Another objective is to provide a stringent specification for organizations developing integration specifications into their systems without administrative overhead.

In Figure 120, the green marked items are areas of responsibility of fieldbus organizations and their integration specifications. The blue marked items are areas of responsibility of IT organizations and their specifications. The red marked items are areas of responsibility of individual automation equipment manufacturers. The white marked item ("Gateway management") represents a coordination layer for the different gateway applications. A corresponding specification is elaborated by a joint working group [12].

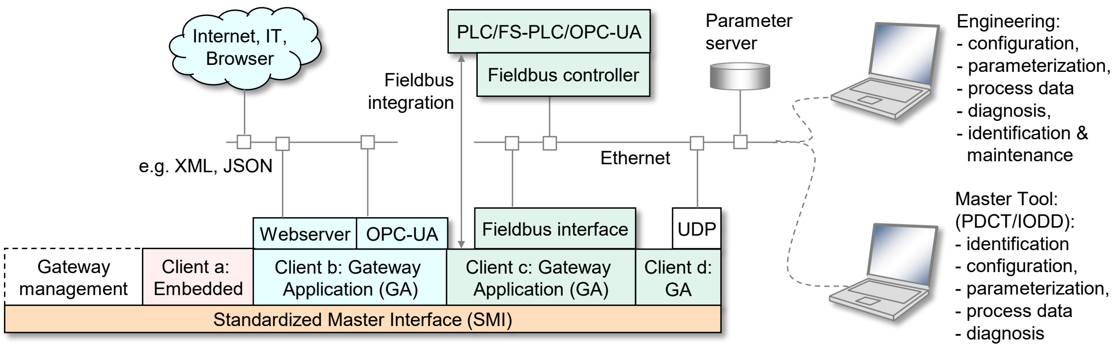


Figure 120 – Generic Master Model for system integration

13.2 Role of gateway applications

13.2.1 Clients

It is the role of gateway applications to provide translations of SMI services into the target systems (clients). Table 105 provides an overview of specified mandatory and optional SMI services. The designer of a gateway application determines the SMI service call technology.

Gateway applications such as shown in Figure 120 include but are not limited to:

- Pure coding tasks of the abstract SMI services, for example for embedded controllers;
- Comfortable webserver providing text and data for standard browsers using for example XML, JSON;
- OPC-UA server used for parameterization and data exchange via IT applications; security solutions available;
- Adapters with a fieldbus interface for programmable logic controllers (PLCs) and human machine interfaces based on OPC-UA;
- Adapters for a User Datagram Protocol (UDP) to connect engineering tools.

13.2.2 Coordination

It is the responsibility of gateway applications to prevent from access conflicts such as

- Different clients to one Device

- Concurrent tasks for one Device, for example prevent from SystemCommand "Restore factory settings" while Block Parameterization is running.

13.3 Security

The aspect of security is important whenever access to Master and Device data is involved. In case of fieldbuses most of the fieldbus organizations provide dedicated guidelines on security. In general, the IEC 62443 series is an appropriate source of protection strategies for industrial automation applications.

13.4 Special gateway applications

13.4.1 Changing Device configuration including Data Storage

After each relevant change of Device configuration/parameterization, the associated previously stored data set within the Master shall be cleared or marked invalid via the variable DS_Delete. Relevant changes via PortConfigList are:

– Change of CVID,

– Change of CDID,

– Change of Validation&Backup except changes between "Backup + Restore" and "Restore",

– Change of PortMode.

13.4.2 Parameter server and recipe control

The Master may combine the entire parameter sets of the connected Devices together with all other relevant data for its own operation and make this data available for upper level applications. For example, this data may be saved within a parameter server which may be accessed by a PLC program to change recipe parameters, thus supporting flexible manufacturing.

NOTE The structure of the data exchanged between the Master and the parameter server is outside the scope of this document.

13.5 Port and Device Configuration Tool (PDCT)

13.5.1 Strategy

Figure 120 demonstrates the necessity of a tool to configure ports, parameterize the Device, display diagnosis information, and provide identification and maintenance information. Depending on the degree of integration into a fieldbus system, the PDCT functions can be reduced, for example if the port configuration can be achieved via the field device description file of the particular fieldbus (engineering).

13.5.2 Accessing Masters via SMI

Figure 121 illustrates sample sequences of a standardized PDCT access to Masters (SMI). The Standardized Master Interface is specified in 11.2.

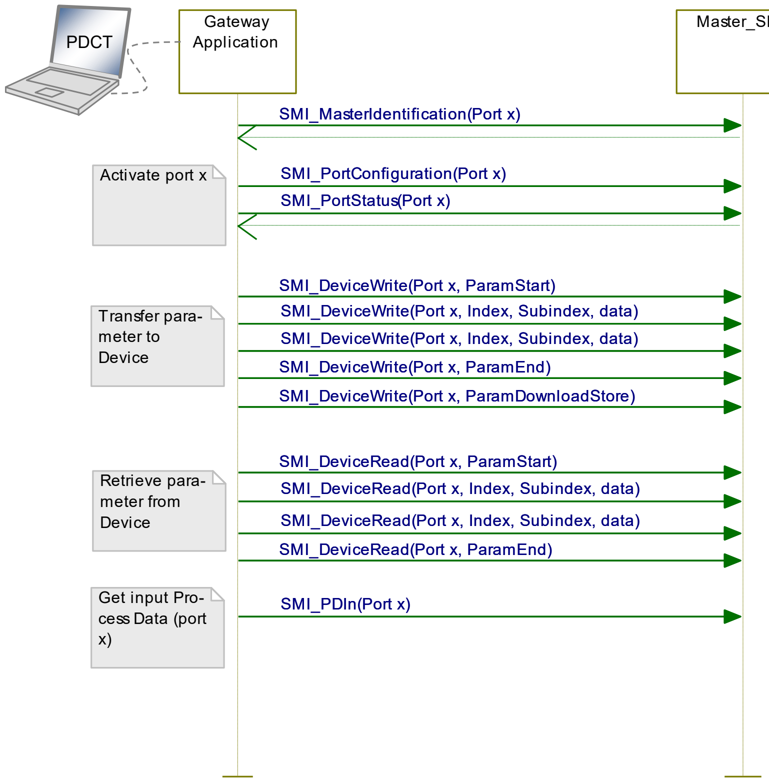


Figure 121 – PDCT via gateway application

13.5.3 Basic layout examples

Figure 122 shows one example of a PDCT display layout.

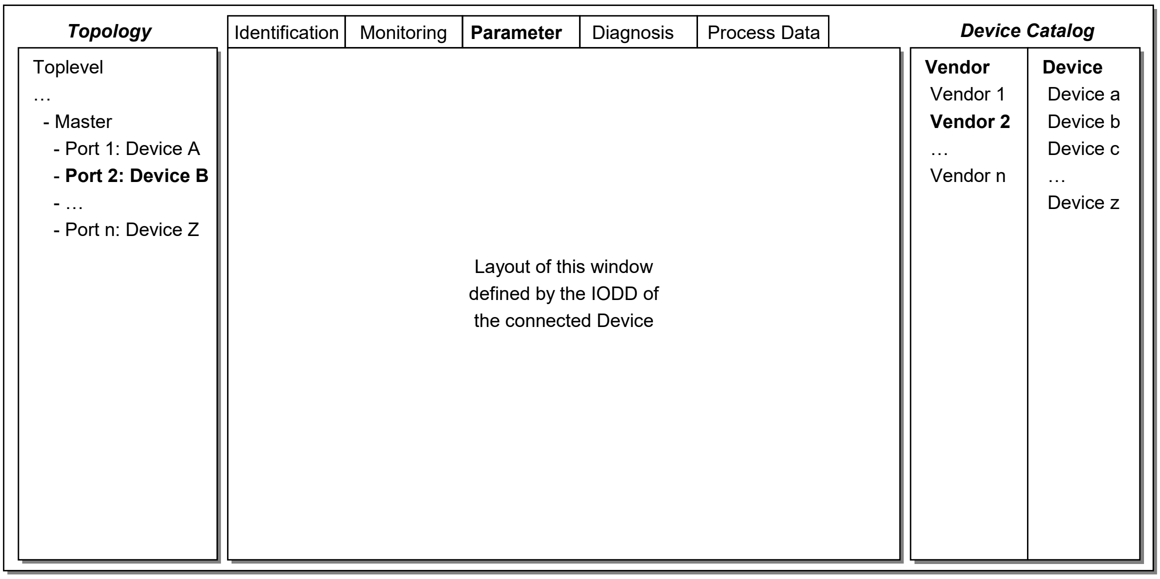


Figure 122 – Example 1 of a PDCT display layout

The PDCT display should always provide a navigation window for a project or a network topology, a window for the particular view on a chosen Device that is defined by its IODD, and a window for the available Devices based on the installed IODD files.

Figure 123 shows another example of a PDCT display layout.

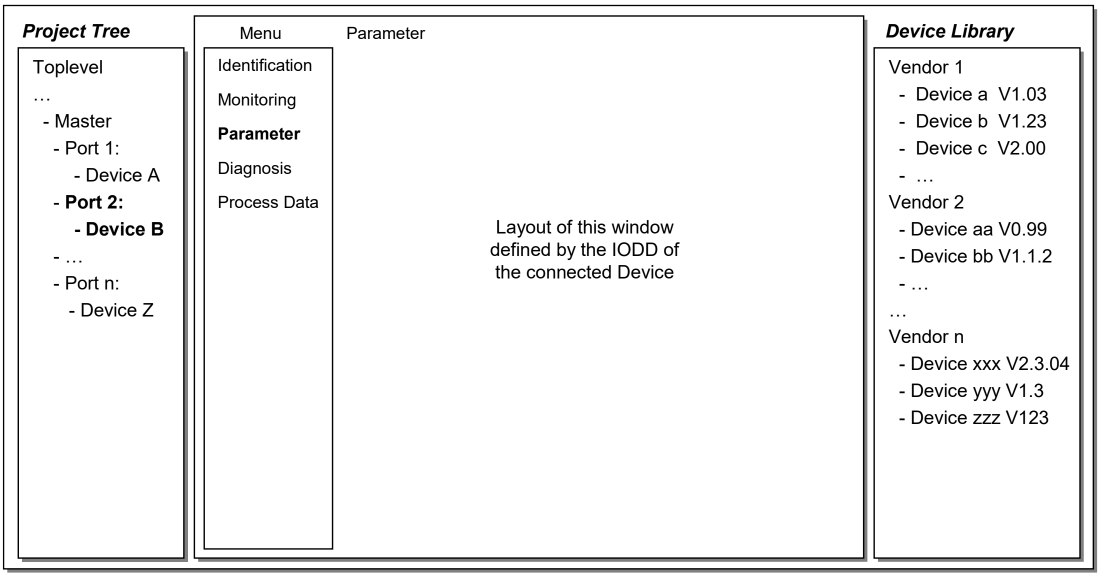


Figure 123 – Example 2 of a PDCT display layout

NOTE Further information can be retrieved from IEC/TR 62453-61.

Annex A
(normative)

Codings, timing constraints, and errors

A.1 General structure and encoding of M-sequences

A.1.1 Overview

The general concept of M-sequences is outlined in 7.3.3.2. Subclauses A.1.2 to A.1.6 provide a detailed description of the individual elements of M-sequences.

A.1.2 M-sequence control (MC)

The Master indicates the manner the user data (see A.1.4) shall be transmitted in an M-sequence control octet. This indication includes the transmission direction (read or write), the communication channel, and the address (offset) of the data on the communication channel. The structure of the M-sequence control octet is shown in Figure A.1.

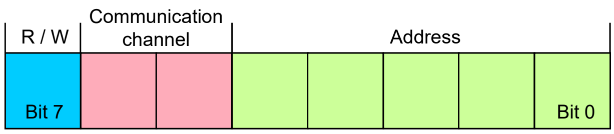


Figure A.1 – M-sequence control

Bit 0 to 4: Address

These bits indicate the address, i.e. the octet offset of the user data on the specified communication channel (see also Table A.1). In case of an ISDU channel, these bits are used for flow control of the ISDU data. The address, which means in this case the position of the user data within the ISDU, is only available indirectly (see 7.3.6.2).

Bit 5 to 6: Communication channel

These bits indicate the communication channel for the access to the user data. The defined values for the communication channel parameter are listed in Table A.1.

Table A.1 – Values of communication channel

Value	Definition
0	Process
1	Page
2	Diagnosis
3	ISDU

Bit 7: R/W

This bit indicates the transmission direction of the user data on the selected communication channel, i.e. read access (transmission of user data from Device to Master) or write access (transmission of user data from Master to Device). The defined values for the R/W parameter are listed in Table A.2.

Table A.2 – Values of R/W

Value	Definition
0	Write access
1	Read access

A Device is not required to support each and every of the 256 values of the M-sequence control octet. For read access to not implemented addresses or communication channels the value "0" shall be returned. A write access to not implemented addresses or communication channels shall be ignored.

A.1.3 Checksum / M-sequence type (CKT)

The M-sequence type is transmitted together with the checksum in the check/type octet. The structure of this octet is demonstrated in Figure A.2.

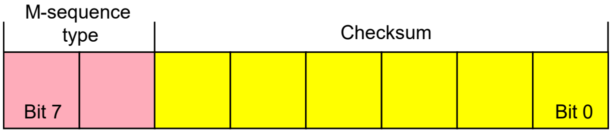


Figure A.2 – Checksum/M-sequence type octet

Bit 0 to 5: Checksum

These bits contain a 6 bit message checksum to ensure data integrity, see also A.1.6 and Clause I.1.

Bit 6 to 7: M-sequence type

These bits indicate the M-sequence type. Herewith, the Master specifies how the messages within the M-sequence are structured. Defined values for the M-sequence type parameter are listed in Table A.3.

Table A.3 – Values of M-sequence types

Value	Definition
0	Type 0
1	Type 1
2	Type 2 (see NOTE)
3	reserved
NOTE Subtypes depend on PD configuration and PD direction.	

A.1.4 User data (PD or OD)

User data is a general term for both Process Data and On-request Data. The length of user data can vary from 0 to 64 octets depending on M-sequence type and transmission direction (read/write). An overview of the available data types is shown in Table A.4. These data types can be arranged as records (different types) or arrays (same types).

Table A.4 – Data types for user data

Data type	Reference
BooleanT	See F.2
UIntegerT	See F.2.3
IntegerT	See F.2.4
StringT	See F.2.6
OctetStringT	See F.2.7
Float32T	See F.2.5
TimeT	See F.2.8
TimeSpanT	See F.2.9

The detailed coding of the data types can be found in Annex F.

A.1.5 Checksum / status (CKS)

The checksum/status octet is part of the reply message from the Device to the Master. Its structure is shown in Figure A.3. It comprises a 6-bit checksum, a flag to indicate valid or invalid Process Data, and an Event flag.

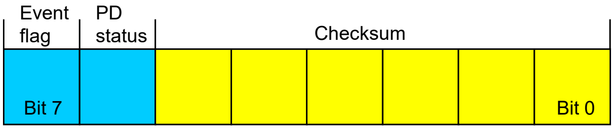


Figure A.3 – Checksum/status octet

Bit 0 to 5: Checksum

These bits contain a 6-bit checksum to ensure data integrity of the reply message. See also A.1.6 and Clause I.1.

Bit 6: PD status

This bit indicates whether the Device can provide valid Process Data or not. Defined values for the parameter are listed in Table A.5.

This PD status flag shall be used for Devices with input Process Data. Devices with only output Process Data shall always indicate "Process Data valid".

If the PD status flag is set to “Process Data invalid” within a message, all the input Process Data of the complete Process Data cycle are invalid.

Table A.5 – Values of PD status

Value	Definition
0	Process Data valid
1	Process Data invalid

Bit 7: Event flag

This bit indicates a Device initiative for the data category "Event" to be retrieved by the Master via the diagnosis communication channel (see Table A.1). The Device can report diagnosis information such as errors, warnings or notifications via Event response messages. Permissible values for the parameter are listed in Table A.6.

Table A.6 – Values of the Event flag

Value	Definition
0	No Event
1	Event

A.1.6 Calculation of the checksum

The message checksum provides data integrity protection for data transmission from Master to Device and from Device to Master. Each UART data octet is protected by the UART parity bit (see Figure 21). Besides this individual data octet protection, all of the UART data octets in a message are XOR (exclusive or) processed octet by octet. The check/type octet is included with checksum bits set to "0". The resulting checksum octet is compressed from 8 to 6 bit in accordance with the conversion procedure in Figure A.4 and its associated formulas (see equations in (A.1)). The 6 bit compressed "Checksum6" is entered into the checksum/ M-sequence type octet (see Figure A.2). The same procedure takes place to secure the message from the Device to the Master. In this case the compressed checksum is entered into the checksum/status octet (see Figure A.3).

A seed value of 0x52 is used for the checksum calculation across the message. It is XORed with the first octet of the message (MC).

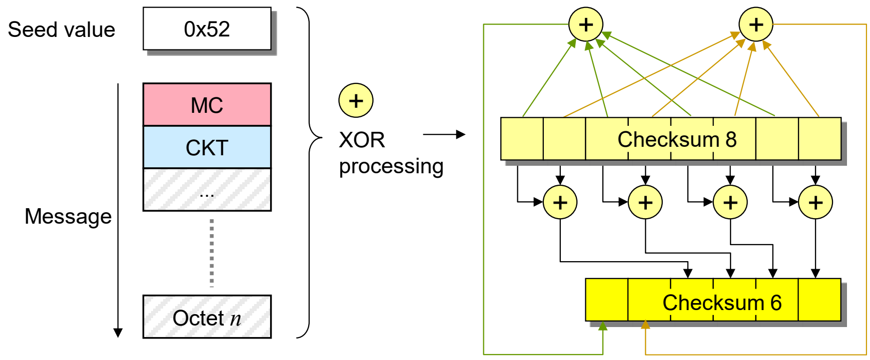


Figure A.4 – Principle of the checksum calculation and compression

The set of equations in (A.1) define the compression procedure from 8 to 6 bit in detail.

$$
\begin{aligned}
D5_6 &= D7_8 \text{ xor } D5_8 \text{ xor } D3_8 \text{ xor } D1_8 \\
D4_6 &= D6_8 \text{ xor } D4_8 \text{ xor } D2_8 \text{ xor } D0_8 \\
D3_6 &= D7_8 \text{ xor } D6_8 \\
D2_6 &= D5_8 \text{ xor } D4_8 \\
D1_6 &= D3_8 \text{ xor } D2_8 \\
D0_6 &= D1_8 \text{ xor } D0_8
\end{aligned}
\qquad (A.1)
$$

A.2 M-sequence types

A.2.1 Overview

Process Data and On-request Data use separate cyclic and acyclic communication channels (see Figure 8) to ensure scheduled and deterministic delivery of Process Data while delivery of On-request Data does not have consequences on the Process Data transmission performance.

Within SDCI, M-sequences provide the access to the communication channels via the M-sequence Control octet. The number of different M-sequence types meets the various requirements of sensors and actuators regarding their Process Data width. See Figure 39 for an overview of the available M-sequence types that are specified in A.2.2 to A.2.5. See A.2.6 for rules on how to use the M-sequence types.

A.2.2 M-sequence TYPE_0

M-sequence TYPE_0 is mandatory for all Devices. It only transmits On-request Data. One octet of user data is read or written per cycle. This M-sequence is shown in Figure A.5.

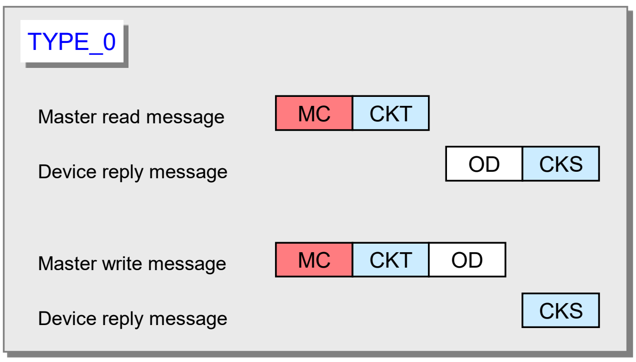


Figure A.5 – M-sequence TYPE_0

A.2.3 M-sequence TYPE_1_x

M-sequence TYPE_1_x is optional for all Devices.

M-sequence TYPE_1_1 is shown in Figure A.6.

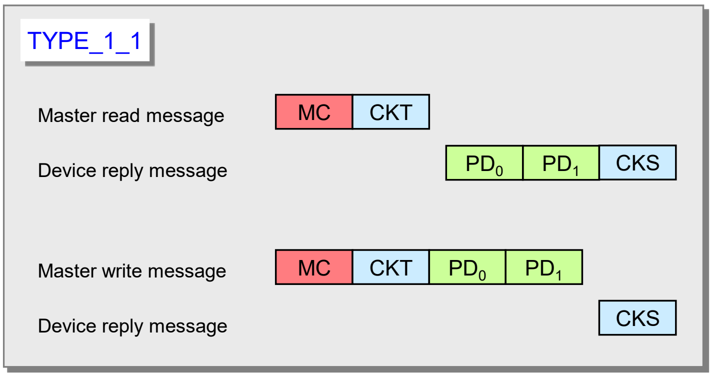


Figure A.6 – M-sequence TYPE_1_1

Two octets of Process Data are read or written per cycle. Address (bit offset) belongs to the process communication channel (see A.2.1).

In case of interleave mode (see 7.3.4.2) and odd-numbered PD length the remaining octets within the messages are padded with 0x00.

M-sequence TYPE_1_2 is shown in Figure A.7. Two octets of On-request Data are read or written per cycle.

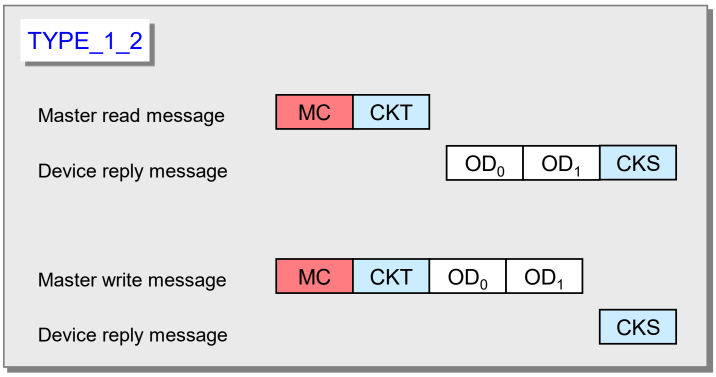


Figure A.7 – M-sequence TYPE_1_2

M-sequence TYPE_1_V providing variable (extendable) message length is shown in Figure A.8. A number of m octets of On-request Data are read or written per cycle.

When accessing octets via page and diagnosis communication channels using an M-sequence TYPE with multi-octet ODs, the following rules apply:

- At write access, only the first octet (OD_0) of On-request Data is relevant. The Master shall send all subsequent ODs filled with "0x00". Any Device shall evaluate only the first octet of ODs and ignore the remaining octets.
- At read access, the Device shall return the first relevant data octet as OD_0 and all subsequent ODs filled with either "0x00" or with subsequent data octets if appropriate. Master shall evaluate only the octet in OD_0.

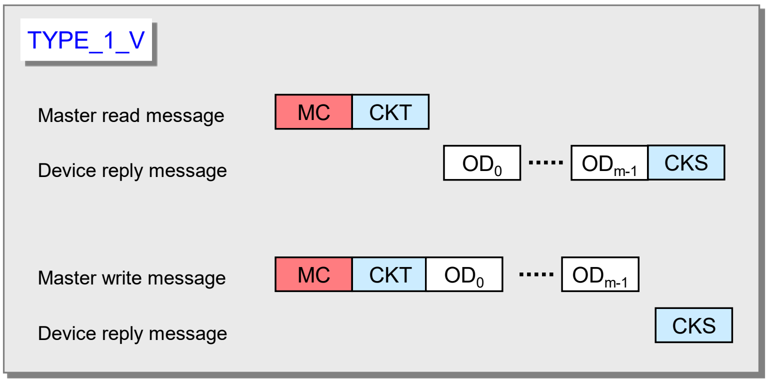


Figure A.8 – M-sequence TYPE_1_V

A.2.4 M-sequence TYPE_2_x

M-sequence TYPE_2_x is optional for all Devices. M-sequences TYPE_2_1 through TYPE_2_5 are defined. M-sequence TYPE_2_V provides variable (extendable) message length. M-sequence TYPE_2_x transmits Process Data and On-request Data in one message. The number of process and On-request Data read or written in each cycle depends on the type. The Address parameter (see Figure A.1) belongs in this case to the on-request communication channel. The Process Data address is specified implicitly starting at "0". The format of Process Data is characterizing the M-sequence TYPE_2_x.

M-sequence TYPE_2_1 transmits one octet of read Process Data and one octet of read or write On-request Data per cycle. This M-sequence type is shown in Figure A.9.

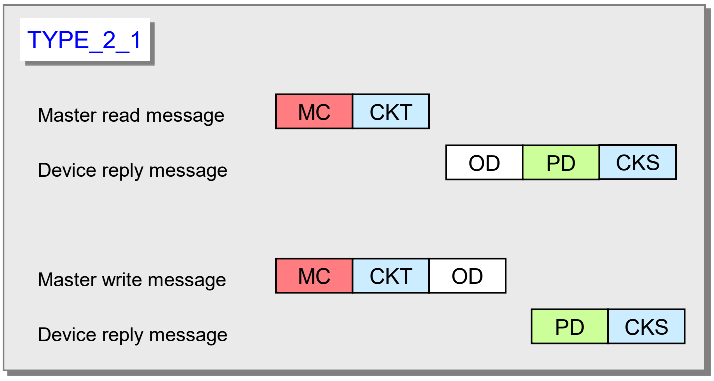


Figure A.9 – M-sequence TYPE_2_1

M-sequence TYPE_2_2 transmits 2 octets of read Process Data and one octet of On-request Data per cycle. This M-sequence type is shown in Figure A.10.

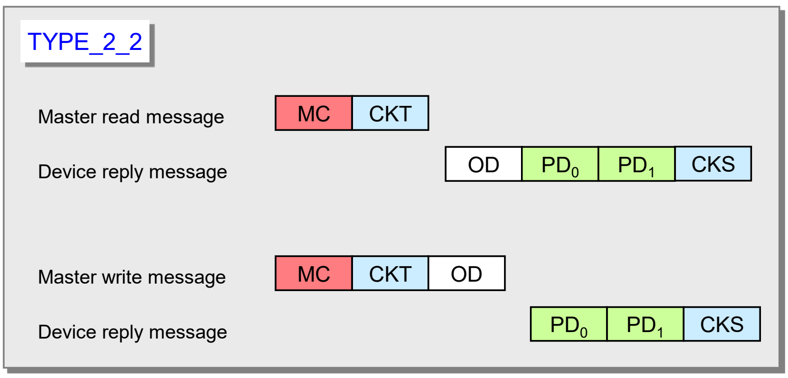


Figure A.10 – M-sequence TYPE_2_2

M-sequence TYPE_2_3 transmits one octet of write Process Data and one octet of read or write On-request Data per cycle. This M-sequence type is shown in Figure A.11.

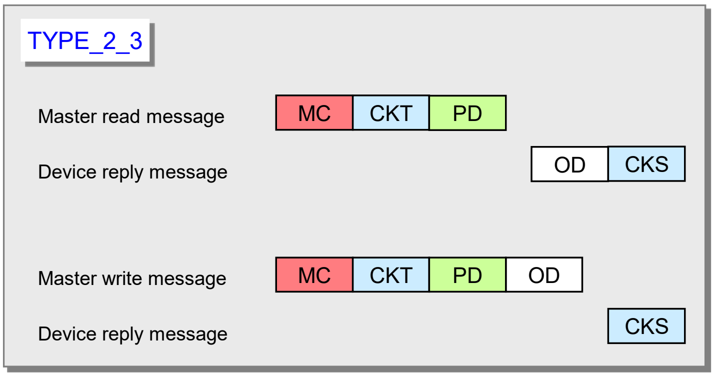


Figure A.11 – M-sequence TYPE_2_3

M-sequence TYPE_2_4 transmits 2 octets of write Process Data and one octet of read or write On-request Data per cycle. This M-sequence type is shown in Figure A.12

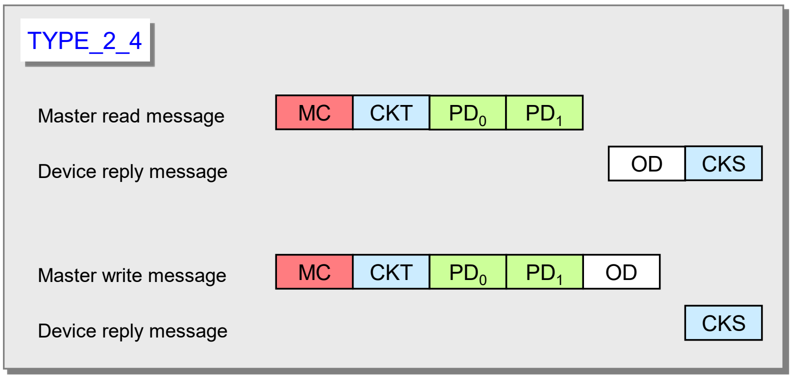


Figure A.12 – M-sequence TYPE_2_4

M-sequence TYPE_2_5 transmits one octet of write and read Process Data and one octet of read or write On-request Data per cycle. This M-sequence type is shown in Figure A.13.

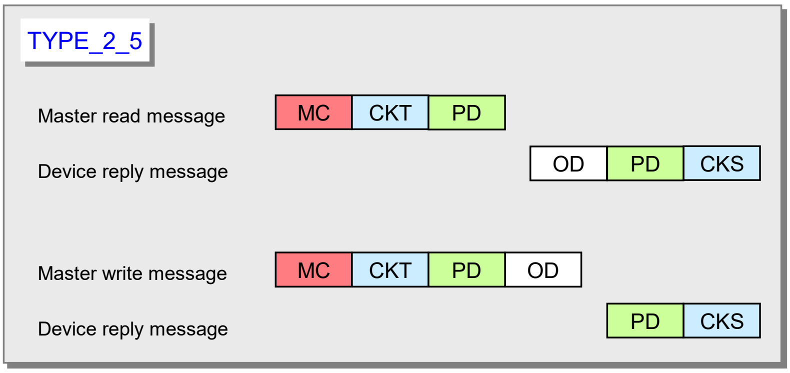


Figure A.13 – M-sequence TYPE_2_5

M-sequence TYPE_2_V transmits the entire write (read) ProcessDataIn n (k) octets per cycle. The range of n (k) is 0 to 32. Either PDin or PDout are not existing when n = 0 or k = 0. TYPE_2_V also transmits m octets of (segmented) read or write On-request Data per cycle using the address in Figure A.1. Permitted values for m are 1, 2, 8, and 32. This variable M-sequence type is shown in Figure A.14.

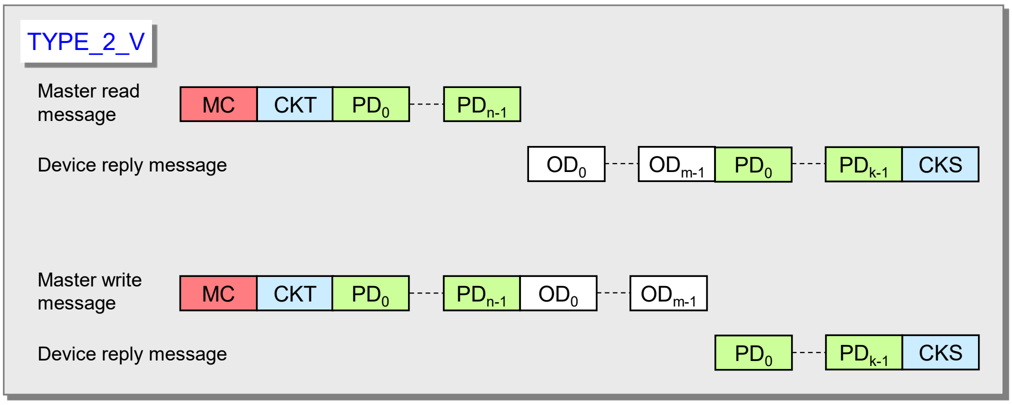


Figure A.14 – M-sequence TYPE_2_V

When using M-sequence TYPE with multi-octet ODs, the rules of M-sequence TYPE_1_V apply (see Figure A.8).

A.2.5 M-sequence type 3

M-sequence type 3 is reserved and shall not be used.

A.2.6 M-sequence type usage for STARTUP, PREOPERATE and OPERATE modes

Table A.7 lists the M-sequence types for the STARTUP mode together with the minimum recovery time (T_{initcyc}) that shall be observed for Master implementations (see A.3.9). The M-sequence code refers to the coding in B.1.4.

Table A.7 – M-sequence types for the STARTUP mode

STARTUP M-sequence code	On-request Data	M-sequence type	Minimum recovery time
	Octets		T_{BIT}
n/a	1	TYPE_0	100

Table A.8 lists the M-sequence types for the PREOPERATE mode together with the minimum recovery time (T_{initcyc}) that shall be observed for Master implementations.

Table A.8 – M-sequence types for the PREOPERATE mode

PREOPERATE M-sequence code	On-request Data	M-sequence type	Minimum recovery time [a]
	Octets		T_{BIT}
0 [b]	1	TYPE_0	100
1	2	TYPE_1_2	100
2	8	TYPE_1_V	210
3	32	TYPE_1_V	550

NOTE a The minimum recovery time in PREOPERATE mode is a requirement for the Master

NOTE b It is highly recommended for Devices not to use TYPE_0 thus improving error discovery when Master restarts communication

Table A.9 lists the M-sequence types for the OPERATE mode for legacy Devices. The minimum cycle time for Master in OPERATE mode is specified by the parameter "MinCycleTime" of the Device (see B.1.3).

4958 **Table A.9 – M-sequence types for the OPERATE mode (legacy protocol)**

OPERATE M-sequence code	On-request Data	Process Data (PD)		M-sequence type
	Octets	PDin	PDout	Legacy protocol (see [8])
0	1	0	0	TYPE_0 NOTE
1	2	0	0	TYPE_1_2
don't care	2	PDin + PDout > 2 octets		TYPE_1_1/1_2 (interleaved)
don't care	1	1…8 bit	0	TYPE_2_1
don't care	1	9…16 bit	0	TYPE_2_2
don't care	1	0	1…8 bit	TYPE_2_3
don't care	1	0	9…16 bit	TYPE_2_4
don't care	1	1…8 bit	1…8 bit	TYPE_2_5
NOTE It is highly recommended for Devices not to use TYPE_0 thus improving error discovery when Master restarts communication				

4959

Table A.10 lists the M-sequence types for the OPERATE mode for Devices according to this specification. The minimum cycle time for Master in OPERATE mode is specified by the parameter MinCycleTime of the Device (see B.1.3).

Table A.10 – M-sequence types for the OPERATE mode

OPERATE M-sequence code	On-request Data	Process Data (PD)		M-sequence type
	Octets	PDin	PDout	
0	1	0	0	TYPE_0 NOTE 1
1	2	0	0	TYPE_1_2
6	8	0	0	TYPE_1_V
7	32	0	0	TYPE_1_V
0	2	3..32 octets	0…32 octets	TYPE 1_1 / 1_2 interleaved NOTE 3
0	2	0…32 octets	3…32 octets	TYPE 1_1 / 1_2 interleaved NOTE 3
0	1	1…8 bit	0	TYPE_2_1
0	1	9…16 bit	0	TYPE_2_2
0	1	0	1…8 bit	TYPE_2_3
0	1	0	9…16 bit	TYPE_2_4
0	1	1…8 bit	1…8 bit	TYPE_2_5
0	1	9…16 bit	1…16 bit	TYPE_2_V NOTE 2
0	1	1…16 bit	9…16 bit	TYPE_2_V NOTE 2
4	1	0…32 octets	3…32 octets	TYPE_2_V
4	1	3…32 octets	0…32 octets	TYPE_2_V
5	2	>0 bit, octets	≥0 bit, octets	TYPE_2_V
5	2	≥0 bit, octets	>0 bit, octets	TYPE_2_V
6	8	>0 bit, octets	≥0 bit, octets	TYPE_2_V
6	8	≥0 bit, octets	>0 bit, octets	TYPE_2_V
7	32	>0 bit, octets	≥0 bit, octets	TYPE_2_V
7	32	≥0 bit, octets	>0 bit, octets	TYPE_2_V

NOTE1 It is highly recommended for Devices not to use TYPE_0 thus improving error discovery when Master restarts communication

NOTE2 Former TYPE_2_6 has been replaced in support of TYPE_2_V due to inefficiency.

NOTE3 Interleaved mode shall not be implemented in Devices, but shall be supported by Masters

A.3 Timing constraints

A.3.1 General

The interactions of a Master and its Device are characterized by several time constraints that apply to the UART frame, Master and Device message transmission times, supplemented by response, cycle, delay, and recovery times.

A.3.2 Bit time

The bit time T_{BIT} is the time it takes to transmit a single bit. It is the inverse value of the transmission rate (see equation (A.2)).

$$T_{BIT} = 1/(\text{transmission rate}) \qquad \text{(A.2)}$$

Values for T_{BIT} are specified in Table 9.

A.3.3 UART frame transmission delay of Master (ports)

The UART frame transmission delay t_1 of a port is the duration between the end of the stop bit of a UART frame and the beginning of the start bit of the next UART frame. The port shall transmit the UART frames within a maximum delay of one bit time (see equation (A.3)).

$$0 \leq t_1 \leq 1\ T_{BIT} \tag{A.3}$$

A.3.4 UART frame transmission delay of Devices

The Device’s UART frame transmission delay t_2 is the duration between the end of the stop bit of a UART frame and the beginning of the start bit of the next UART frame. The Device shall transmit the UART frames within a maximum delay of 3 bit times (see equation (A.4)).

$$0 \leq t_2 \leq 3\ T_{BIT} \tag{A.4}$$

A.3.5 Response time of Devices

The Device's response time t_A is the duration between the end of the stop bit of a port's last UART frame being received and the beginning of the start bit of the first UART frame being sent. The Device shall observe a delay of at least one bit time but no more than 10 bit times (see equation (A.5)).

$$1\ T_{BIT} \leq t_A \leq 10\ T_{BIT} \tag{A.5}$$

A.3.6 M-sequence time

Communication between a port and and its associated Device takes place in a fixed schedule, called the M-sequence time (see equation (A.6)).

$$t_{\text{M-sequence}} = (m+n) * 11 * T_{BIT} + t_A + (m-1) * t_1 + (n-1) * t_2 \tag{A.6}$$

In this formula, m is the number of UART frames sent by the port to the Device and n is the number of UART frames sent by the Device to the port. The formula can only be used for estimates as the times t_1 and t_2 may not be constant.

Figure A.15 demonstrates the timings of an M-sequence consisting of a Master (port) message and a Device message.

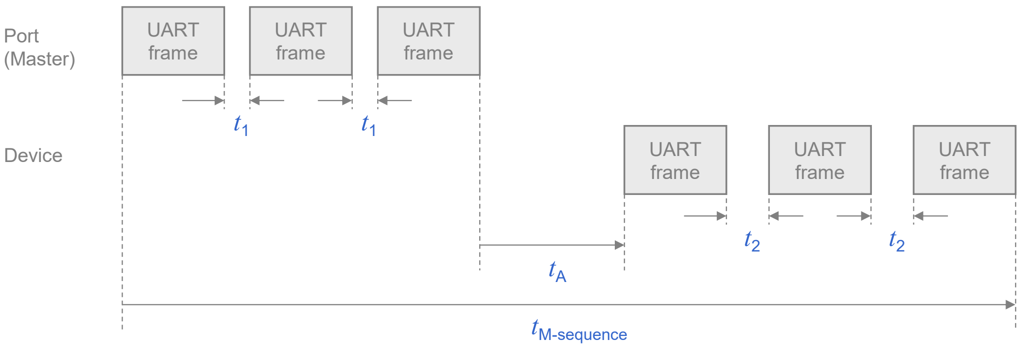


Figure A.15 – M-sequence timing

A.3.7 Cycle time

The cycle time t_{CYC} (see equation (A.7)) depends on the Device's parameter “MinCycleTime” and the design and implementation of a Master and the number of ports.

$$t_{CYC} = t_{\text{M-sequence}} + t_{idle} \tag{A.7}$$

The adjustable Device parameter “MasterCycleTime” can be used for the design of a Device specific technology such as an actuator to derive the timing conditions for a default appropriate action such as de-activate or de-energize the actuator (see 7.3.3.5 "MaxCycleTime", 10.2, and 10.8.3).

Table A.11 lists recommended minimum cycle time values for the specified transmission mode of a port. The values are calculated based on M-sequence Type_2_1.

Table A.11 – Recommended MinCycleTimes

Transmission mode	t_{CYC}
COM1	18,0 ms
COM2	2,3 ms
COM3	0,4 ms

A.3.8 Idle time

The idle time t_{idle} results from the configured cycle time t_{CYC} and the M-sequence time $t_{M\text{-}sequence}$. With reference to a port, it comprises the time between the end of the message of a Device and the beginning of the next message from the Master (port).

A.3.9 Recovery time

The Master shall wait for a recovery time $t_{initcyc}$ between any two subsequent acyclic Device accesses while in the STARTUP or PREOPERATE phase (see A.2.6). Recovery time is defined between the beginnings of two subsequent Master requests. Calculations shall refer to equation (A.7).

A.4 Errors and remedies

A.4.1 UART errors

A.4.1.1 Parity errors

The UART parity bit (see Figure 21) and the checksum (see A.1.6) are two independent mechanisms to secure the data transfer. This means that for example two bit errors in different octets of a message, which are resulting in the correct checksum, can also be detected. Both mechanisms lead to the same error processing.

Remedy: The Master shall repeat the Master message 2 times (see 7.2.2.1). Devices shall reject all data with detected errors and create no reaction.

A.4.1.2 UART framing errors

The conditions for the correct detection of a UART frame are specified in 5.3.3.2. Error processing shall take place whenever perturbed signal shapes or incorrect timings lead to an invalid UART stop bit.

Remedy: See A.4.1.1.

A.4.2 Wake-up errors

The wake-up current pulse is specified in 5.3.3.3 and the wake-up procedures in 7.3.2.1. Several faults may occur during the attempts to establish communication.

Remedy: Retries are possible. See 7.3.2.1 for details.

A.4.3 Transmission errors

A.4.3.1 Checksum errors

The checksum mechanism is specified in A.1.6. Any checksum error leads to an error processing.

Remedy: See A.4.1.1.

A.4.3.2 Timeout errors

The diverse timing constraints with M-sequences are specified in A.3. Master (ports) and Devices are checking several critical timings such as lack of synchronism within messages.

Remedy: See A.4.1.1.

A.4.3.3 Collisions

A collision occurs whenever the Master and Device are sending simultaneously due to an error. This error is interpreted as a faulty M-sequence.

Remedy: See A.4.1.1.

A.4.4 Protocol errors

A protocol error occurs for example whenever the sequence of the segmented transmission of an ISDU is wrong (see flow control case in A.1.2).

Remedy: Abort of service with ErrorType information (see Annex C).

A.5 General structure and encoding of ISDUs

A.5.1 Overview

The purpose and general structure of an ISDU is specified in 7.3.6.1. Subclauses A.5.2 to A.5.7 provide a detailed description of the individual elements of an ISDU and some examples.

A.5.2 I-Service

Figure A.16 shows the structure of the I-Service octet.

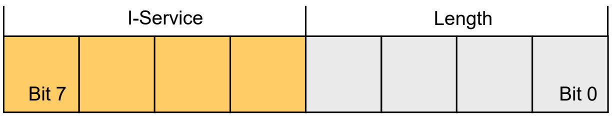


Figure A.16 – I-Service octet

Bits 0 to 3: Length
The encoding of the nibble Length of the ISDU is specified in Table A.14 .

Bits 4 to 7: I-Service
The encoding of the nibble I-Service of the ISDU is specified in Table A.12.

All other elements of the structure specified in 7.3.6.1 are transmitted as independent octets.

Table A.12 – Definition of the nibble "I-Service"

I-Service (binary)	Definition		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

I-Service (binary)	Definition		Index format
	Master	Device	
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	

Table A.13 specifies the syntax of the ISDUs. ErrorType can be found in Annex C.

Table A.13 – ISDU syntax

ISDU name	ISDU structure
Write Request	{I-Service(0x1), LEN, Index, [Data*], CHKPDU} ^ {I-Service(0x2), LEN, Index, Subindex, [Data*], CHKPDU} ^ {I-Service(0x3), LEN, Index, Index, Subindex, [Data*], CHKPDU}
Write Response (+)	I-Service(0x5), Length(0x2), CHKPDU
Write Response (-)	I-Service(0x4), Length(0x4), ErrorType, CHKPDU
Read Request	{I-Service(0x9), Length(0x3), Index, CHKPDU} ^ {I-Service(0xA), Length(0x4), Index, Subindex, CHKPDU} ^ {I-Service(0xB), Length(0x5), Index, Index, Subindex, CHKPDU}
Read Response (+)	I-Service(0xD), LEN, [Data*], CHKPDU
Read Response (-)	I-Service(0xC), Length(0x4), ErrorType, CHKPDU
Key LEN = {Length(0x1), ExtLength} ^ {Length}	

A.5.3 Extended length (ExtLength)

The number of octets transmitted in this I-Service, including all protocol information (6 octets), is specified in the “Length” element of an ISDU. If the total length is more than 15 octets, the length is specified using extended length information ("ExtLength"). Permissible values for “Length” and "ExtLength" are listed in Table A.14.

Table A.14 – Definition of nibble Length and octet ExtLength

I-Service	Length	ExtLength	Definition
0	0	n/a	No service, ISDU length is 1. Protocol use.
0	1	n/a	Device busy, ISDU length is 1. Protocol use.
0	2 to 15	n/a	Reserved and shall not be used
1 to 15	0	n/a	Reserved and shall not be used
1 to 15	1	0 to 16	Reserved and shall not be used
1 to 15	1	17 to 238	Length of ISDU in "ExtLength"
1 to 15	1	239 to 255	Reserved and shall not be used
1 to 15	2 to 15	n/a	Length of ISDU

A.5.4 Index and Subindex

The parameter address of the data object to be transmitted using the ISDU is specified in the “Index” element. “Index” has a range of values from 0 to 65535 (see B.2.1 for constraints). Index values 0 and 1 shall be rejected by the Device.

There is no requirement for the Device to support all Index and Subindex values. The Device shall send a negative response to Index or Subindex values not supported.

The data element address of a structured parameter of the data object to be transmitted using the ISDU 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 data object (see Figure 6).

Table A.15 lists the Index formats used in the ISDU depending on the parameters transmitted.

Table A.15 – Use of Index formats

Index	Subindex	Index format of ISDU
0 to 255	0	8 bit Index
0 to 255	1 to 255	8 bit Index and 8 bit Subindex
256 to 65535	0 to 255	16 bit Index and 8 bit Subindex (see NOTE)
NOTE See B.2.1 for constraints on the Index range		

A.5.5 Data

The “Data” element can contain the data objects specified in Annex B or Device specific data objects respectively. The data length corresponds to the entries in the “Length” element minus the ISDU protocol elements.

A.5.6 Check ISDU (CHKPDU)

The “CHKPDU” element provides data integrity protection. The sender calculates the value of "CHKPDU" by XOR processing all of the octets of an ISDU, including "CHKPDU" with a preliminary value "0", which is then replaced by the result of the calculation (see Figure A.17).

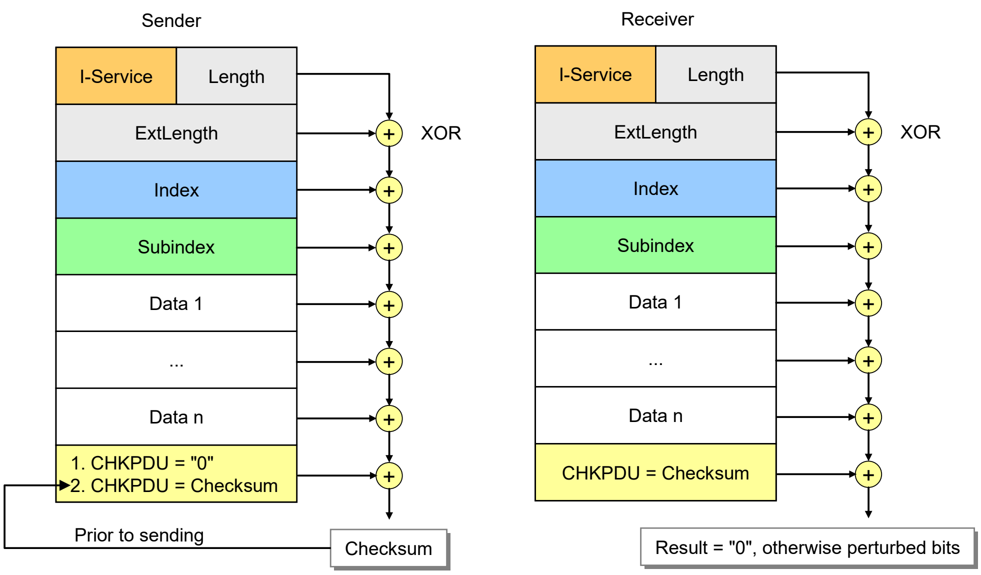
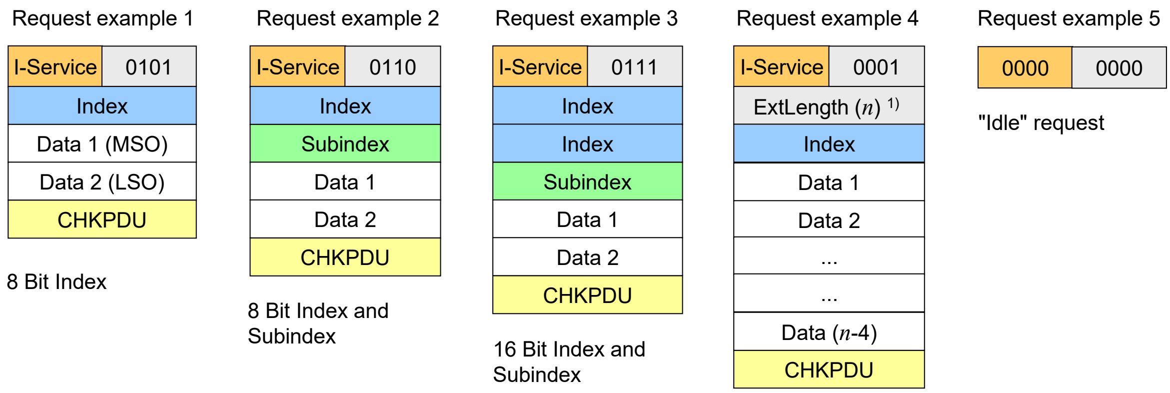


Figure A.17 – Check of ISDU integrity via CHKPDU

The receiver checks whether XOR processing of all of the octets of the ISDU will lead to the result "0" (see Figure A.17). If the result is different from "0", error processing shall take place. See also A.1.6.

A.5.7 ISDU examples

Figure A.18 demonstrates typical examples of request formats for ISDUs, which are explained in the following paragraphs.



1) Overall ISDU ExtLength = n (17 to 238); Length = 1 ("0001")

Figure A.18 – Examples of request formats for ISDUs

The ISDU request in example 1 comprises one Index element allowing addressing from 0 to 255 (see Table A.15 and Table B.8 for restrictions). In this example the Subindex is "0" and the whole content of Index is Data 1 with the most significant octet (MSO) and Data 2 with the least significant octet (LSO). The total length is 5 ("0101").

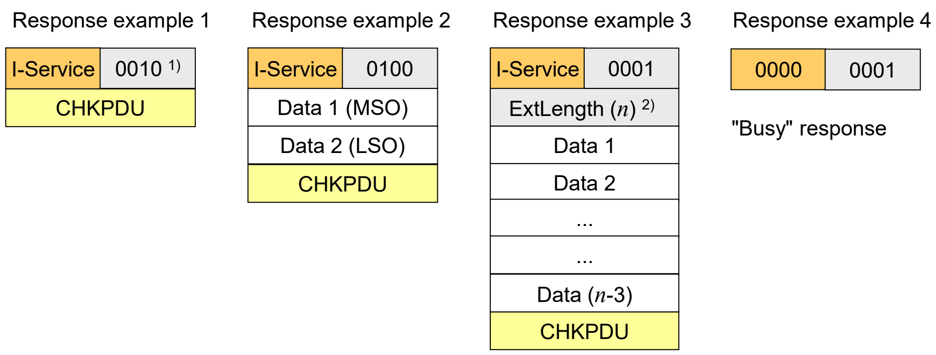
The ISDU request in example 2 comprises one Index element allowing addressing from 0 to 255 and the Subindex element allowing addressing an element of a data structure. The total length is 6 ("0110").

The ISDU request in example 3 comprises two Index elements allowing to address from 256 to 65535 (see Table A.15) and the Subindex element allowing to address an element of a data structure. The total length is 7 ("0111").

The ISDU request in example 4 comprises one Index element and the ExtLength element indicating the number of ISDU elements (n), permitting numbers from 17 to 238. In this case the Length element has the value "1".

The ISDU request "Idle" in example 5 is used to indicate that no service is pending.

Figure A.19 demonstrates typical examples of response ISDUs, which are explained in the following paragraphs.



1) Minimum length = 2 ("0010")
2) Overall ISDU ExtLength = n (17 to 238);
Length = 1 ("0001")

Figure A.19 – Examples of response ISDUs

The ISDU response in example 1 shows the minimum value 2 for the Length element ("0010").

The ISDU response in example 2 shows two Data elements and a total number of 4 elements in the Length element ("0100"). Data 1 carries the most significant octet (MSO) and Data 2 the least significant octet (LSO).

The ISDU response in example 3 shows the ExtLength element indicating the number of ISDU elements (n), permitting numbers from 17 to 238. In this case the Length element has the value "1".

The ISDU response "Busy" in example 4 is used when a Device is currently not able to respond to the read request of the Master due to the necessary preparation time for the response.

Figure A.20 shows a typical example of both a read and a write request ISDU, which are explained in the following paragraphs.

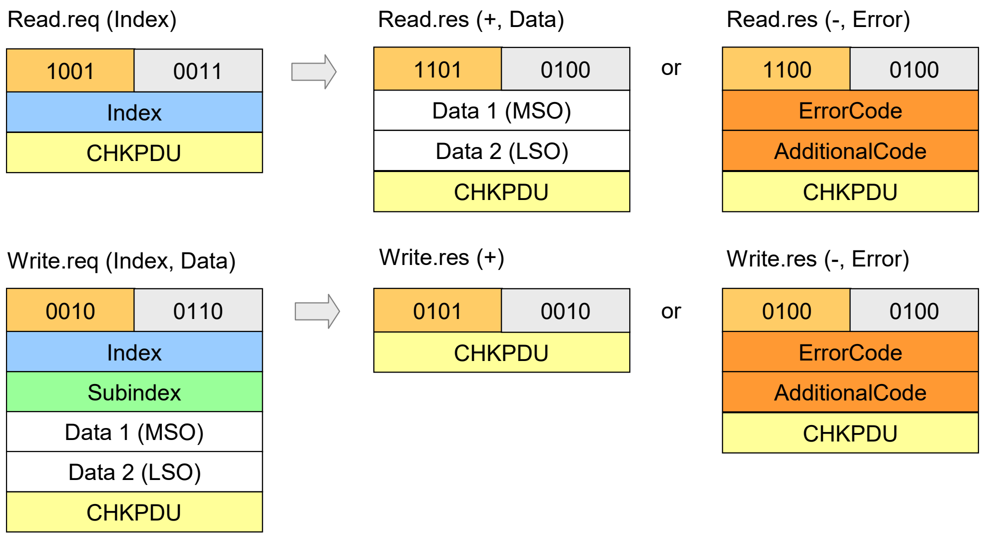


Figure A.20 – Examples of read and write request ISDUs

The code of the read request I-Service is "1001". According to Table A.13 this comprises an Index element. A successful read response (+) of the Device with code "1101" is shown next to the request with two Data elements. Total length is 4 ("0100"). An unsuccessful read response (-) of the Device with code "1100" is shown next in line. It carries the ErrorType with the two Data elements ErrorCode and AdditionalCode (see Annex C).

The code of the write request I-Service is "0010". According to Table A.13 this comprises an Index and a Subindex element. A successful write response (+) of the Device with code "0101" is shown next to the request with no Data elements. Total length is 2 ("0010"). An unsuccessful read response (-) of the Device with code "0100" is shown next in line. It carries the ErrorType with the two Data elements ErrorCode and AdditionalCode (see Annex C).

A.6 General structure and encoding of Events

A.6.1 General

In 7.3.8.1 and Table 58 the purpose and general structure of the Event memory is specified. This memory accommodates a StatusCode, several EventQualifiers and their associated EventCodes. The coding of these memory elements is specified in the subsequent sections.

A.6.2 StatusCode type 1 (no details)

Figure A.21 shows the structure of this StatusCode.

NOTE 1 StatusCode type 1 is only used in Events generated by legacy devices (see 7.3.8.1).

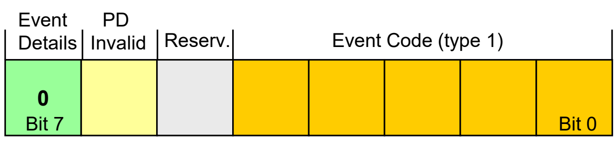


Figure A.21 – Structure of StatusCode type 1

Bits 0 to 4: EventCode (type 1)

The coding of this data structure is listed in Table A.16. The EventCodes are mapped into EventCodes (type 2) as listed in Annex D. See 7.3.8.2 for additional information.

Table A.16 – Mapping of EventCodes (type 1)

EventCode (type 1)	EventCode (type2)	Instance	Type	Mode
****1	0xFF80	Application	Notification	Event single shot
***1*	0xFF80	Application	Notification	Event single shot
1	0x6320	Application	Notification	Event single shot
*1***	0xFF80	Application	Notification	Event single shot
1****	0xFF10	Application	Notification	Event single shot
Key * Don't care				

Bit 5: Reserved

This bit is reserved and shall be set to zero in StatusCode type 1.

Bit 6: PD Invalid

NOTE 2 This bit is used in legacy protocol (see [8]) for PDinvalid indication.

Bit 7: Event Details

This bit indicates that no detailed Event information is available. It shall always be set to zero in StatusCode type 1.

A.6.3 StatusCode type 2 (with details)

Figure A.22 shows the structure of the StatusCode type 2.

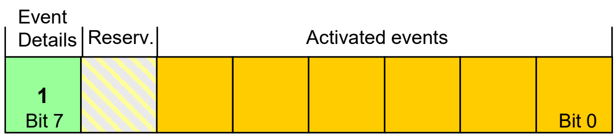


Figure A.22 – Structure of StatusCode type 2

Bits 0 to 5: Activated Events

Each bit is linked to an Event in the memory (see 7.3.8.1) as demonstrated in Figure A.23. Bit 0 is linked to Event 1, bit 1 to Event 2, etc. A bit with value "1" indicates that the corresponding EventQualifier and the EventCode have been entered in valid formats in the memory. A bit with value "0" indicates an invalid entry.

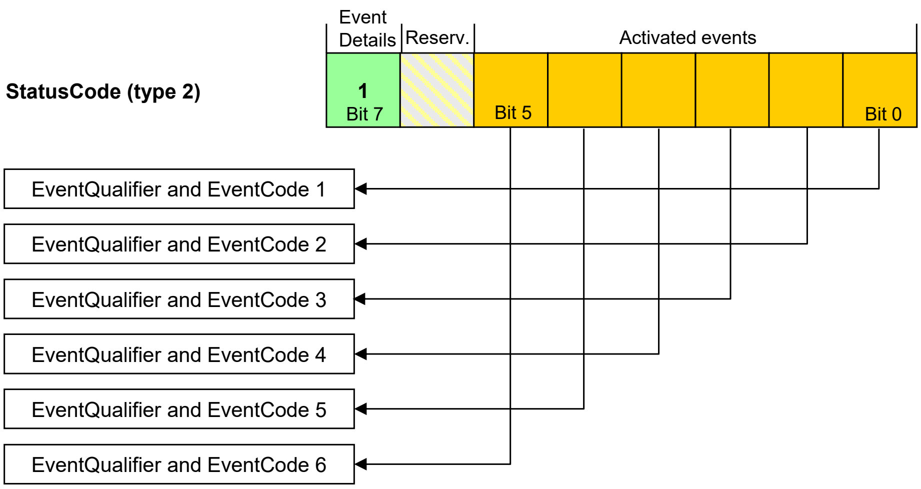


Figure A.23 – Indication of activated Events

Bit 6: Reserved

This bit is reserved and shall be set to zero.

NOTE This bit is used in the legacy protocol version according to [8] for PDinvalid indication

Bit 7: Event Details

This bit indicates that detailed Event information is available. It shall always be set in StatusCode type 2.

A.6.4 EventQualifier

The structure of the EventQualifier is shown in Figure A.24.

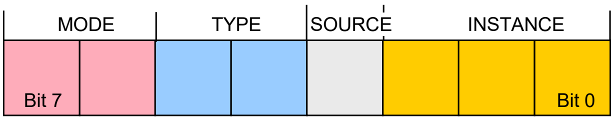


Figure A.24 – Structure of the EventQualifier

Bits 0 to 2: INSTANCE

These bits indicate the particular source (instance) of an Event thus refining its evaluation on the receiver side. Permissible values for INSTANCE are listed in Table A.17.

Table A.17 – Values of INSTANCE

Value	Definition
0	Unknown
1 to 3	Reserved
4	Application
5	System
6 to 7	Reserved

Bit 3: SOURCE

This bit indicates the source of the Event. Permissible values for SOURCE are listed in Table A.18.

Table A.18 – Values of SOURCE

Value	Definition
0	Device (remote)
1	Master/Port

Bits 4 to 5: TYPE

These bits indicate the Event category. Permissible values for TYPE are listed in Table A.19.

Table A.19 – Values of TYPE

Value	Definition
0	Reserved
1	Notification
2	Warning
3	Error

Bits 6 to 7: MODE

These bits indicate the Event mode. Permissible values for MODE are listed in Table A.20.

Table A.20 – Values of MODE

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

A.6.5 EventCode

The EventCode entry contains the identifier of an actual Event. Permissible values for EventCode are listed in Annex D.

Annex B
(normative)

Parameter and commands

B.1 Direct Parameter page 1 and 2

B.1.1 Overview

In principle, the designer of a Device has a large amount of space for parameters and commands as shown in Figure 6. SDCI offers the so-called Direct Parameter pages 1 and 2 with a simplified access method (page communication channel according to Table A.1).

The range of Direct Parameters is structured as shown in Figure B.1. It is split into page 1 and page 2.

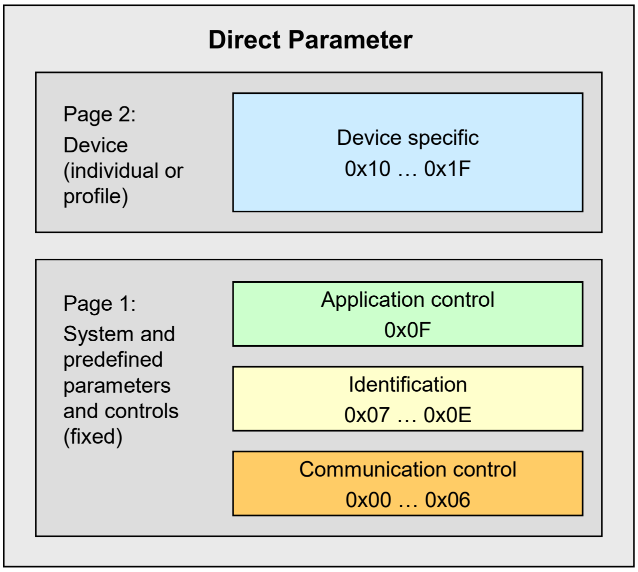


Figure B.1 – Classification and mapping of Direct Parameters

Page 1 ranges from 0x00 to 0x0F. It comprises the following categories of parameters:

- Communication parameter
- Identification parameter
- Application parameter

The Master application layer (AL) provides read only access to Direct Parameter page 1 as data objects (see 8.2.1) via Index 0. Single octets can be read via Index 0 and the corresponding Subindex. Subindex 1 indicates address 0x00 and Subindex 16 address 0x0F.

Page 2 ranges from 0x10 to 0x1F. This page comprises parameters optionally used by the individual Device technology. The Master application layer (AL) provides read/write access to Direct Parameter page 2 in form of data objects (see 8.2.1) via Index 1. Single octets can be written or read via Index 1 and the corresponding Subindex. Subindex 1 indicates address 0x10 and Subindex 16 address 0x1F.

A Device shall always return the value "0" upon a read access to Direct Parameter addresses, which are not implemented (for example in case of reserved parameter addresses or not supported optional parameters). The Device shall ignore a write access to not implemented parameters.

The structure of the Direct Parameter pages 1 and 2 is specified in Table B.1.

Table B.1 – Direct Parameter page 1 and 2

Address	Parameter name	Access	Implementation /reference	Description
Direct Parameter page 1				
0x00	Master-Command	W	Mandatory/ see B.1.2	Master command to switch to operating states (see NOTE 1)
0x01	MasterCycle-Time	R/W	Mandatory/ see B.1.3	Actual cycle duration used by the Master to address the Device. Can be used as a parameter to monitor Process Data transfer.
0x02	MinCycleTime	R	Mandatory/ see B.1.3	Minimum cycle duration supported by a Device. This is a performance feature of the Device and depends on its technology and implementation.
0x03	M-sequence Capability	R	Mandatory/ see B.1.4	Information about implemented options related to M-sequences and physical configuration
0x04	RevisionID	R/W	Mandatory/ see B.1.5	ID of the used protocol version for implementation (shall be set to 0x11)
0x05	ProcessDataIn	R	Mandatory/ see B.1.6	Type and length of input data (Process Data from Device to Master)
0x06	ProcessData-Out	R	Mandatory/ see B.1.7	Type and length of output data (Process Data from Master to Device)
0x07	VendorID 1 (MSB)	R	Mandatory/ see B.1.8	Unique vendor identification (see NOTE 2)
0x08	VendorID 2 (LSB)			
0x09	DeviceID 1 (Octet 2, MSB)	R/W	Mandatory/ see B.1.9	Unique Device identification allocated by a vendor
0x0A	DeviceID 2 (Octet 1)			
0x0B	DeviceID 3 (Octet 0, LSB)			
0x0C	FunctionID 1 (MSB)	R	see B.1.10	Reserved (see Table 102)
0x0D	FunctionID 2 (LSB)			
0x0E		R	reserved	
0x0F	System-Command	W	Optional/ see B.1.11	Command interface for end user applications only and Devices without ISDU support (see NOTE 1)
Direct Parameter page 2				
0x10… 0x1F	Vendor specific	Optional	Optional/ see B.1.12	Device specific parameters
NOTE 1 A read operation returns unspecified values NOTE 2 VendorIDs are assigned by the IO-Link community				

B.1.2 MasterCommand

The Master application is able to check the status of a Device or to control its behaviour with the help of MasterCommands (see 7.3.7).

Permissible values for these parameters are specified in Table B.2.

Table B.2 – Types of MasterCommands

Value	MasterCommand	Description
0x00 to 0x59	Reserved	

Value	MasterCommand	Description
0x5A	Fallback	Transition from communication to SIO mode. The Device shall execute this transition after 3 Master-CycleTimes and before 500 ms elapsed after the MasterCommand.
0x5B to 0x94	Reserved	
0x95	MasterIdent	Indicates a Master revision higher than 1.0
0x96	DeviceIdent	Start check of Direct Parameter page for changed entries
0x97	DeviceStartup	Switches the Device from OPERATE or PREOPERATE to STARTUP
0x98	ProcessDataOutputOperate	Process output data valid
0x99	DeviceOperate	Process output data invalid or not available. Switches the Device from STARTUP or PREOPERATE to OPERATE
0x9A	DevicePreoperate	Switches the Device from STARTUP to state PREOPERATE
0x9B to 0xFF	Reserved	

B.1.3 MasterCycleTime and MinCycleTime

The MasterCycleTime is a Master parameter and sets up the actual cycle time of a particular port.

The MinCycleTime is a Device parameter to inform the Master about the shortest cycle time supported by this Device.

See A.3.7 for the application of the MasterCycleTime and the MinCycleTime. The structure of these two parameters is shown in Figure B.2.

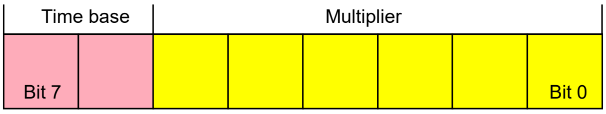


Figure B.2 – MinCycleTime

Bits 0 to 5: Multiplier

These bits contain a 6-bit multiplier for the calculation of MasterCycleTime and MinCycleTime. Permissible values for the multiplier are 0 to 63, further restrictions see Table B.3.

Bits 6 to 7: Time Base

These bits specify the time base for the calculation of MasterCycleTime and MinCycleTime.

In the following cases, when

- the Device provides no MinCycleTime, which is indicated by a MinCycleTime equal zero (binary code 0x00),
- or the MinCycleTime is shorter than the calculated M-sequence time with the M-sequence type used by the Device, with (t_1, t_2, t_{idle}) equal zero and t_A equal one bit time (see A.3.4 to A.3.6)

the Master shall use the calculated worst case M-sequence timing, with the M-sequence type used by the Device, and the maximum times for t_A and t_2 (see A.3.4 to A.3.6):

The permissible combinations for time base and multiplier are listed in Table B.3 along with the resulting values for MasterCycleTime or MinCycleTime.

Table B.3 – Possible values of MasterCycleTime and MinCycleTime

Time base encoding	Time Base value	Calculation	Cycle Time
00	0,1 ms	Multiplier × Time Base	0,4 ms to 6,3 ms
01	0,4 ms	6,4 ms + Multiplier × Time Base	6,4 ms to 31,6 ms
10	1,6 ms	32,0 ms + Multiplier × Time Base	32,0 ms to 132,8 ms
11	Reserved	Reserved	Reserved

B.1.4 M-sequenceCapability

The structure of the M-sequenceCapability parameter is shown in Figure B.3.

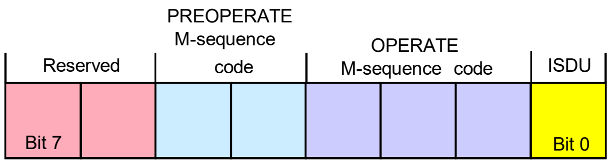


Figure B.3 – M-sequenceCapability

Bit 0: ISDU

This bit indicates whether or not the ISDU communication channel is supported. Permissible values for ISDU are listed in Table B.4.

Table B.4 – Values of ISDU

Value	Definition
0	ISDU not supported
1	ISDU supported

NOTE By future mandatory support of the Common Profile [7], the support of ISDUs will become mandatory in future releases.

Bits 1 to 3: Coding of the OPERATE M-sequence type

This parameter indicates the available M-sequence type during the OPERATE state. Permissible codes for the OPERATE M-sequence type are listed in Table A.9 for legacy Devices and in Table A.10 for Devices according to this standard.

Bits 4 to 5: Coding of the PREOPERATE M-sequence type

This parameter indicates the available M-sequence type during the PREOPERATE state. Permissible codes for the PREOPERATE M-sequence type are listed in Table A.8.

Bits 6 to 7: Reserved

These bits are reserved and shall be set to zero in this version of the specification.

B.1.5 RevisionID (RID)

The RevisionID parameter is the two-digit version number of the SDCI protocol currently used within the Device. Its structure is shown in Figure B.4. The initial value of RevisionID at powerup is the inherent value for protocol RevisionID. It can be overwritten (see 10.6.3 and Table 101) until the next powerup.

This revision of the standard specifies protocol version 1.1.

NOTE The legacy protocol version 1.0 is specified in [8].

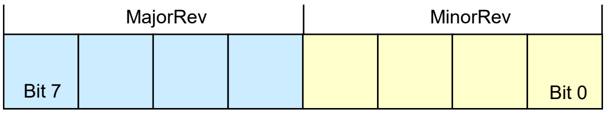


Figure B.4 – RevisionID

Bits 0 to 3: MinorRev

These bits contain the minor digit of the version number, for example 0 for the protocol version 1.0. Permissible values for MinorRev are 0x0 to 0xF.

Bits 4 to 7: MajorRev

These bits contain the major digit of the version number, for example 1 for the protocol version 1.0. Permissible values for MajorRev are 0x0 to 0xF.

B.1.6 ProcessDataIn

The structure of the ProcessDataIn parameter is shown in Figure B.5.

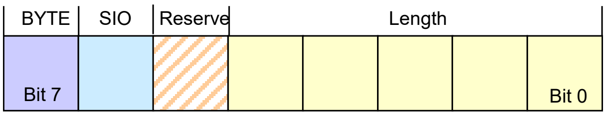


Figure B.5 – ProcessDataIn

Bits 0 to 4: Length

These bits contain the length of the input data (Process Data from Device to Master) in the length unit designated in the BYTE parameter bit. Permissible codes for Length are specified in Table B.6.

Bit 5: Reserve

This bit is reserved and shall be set to zero in this version of the specification.

Bit 6: SIO

This bit indicates whether the Device provides a switching signal in SIO mode. Permissible values for SIO are listed in Table B.5.

Table B.5 – Values of SIO

Value	Definition
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 definition of the Process Data length in conjunction with Length are listed in Table B.6.

Table B.6 – Permitted combinations of BYTE and Length

BYTE	Length	Definition
0	0	no Process Data
0	1	1 bit Process Data, structured in bits
0	n (2-15)	*n* bit Process Data, structured in bits
0	16	16 bit Process Data, structured in bits
0	17 to 31	Reserved
1	0, 1	Reserved

BYTE	Length	Definition
1	2	3 octets Process Data, structured in octets
1	n (3-30)	n+1 octets Process Data, structured in octets
1	31	32 octets Process Data, structured in octets

B.1.7 ProcessDataOut

The structure of the ProcessDataOut parameter is the same as with ProcessDataIn, except with bit 6 ("SIO") reserved.

B.1.8 VendorID (VID)

These octets contain a worldwide unique value per vendor.

NOTE VendorIDs are assigned by the IO-Link community.

B.1.9 DeviceID (DID)

These octets contain the currently used DeviceID. A value of "0" is not permitted. It is highly recommended to store the value of DeviceID in non-volatile memory after a compatibility switch until a reset to the initial value through SystemCommands "Restore factory settings" or " Back-to-box". The value can be overwritten during StartUp (see 10.6.2).

NOTE The communication parameters MinCycleTime, M-sequence Capability, Process Data In and Process Data Out can be changed to achieve compatibility to the requested DeviceID.

B.1.10 FunctionID (FID)

This parameter will be defined in a later version.

B.1.11 SystemCommand

Only Devices without ISDU support shall use the parameter SystemCommand in the Direct Parameter page 1. The implementation of SystemCommand is optional. See Table B.9 for a detailed description of the SystemCommand functions.

NOTE The SystemCommand on the Direct Parameter page 1 does not provide a positive or negative response upon execution of a selected function

B.1.12 Device specific Direct Parameter page 2

The Device specific Direct Parameters are a set of parameters available to the Device specific technology. The implementation of Device specific Direct Parameters is optional. It is highly recommended for Devices (with ISDU) not to use parameters on Direct Parameter page 2.

NOTE The complete parameter list of the Direct Parameter page 2 is read or write accessible via index 1 (see B.1.1).

B.2 Predefined Device parameters

B.2.1 Overview

The many different technologies and designs of sensors and actuators require individual and easy access to complex parameters and commands beyond the capabilities of the Direct Parameter page 2. From a Master's point of view, these complex parameters and commands are called application data objects.

Figure B.6 shows the general mapping of data objects for the ISDU transmission.

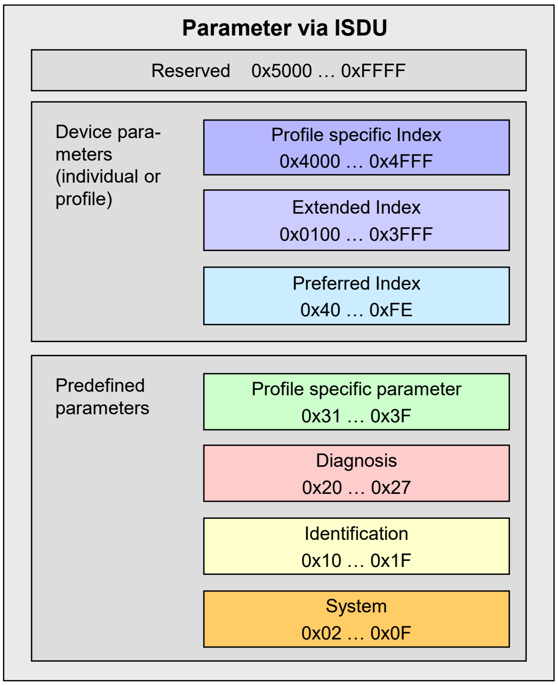


Figure B.6 – Index space for ISDU data objects

So-called ISDU "containers" are the transfer means to exchange application data objects or short data objects. The index of the ISDU is used to address the data objects.

Subclause B.2 contains definitions and requirements for the implementation of technology specific Device applications. Implementation rules for parameters and commands are specified in Table B.7.

Table B.7 – Implementation rules for parameters and commands

Rule number	Rule specification
1	All parameters of an Index shall be readable and/or writeable as an entire data object via Subindex 0
2	The technology specific Device application shall resolve inconsistencies of dependent parameter sets during parameterization
3	The duration of an ISDU service request is limited (see Table 102). A master application can abort ISDU services after this timeout
4	Application commands (for example teach-in, reset to factory settings, etc.) are treated like parameters.

Table B.8 specifies the assignment of data objects (parameters and commands) to the Index range of ISDUs. All indices above 2 are ISDU related.

Table B.8 – Index assignment of data objects (Device parameter)

Index (dec)	Object name	Access	Length	Data type	M/O/C	Remark
0x0000 (0)	Direct Parameter Page 1	R		RecordT	M	Redirected to the page communication channel, see 10.8.5
0x0001 (1)	Direct Parameter Page 2	R/W		RecordT	M	Redirected to the page communication channel, see 10.8.5
0x0002 (2)	System-Command	W	1 octet	UIntegerT	C	Command Code Definition (See B.2.2)

Index (dec)	Object name	Access	Length	Data type	M/O/ C	Remark
0x0003 (3)	Data-Storage-Index	R/W	variable	RecordT	M	Set of data objects for storage (See B.2.3)
0x0004-0x000B (4-11)	Reserved					Reserved for exceptional operations
0x000C (12)	Device-Access-Locks-	R/W	2 octets	RecordT	O	Standardized Device locking functions (See B.2.4)
0x000D (13)	Profile-Charac-teristic	R	variable	ArrayT of UIntegerT16	C	Reserved for Common Profile [7] (see B.2.5)
0x000E (14)	PDInput-Descriptor	R	variable	ArrayT of OctetStringT3	C	Reserved for Common Profile [7] (see B.2.6)
0x000F (15)	PDOutput-Descriptor	R	variable	ArrayT of OctetStringT3	C	Reserved for Common Profile [7] (see B.2.7)
0x0010 (16)	Vendor-Name	R	max. 64 octets	StringT NOTE	M	Vendor information (See B.2.8)
0x0011 (17)	Vendor-Text	R	max. 64 octets	StringT NOTE	O	Additional vendor information (See B.2.9)
0x0012 (18)	Product-Name	R	max. 64 octets	StringT NOTE	M	Detailed product or type name (See B.2.10)
0x0013 (19)	ProductID	R	max. 64 octets	StringT NOTE	O	Product or type identification (See B.2.11)
0x0014 (20)	Product-Text	R	max. 64 octets	StringT NOTE	O	Description of Device function or characteristic (See B.2.12)
0x0015 (21)	Serial-Number	R	max. 16 octets	StringT NOTE	O	Vendor specific serial number (See B.2.13)
0x0016 (22)	Hardware-Revision	R	max. 64 octets	StringT NOTE	O	Vendor specific format (See B.2.14)
0x0017 (23)	Firmware-Revision	R	max. 64 octets	StringT NOTE	O	Vendor specific format (See B.2.15)
0x0018 (24)	Application-Specific-Tag	R/W	min. 16, max. 32 octets	StringT NOTE	O	Tag defined by user (See B.2.16)
0x0019 (25)	Function-Tag	R/W	max. 32 octets	StringT NOTE	C	Reserved for Common Profile [7] (See B.2.17)
0x001A (26)	Location-Tag	R/W	max. 32 octets	StringT NOTE	C	Reserved for Common Profile [7] (See B.2.18)
0x001B (27)	Product-URI	R	max. 100 octets	StringT NOTE	C	Reserved for Common Profile [7] (See B.2.19)
0x001C-0x001F (28-31)	Reserved					
0x0020 (32)	ErrorCount	R	2 octets	UIntegerT	O	Errors since power-on or reset (See B.2.20)
0x0021-0x0023 (33-35)	Reserved					
0x0024 (36)	Device-Status	R	1 octet	UIntegerT	O	Contains current status of the Device (See B.2.21)
0x0025 (37)	Detailed-Device-Status	R	variable	ArrayT of OctetStringT3	O	See B.2.22

Index (dec)	Object name	Access	Length	Data type	M/O/C	Remark
0x0026-0x0027 (38-39)	Reserved					
0x0028 (40)	Process-DataInput	R	PD length	Device specific	O	Read last valid Process Data from PDin channel (See B.2.23)
0x0029 (41)	Process-DataOutput	R	PD length	Device specific	O	Read last valid Process Data from PDout channel (See B.2.24)
0x002-0x002F (42-47)	Reserved					
0x0030 (48)	Offset- Time	R/W	1 octet	RecordT	O	Synchronization of Device application timing to M-sequence timing (See B.2.25)
0x0031-0x003F (49-63)	Reserved for profiles					
0x0040-0x00FE (64-254)	Preferred Index					Device specific (8 bit)
0x00FF (255)	Reserved					
0x0100-0x3FFF (256-16383)	Extended Index					Device specific (16 bit)
0x4000-0x41FF (16384-16895)	Profile specific Index					Reserved for Device profile
0x4200-0x42FF (16896-17151)	Safety specific Index					Reserved for Safety system extensions [10]
0x4300-0x4FFF (17152-20479)	Profile specific Index					Reserved for Device profile
0x5000-0x50FF (20480-20735)	Wireless specific Index					Reserved for Wireless system extensions [11]
0x5100-0xFFFF (20736-65535)	Reserved					

Key M = mandatory; O = optional; C = conditional, see full description of parameter for condition

NOTE UTF8 coding required for StringT

B.2.2 SystemCommand

Devices with ISDU support shall use the ISDU Index 0x0002 to receive the SystemCommand. The commands shall be acknowledged. The possible responses are defined in 10.3.7. The timing of the appropriate response is defined together with the SystemCommand functionality.

The coding of SystemCommands is specified in Table B.9.

Table B.9 – Coding of SystemCommand

Command (hex)	Command (dec)	Command name	H/O/C	Definition
0x00	0	Reserved		
0x01	1	ParamUploadStart	C	Start parameter upload
0x02	2	ParamUploadEnd	C	Stop parameter upload
0x03	3	ParamDownloadStart	C	Start parameter download
0x04	4	ParamDownloadEnd	C	Stop parameter download
0x05	5	ParamDownloadStore	C	Finalize parameterization and start Data Storage
0x06	6	ParamBreak	C	Cancel all Param commands
0x07 to 0x3F	7 to 63	Reserved		
0x40 to 0x7F	64 to 127	Reserved for profiles		
0x80	128	Device reset	O	See 10.7.2
0x81	129	Application reset	H	See 10.7.3
0x82	130	Restore factory settings	O	See 10.7.4
0x83	131	Back-to-box	C	See 10.7.5
0x84 to 0x9F	132 to 159	Reserved		
0xA0 to 0xFF	160 to 255	Vendor specific		
NOTE See 10.3 **Key** H = highly recommended;O = optional; C = conditional, see full description of command for condition				

The SystemCommand 0x05 (ParamDownloadStore) shall be implemented according to 10.4.2, whenever the Device provides parameters to be stored via the Data Storage mechanism, i.e. parameter "Index_List" in Index 0x0003 is not empty (see Table B.10).

The implementation of the SystemCommands 0x01 to 0x06 required for Block Parameterization according to 10.3.5 is optional. However, all of these commands or none of them shall be implemented (for SystemCommand 0x05 the rule for Data Storage dominates).

See B.1.11 for SystemCommand options on the Direct Parameter page 1.

Implementation of the SystemCommand feature is conditional for Devices and depends on the availability of any conveyed functionality like Block Parametrization, profiled or manufacturer specific functionalities."

B.2.3 DataStorageIndex

Table B.10 specifies the DataStorageIndex assignments. Record items shall not be separated by offset gaps. Offsets shall be built according Table F.19.

Table B.10 – DataStorageIndex assignments

Index	Sub-index	Offset	Access	Parameter Name	Coding	Data type
0x0003	01	N+72	R/W	DS_Command	0x00: Reserved 0x01: DS_UploadStart 0x02: DS_UploadEnd 0x03: DS_DownloadStart 0x04: DS_DownloadEnd 0x05: DS_Break 0x06 to 0xFF: Reserved	UIntegerT8 (8 bit)

Index	Sub-index	Offset	Access	Parameter Name	Coding	Data type
	02	N+64	R	State_Property	Bit 0: Reserved Bit 1 and 2: State of Data Storage 0b00: Inactive 0b01: Upload 0b10: Download 0b11: Data Storage locked Bit 3 to 6: Reserved Bit 7: DS_UPLOAD_FLAG "1": DS_UPLOAD_REQ pending "0": no DS_UPLOAD_REQ	UIntegerT8 (8 bit)
	03	N+32	R	Data_Storage_ Size	Number of octets for storing all the necessary information for the Device replacement (see 10.4.5). Maximum size is 2 048 octets.	UIntegerT32 (32 bit)
	04	N	R	Parameter_ Checksum	Parameter set revision indication: CRC signature or Revision Counter (see 10.4.8)	UIntegerT32 (32 bit)
	05	0	R	Index_List	List of parameter indices to be saved (see Table B.11)	OctetStringT (variable)
NOTE N = (n × 3 + 2) × 8; for n see Table B.11						

The parameter DataStorageIndex 0x0003 contains all the information to be used for the Data Storage handling. This parameter is reserved for private exchanges between the Master and the Device; the Master shall block any write access request from a gateway application to this Index (see Figure 5). The parameters within this Index 0x0003 are specified as follows.

DS_Command

This octet carries the Data Storage commands for the Device.

A read operation returns unspecified values.

Note: The reaction of the DS_Command is similar to the SystemCommand, but it is assumed, that the Master implementation will not cause any erroneous access.

State_Property

This octet indicates the current status of the Data Storage mechanism. Bit 7 shall be stored in non-volatile memory. The Master checks this bit at start-up and performs a parameter upload if requested.

Data_Storage_Size

These four octets provide the requested memory size as number of octets for storing all the information required for the replacement of a Device including the structural information (Index, Subindex). Data type is UIntegerT32 (32 bit). The maximum size is 2 048 octets. See Table G.1 for the elements to be taken into account in the size calculation.

Parameter_Checksum

This checksum is used to detect changes in the parameter set without reading all parameters. The value of the checksum is calculated according to the procedure in 10.4.8. The Device shall change the checksum whenever a parameter out of the parameter set has been altered. Different parameter sets shall hold different checksums. It is recommended that the Device stores this parameter locally in non-volatile memory.

Index_List

Table B.11 specifies the structure of the Index_List. Each Index_List can carry up to 70 entries (see Table 102).

Table B.11 – Structure of Index_List

Entry	Address	Definition	Data type
X1	Index	Index of first parameter to be saved	Unsigned16
	Subindex	Subindex of first parameter to be saved	Unsigned8
X2	Index	Index of next parameter to be saved	Unsigned16
	Subindex	Subindex of next parameter to be saved	Unsigned8
…..	………	………………………………………	………..
Xn	Index	Index of last parameter to be saved	Unsigned16
	Subindex	Subindex of last parameter to be saved	Unsigned8
Xn+1	Index	Termination_Marker 0x0000: End of Index_List >0x0000: Next Index containing an Index_List	Unsigned16

Large sets of parameters can be handled via concatenated Index_Lists. The last two octets of the Index_List shall carry the Termination Marker. A value "0" indicates the end of the Index List. In case of concatenation the Termination Marker is set to the next Index containing an Index List. The structure of the following Index List is the same as specified in Table B.11. Thus, the concatenation of lists ends if a Termination Marker with the value "0" is found.

B.2.4 DeviceAccessLocks

The parameter DeviceAccessLocks allows control of the Device behaviour. Standardized Device functions can independently be configured via defined flags in this parameter. The DeviceAccessLocks configuration can be changed by overwriting the parameter. The actual configuration setting is available per read access to this parameter. The data type is RecordT of BooleanT. Access is only permitted via Subindex 0.

This parameter is optional. If implemented it shall be non-volatile.

The following Device access lock categories are specified.

- Parameter write access (obsolete)
- Data Storage (obsolete)
- Local parameterization (optional)
- Local user interface operation (optional)

Table B.12 lists the Device locking possibilities.

Table B.12 – Device locking possibilities

Bit	Category	Definition
0	Parameter (write) access	0: unlocked (default) 1: locked (highly recommended not to implement/use)
1	Data Storage	0: unlocked (default) NOTE 1: locked (highly recommended not to implement/use)
2	Local parameterization (optional)	0: unlocked (default) 1: locked
3	Local user interface (optional)	0: unlocked (default) 1: locked
4 – 15	Reserved	

NOTE For compatibility reasons, the Master still reads the parameter State_Property /State of Data Storage (see Table B.10).

Parameter (write) access:

If this bit is set, write access to all Device parameters over the SDCI communication interface is inhibited for all read/write parameters of the Device except the parameter Device Access Locks. Read access is not affected. The Device shall respond with the negative service response – access denied – to a write access, if the parameter access is locked.

The parameter (write) access lock mechanism shall not block downloads of the Data Storage mechanism (between DS_DownloadStart and DS_DownloadEnd or DS_Break).

Data Storage:

If this bit is set in the Device, the Data Storage mechanism is disabled (see 10.4.2 and 11.4.4). In this case, the Device shall respond to a write access (within its Data Storage Index) with a negative service response – access denied – (see B.2.3). Read access to its DataStorageIndex is not affected.

This setting is also indicated in the State Property within Data Storage Index.

Local parameterization:

If this bit is set, the parameterization via local control elements on the Device is inhibited (write protection). Read only is possible (see 10.6.7).

Local user interface:

If this bit is set, operation of the human machine interface on the Device is disabled (see 10.6.8).

B.2.5 ProfileCharacteristic

This parameter contains the list of ProfileIdentifiers (PID's) corresponding to the Device Profile implemented in the Device. This parameter is conditional on the associated Profile.

NOTE Details are provided in [7].

B.2.6 PDInputDescriptor

This parameter contains the description of the data structure of the process input data for a profile Device. This parameter is conditional on the associated Profile.

NOTE Details are provided in [7].

B.2.7 PDOutputDescriptor

This parameter contains the description of the data structure of the process output data for a profile Device. This parameter is conditional on the associated Profile.

NOTE Details are provided in [7].

B.2.8 VendorName

The parameter VendorName contains only one of the vendor names listed for the assigned VendorID. The parameter is a read-only data object. The data type is StringT with a maximum fixedLength of 64. This parameter is mandatory.

NOTE The list of vendor names associated with a given VendorID is maintained by the IO-Link community.

B.2.9 VendorText

The parameter VendorText contains additional information about the vendor. The parameter is a read-only data object. The data type is StringT with a maximum fixedLength of 64. This parameter is optional.

B.2.10 ProductName

The parameter ProductName contains the complete product name. The parameter is a read-only data object. The data type is StringT with a maximum fixedLength of 64. This parameter is mandatory.

NOTE The corresponding entry in the IODD Device variant list is expected to match this parameter.

B.2.11 ProductID

The parameter ProductID shall contain the vendor specific product or type identification of the Device. The parameter is a read-only data object. The data type is StringT with a maximum fixedLength of 64. This parameter is optional.

B.2.12 ProductText

The parameter ProductText shall contain additional product information for the Device, such as product category (for example Photoelectric Background Suppression, Ultrasonic Distance Sensor, Pressure Sensor, etc.). The parameter is a read-only data object. The data type is StringT with a maximum fixedLength of 64. This parameter is optional.

B.2.13 SerialNumber

The parameter SerialNumber shall contain a unique vendor specific notation for each individual Device. The parameter is a read-only data object. The data type is StringT with a maximum fixedLength of 16. This parameter is optional.

B.2.14 HardwareRevision

The parameter HardwareRevision shall contain a vendor specific notation for the hardware revision of the Device. The parameter is a read-only data object. The data type is StringT with a maximum fixedLength of 64. This parameter is optional.

B.2.15 FirmwareRevision

The parameter FirmwareRevision shall contain a vendor specific notation for the firmware revision of the Device. The parameter is a read-only data object. The data type is StringT with a maximum fixedLength of 64. This parameter is optional.

B.2.16 ApplicationSpecificTag

The parameter ApplicationSpecificTag shall be provided as read/write data object for the user application. It can serve as a free user specific tag. The data type is StringT with a minimum fixedLength of 16, and a preferred fixedLength of 32 octets (see [7]). As default it is recommended to fill this parameter with "***". This parameter is optional.

B.2.17 FunctionTag

The parameter FunctionTag contains the description of the specific function of a profile Device within an application. As default it is recommended to fill this parameter with "***". This parameter is conditional on the associated Profile.

NOTE Details are provided in [7]

B.2.18 LocationTag

The parameter LocationTag contains the description of the location of a profile Device within an application. As default it is recommended to fill this parameter with "***". This parameter is conditional on the associated Profile.

NOTE Details are provided in [7]

B.2.19 ProductURI

The parameter ProductURI contains the globally biunique identification of a profile Device. This parameter is conditional on the associated Profile.

NOTE Details are provided in [7]

B.2.20 ErrorCount

The parameter ErrorCount provides information on errors occurred in the Device application since power-on or reset. Usage of this parameter is vendor or Device specific. The data type is UIntegerT with a bitLength of 16. The parameter is a read-only data object. This parameter is optional.

B.2.21 DeviceStatus

B.2.21.1 Overview

The parameter DeviceStatus shall provide information about the Device condition (diagnosis) by the Device's technology. The data type is UIntegerT with a bitLength of 8. The parameter is a read-only data object. This parameter is optional.

The following Device conditions in Table B.13 are specified. They shall be generated by the Device applications, the relation to the DetailedDeviceStatus is defined in 10.10.1. The parameter DeviceStatus can be read by any PLC program or tools such as Asset Management (see Clause 11).

Table B.13 lists the different DeviceStatus information. The criteria for these indications are specified in subclauses B.2.21.3 through B.2.21.6. The priority column defines which status value is signalled in case of multiple active events, the lowest priority value dominates higher priority values.

Table B.13 – DeviceStatus parameter

Value	Priority	Definition
0	5	Device is operating properly (see B.2.21.2)
1	3	Maintenance-Required (see B.2.21.3)
2	4	Out-of-Specification (see B.2.21.4)
3	2	Functional-Check (see B.2.21.5)
4	1	Failure (see B.2.21.6)
5 – 255	-	Reserved

B.2.21.2 Device is operating properly

The Device is working without any impairment and no Event is pending, see B.2.22.

B.2.21.3 Maintenance-required

Although the Process Data are valid, internal diagnostics indicate that the Device is close to lose its ability of correct functioning.

EXAMPLES Optical lenses getting dusty, build-up of deposits, lubricant level low.

B.2.21.4 Out-of-Specification

Although the Process Data are valid, internal diagnostics indicate that the Device is operating outside its specified measuring range or environmental conditions.

EXAMPLES Power supply, auxiliary energy, temperature, pneumatic pressure, magnetic interference, vibrations, acceleration, interfering light, bubble formation in liquids.

B.2.21.5 Functional-Check

User intended manipulations on the Device are ongoing and the Device may not be able to provide valid Process Data.

EXAMPLES Calibrations, position adjustments, and simulation.

B.2.21.6 Failure

The Device is unable to perform its intended function. The Process Data shall be marked as invalid if no part of the process data content can be provided. In the case of partially invalid process data, the process data may be marked as invalid at the discretion of the device manufacturer. The method of indicating partially invalid process data content is profile or vendor specific.

B.2.22 DetailedDeviceStatus

The parameter DetailedDeviceStatus shall provide information about currently pending Events in the Device. Events of TYPE "Error" or "Warning" and MODE "Event appears" (see A.6.4) shall be entered into the list of DetailedDeviceStatus with EventQualifier and EventCode. Upon occurrence of an Event with MODE "Event disappears", the corresponding entry in DetailedDeviceStatus shall be set to EventQualifier "0x00" and EventCode "0x0000". This way this parameter always provides the current diagnosis status of the Device. The parameter is a read-only data object. The data type is ArrayT with a maximum number of 64 array elements (Event entries). The number of array elements of this parameter is Device specific. Upon power-off or reset of the Device the contents of all array elements are set to initial settings – EventQualifier "0x00", EventCode "0x0000". This parameter is optional.

Table B.14 specifies the structure of the parameter DetailedDeviceStatus.

Table B.14 – DetailedDeviceStatus (Index 0x0025)

Sub-index	Object name	Data Type	Comment
1	Error_Warning_1	3 octets	All octets 0x00: no Error/ Warning Octet 1: EventQualifier Octet 2,3: EventCode
2	Error_Warning_2	3 octets	
3	Error_Warning_3	3 octets	
4	Error_Warning_4	3 octets	
…			
n	Error_Warning_n	3 octets	

The designer may choose the implementation of a static list, i.e. one fix array position for each Event with a specific EventCode, or a dynamic list, i.e. each Event entry is stored into the next free array position. Subindex access is not supported.

B.2.23 ProcessDataInput

The parameter ProcessDataInput shall provide the last valid process input data from the Device application. The data type and structure are identical to the Process Data In transferred in the process communication channel. The parameter is a read-only data object. This parameter is optional.

B.2.24 ProcessDataOutput

The parameter ProcessDataOutput shall provide the last valid process output data written to the Device application. The data type and structure are identical to the Process Data Out transferred in the process communication channel. The parameter is a read-only data object. This parameter is optional.

B.2.25 OffsetTime

The parameter OffsetTime (t_{offset}) allows a Device application to synchronize on M-sequence cycles of the data link layer via adjustable offset times. The data type is RecordT. Access is only possible via Subindex "0". The parameter is a read/write data object. This parameter is optional.

The structure of the parameter OffsetTime is shown in Figure B.7:

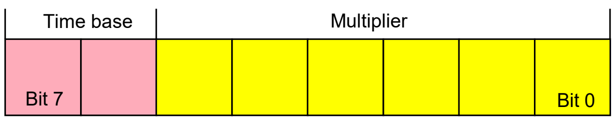


Figure B.7 – Structure of the OffsetTime

Bits 0 to 5: Multiplier

These bits contain a 6-bit factor for the calculation of the OffsetTime. Permissible values for the multiplier are 0 to 63.

Bits 6 to 7: Time Base

These bits contain the time base for the calculation of the OffsetTime.

The permissible combinations for Time Base and Multiplier are listed in Table B.15 along with the resulting values for OffsetTime. Setting both Multiplier and Time Base to zero deactivates synchronization with the help of an OffsetTime. The value of OffsetTime shall not exceed the MasterCycleTime (see B.1.3)

Table B.15 – Time base coding and values of OffsetTime

Time base encoding	Time Base value	Calculation	OffsetTime
00	0,01 ms	Multiplier × Time Base	0,01 ms to 0,63 ms
01	0,04 ms	0,64 ms + Multiplier × Time Base	0,64 ms to 3,16 ms
10	0,64 ms	3,20 ms + Multiplier × Time Base	3,20 ms to 43,52 ms
11	2,56 ms	44,16 ms + Multiplier × Time Base	44,16 ms to 126,08 ms

B.2.26 Profile parameter (reserved)

Indices 0x0031 to 0x003F are reserved for Device profiles.

NOTE Details are provided in [7].

B.2.27 Preferred Index

Preferred Indices (0x0040 to 0x00FE) can be used for vendor specific Device functions. This range of indices is considered preferred due to lower protocol overhead within the ISDU and thus higher data throughput for small data objects as compared to the Extended Index (see B.2.28).

B.2.28 Extended Index

Extended Indices (0x0100 to 0x3FFF) can be used for vendor specific Device functions.

B.2.29 Profile specific Index (reserved)

Indices 0x4000 to 0x4FFF are reserved for Device profiles.

NOTE Details are provided in [7].

Annex C
(normative)

ErrorTypes (ISDU errors)

C.1 General

An ErrorType is used within negative service confirmations of ISDUs (see A.5.2 and Table A.13) or negative acknowledgements of SMI services (see E.18). It indicates the cause of a negative confirmation of a Read or Write service. The origin of the error may be located in the Master (local) or in the Device (remote).

The ErrorType consists of two octets, the main error cause and more specific information:

- ErrorCode (high order octet)
- AdditionalCode (low order octet)

The ErrorType represents information about the incident, the origin and the instance. The permissible ErrorTypes and the criteria for their deployment are listed in C.2, C.3, and C.4. All other ErrorType values are reserved and shall not be used.

C.2 Application related ErrorTypes

C.2.1 Overview

The permissible ErrorTypes resulting from the Device application are listed in Table C.1.

Table C.1 – ErrorTypes

Incident	Error Code	Additional Code	Name	Definition
Device application error – no details	0x80	0x00	APP_DEV	See C.2.2
Index not available	0x80	0x11	IDX_NOTAVAIL	See C.2.3
Subindex not available	0x80	0x12	SUBIDX_NOTAVAIL	See C.2.4
Service temporarily not available	0x80	0x20	SERV_NOTAVAIL	See C.2.5
Service temporarily not available – local control	0x80	0x21	SERV_NOTAVAIL_LOCCTRL	See C.2.6
Service temporarily not available – Device control	0x80	0x22	SERV_NOTAVAIL_DEVCTRL	See C.2.7
Access denied	0x80	0x23	IDX_NOT_ACCESSIBLE	See C.2.8
Parameter value out of range	0x80	0x30	PAR_VALOUTOFRNG	See C.2.9
Parameter value above limit	0x80	0x31	PAR_VALGTLIM	See C.2.10
Parameter value below limit	0x80	0x32	PAR_VALLTLIM	See C.2.11
Parameter length overrun	0x80	0x33	VAL_LENOVRRUN	See C.2.12
Parameter length underrun	0x80	0x34	VAL_LENUNDRUN	See C.2.13
Function not available	0x80	0x35	FUNC_NOTAVAIL	See C.2.14

Incident	Error Code	Additional Code	Name	Definition
Function temporarily unavailable	0x80	0x36	FUNC_UNAVAILTEMP	See C.2.15
Invalid parameter set	0x80	0x40	PAR_SETINVALID	See C.2.16
Inconsistent parameter set	0x80	0x41	PAR_SETINCONSIST	See C.2.17
Application not ready	0x80	0x82	APP_DEVNOTRDY	See C.2.18
Vendor specific	0x81	0x00	UNSPECIFIC	See C.2.19
Vendor specific	0x81	0x01 to 0xFF	VENDOR_SPECIFIC	See C.2.19

C.2.2 Device application error – no details

This ErrorType shall be used if the requested service has been refused by the Device application and no detailed information of the incident is available.

C.2.3 Index not available

This ErrorType shall be used whenever a read or write access occurs to a non-existing Index with or without Subindex access.

C.2.4 Subindex not available

This ErrorType shall be used whenever a read or write access occurs to a non-existing Subindex of an existing Index.

C.2.5 Service temporarily not available

This ErrorType shall be used if a parameter is not accessible for a read or write service due to the current state of the Device application.

C.2.6 Service temporarily not available – local control

This ErrorType shall be used if a parameter is not accessible for a read or write service due to an ongoing local operation at the Device (for example operation or parameterization via an on-board Device control panel).

C.2.7 Service temporarily not available – device control

This ErrorType shall be used if a read or write service is not accessible due to a remote triggered state of the device application (for example parameterization during a remote triggered teach-in operation or calibration).

C.2.8 Access denied

This ErrorType shall be used if a Write service tries to access a read-only parameter or if a Read service tries to access a write-only parameter.

C.2.9 Parameter value out of range

This ErrorType shall be used for a write service to a parameter outside its permitted range of values. Example: enumerations (list of single values), combination of value ranges and enumeration.

C.2.10 Parameter value above limit

This ErrorType shall be used for a write service to a parameter above its specified value range.

C.2.11 Parameter value below limit

This ErrorType shall be used for a write service to a parameter below its specified value range.

C.2.12 Parameter length overrun

This ErrorType shall be used when the content of a write service to a parameter is greater than the parameter specified length. This ErrorType shall also be used, if a data object is too large to be processed by the Device application (for example ISDU buffer restriction).

C.2.13 Parameter length underrun

This ErrorType shall be used when the content of a write service to a parameter is less than the parameter specified length (for example write access of an Unsigned16 value to an Unsigned32 parameter).

C.2.14 Function not available

This ErrorType shall be used for a write service with a command value not supported by the Device application (for example a SystemCommand with a value not implemented).

C.2.15 Function temporarily unavailable

This ErrorType shall be used for a write service with a command value calling a Device function not available due to the current state of the Device application (for example a SystemCommand).

C.2.16 Invalid parameter set

This ErrorType shall be used if values sent via single parameter transfer are not consistent with other actual parameter settings (for example overlapping set points for a binary data setting; see 10.3.4).

C.2.17 Inconsistent parameter set

This ErrorType shall be used at the termination of a Block Parameter transfer with ParamDownloadEnd or ParamDownloadStore or after a DS_DownloadEnd Command, if the plausibility check shows inconsistencies (see 10.3.5, B.2.2, and 10.4.1).

C.2.18 Application not ready

This ErrorType shall be used if a read or write service is refused due to a temporarily unavailable application (for example peripheral controllers during startup).

C.2.19 Vendor specific

This ErrorType will be propagated directly to upper level processing elements as an error (no warning) by the Master.

C.3 Derived ErrorTypes

C.3.1 Overview

Derived ErrorTypes are generated in the Master AL and are caused by internal incidents or those received from the Device. Table C.2 lists the specified Derived ErrorTypes.

Table C.2 – Derived ErrorTypes

Incident	Error Code	Additional Code	Name	Definition
Master – Communication error	0x10	0x00	COM_ERR	See C.3.2
Master – ISDU timeout	0x11	0x00	I-SERVICE_TIMEOUT	See C.3.3
Device Event – ISDU error [a] (DL, Error, single shot [b], 0x5600)	0x11	0x00	I-SERVICE_TIMEOUT	See C.3.4
Device Event – ISDU illegal [a] service primitive (AL, Error, single shot [c], 0x5800)	0x11	0x00	I-SERVICE_TIMEOUT	See C.3.5
Master – ISDU checksum error	0x56	0x00	M_ ISDU_CHECKSUM	See C.3.6
Master – ISDU illegal service primitive	0x57	0x00	M_ ISDU_ILLEGAL	See C.3.7
Device Event – ISDU buffer overflow [a] (DL, Error, single shot [b], 0x5200)	0x80	0x33	VAL_LENOVRRUN	See C.3.8 and C.2.12
Key: a) Events from legacy Devices shall be redirected in compatibility mode to the derived ErrorType b) according [8]: Event qualifier code for DL, Error, single shot result is 0x72 c) according [8]: Event qualifier code for AL, Error, single shot result is 0x73				

C.3.2 Master – Communication error

The Master generates a negative service response with this ErrorType if a communication error occurred during a read or write service, for example the SDCI connection is interrupted.

C.3.3 Master – ISDU timeout

The Master generates a negative service response with this ErrorType, if a Read or Write service is pending longer than the specified I-Service timeout (see Table 102) in the Master.

C.3.4 Device Event – ISDU error

If the Master received an Event with the EventQualifier (see A.6.4: DL, Error, Event single shot) and the EventCode 0x5600, a negative service response indicating a service timeout is generated and returned to the requester (see C.3.3).

C.3.5 Device Event – ISDU illegal service primitive

If the Master received an Event with the EventQualifier (see A.6.4: AL, Error, Event single shot) and the EventCode 0x5800, a negative service response indicating a service timeout is generated and returned to the requester (see C.3.3).

C.3.6 Master – ISDU checksum error

The Master generates a negative service response with this ErrorType, if its data link layer detects an ISDU checksum error.

C.3.7 Master – ISDU illegal service primitive

The Master generates a negative service response with this ErrorType, if its data link layer detects an ISDU illegal service primitive.

C.3.8 Device Event – ISDU buffer overflow

If the Master received an Event with the EventQualifier (see A.6.4: DL, Error, Event single shot) and the EventCode 0x5200, a negative service response indicating a parameter length overrun is generated and returned to the requester (see C.2.12).

C.4 SMI related ErrorTypes

C.4.1 Overview

The Master returns SMI related ErrorTypes within a negative response (Result (-) while performing an SMI service (see 11.2). Table C.3 lists the SMI related ErrorTypes.

Table C.3 – SMI related ErrorTypes

Incident	Error Code	Additional Code	Name
ArgBlock unknown	0x40	0x01	ARGBLOCK_NOT_SUPPORTED
Incorrect ArgBlock content type	0x40	0x02	ARGBLOCK_INCONSISTENT
Device not communicating	0x40	0x03	DEVICE_NOT_ACCESSIBLE
Service unknown	0x40	0x04	SERVICE_NOT_SUPPORTED
Process Data not accessible	0x40	0x05	DEVICE_NOT_IN_OPERATE
Insufficient memory	0x40	0x06	MEMORY_OVERRUN
Incorrect Port number	0x40	0x11	PORT_NUM_INVALID
Incorrect ArgBlock content	0x40	0x30	ARGBLOCK_VALOUTOFRANGE
Incorrect ArgBlock length	0x40	0x34	ARGBLOCK_LENGTH_INVALID
Master busy	0x40	0x36	SERVICE_TEMP_UNAVAILABLE
Inconsistent DS data	0x40	0x39	INCONSISTENT_DS_DATA
Device / Master error	ee	aa	Propagated error, for "ee" and "aa" see Annex C.2 and C.3
Reserved	0x40	0x80 to 0xFF	Vendor specific

C.4.2 ArgBlock unknown

This ErrorType shall be used if the requested ArgBlockID is unknown to the SMI.

C.4.3 Incorrect ArgBlock content type

This ErrorType shall be used if the SMI service detects errors in the structure of the provided ArgBlock.

C.4.4 Device not communicating

This ErrorType shall be used if the Port is not communicating with the Device.

C.4.5 Service unknown

This ErrorType shall be used if a requested SMI service is not supported by the Master.

C.4.6 Process Data not accessible

This ErrorType shall be used if the requested Process Data cannot be accessed in current state of communication.

C.4.7 Insufficient memory

This ErrorType shall be used if the requested SMI service requires more memory space.

C.4.8 Incorrect Port number

This ErrorType shall be used if the requested Port number is invalid.

C.4.9 Incorrect ArgBlock content

This ErrorType shall be used if the actual ArgBlock content is not consistent or contains invalid data.

C.4.10 Incorrect ArgBlock length

This ErrorType shall be used if the actual ArgBlock length does not correspond to the ArgBlockID.

C.4.11 Master busy

This ErrorType shall be used if the SMI service is blocked due to other running processes.

C.4.12 Inconsistent DS data

This ErrorType shall be used if Data Storage is not supported or Data Storage is not activated on this Port or Data Storage content is not consistent with Port configuration, for example VendorID does not match.

C.4.13 Device/Master error

These ErrorTypes from Device or Master Port are propagated if the requested SMI service has been denied by the Device.

Annex D
(normative)

EventCodes (diagnosis information)

D.1 General

The concept of Events is described in 7.3.8.1 and the general structure and encoding of Events is specified in Clause A.6. Whenever the StatusCode indicates an Event in case of a Device or a Master incident, the associated EventCode shall be provided as diagnosis information. As specified in A.6, the Event entry contains an EventCode in addition to the EventQualifier. The EventCode identifies an actual incident. Permissible values for EventCode are listed in Table D.1; all other EventCode values are reserved and shall not be used.

D.2 EventCodes for Devices

Table D.1 lists the specified EventCode identifiers and their definitions for Devices (Source = "REMOTE"). The EventCodes are created by the technology specific Device application (instance = APP).

Table D.1 – EventCodes for Devices

EventCode ID	Definition and recommended maintenance action	Preferred DeviceStatus Value (NOTE 1)	Type (NOTE 2)
0x0000	No malfunction	0	Notification
0x0001 to 0x0FFF	Reserved		
0x1000	General malfunction – unknown error	4	Error
0x1001 to 0x17FF	Reserved		
0x1800 to 0x18FF	Vendor specific		
0x1900 to 0x3FFF	Reserved		
0x4000	Temperature fault – Overload	4	Error
0x4001 to 0x420F	Reserved		
0x4210	Device temperature overrun – Clear source of heat	2	Warning
0x4211 to 0x421F	Reserved		
0x4220	Device temperature underrun – Insulate Device	2	Warning
0x4221 to 0x4FFF	Reserved		
0x5000	Device hardware fault – Device exchange	4	Error
0x5001 to 0x500F	Reserved		
0x5010	Component malfunction – Repair or exchange	4	Error
0x5011	Non volatile memory loss – Check batteries	4	Error
0x5012	Batteries low – Exchange batteries	2	Warning
0x5013 to 0x50FF	Reserved		
0x5100	General power supply fault – Check availability	4	Error
0x5101	Fuse blown/open – Exchange fuse	4	Error

EventCode ID	Definition and recommended maintenance action	Preferred DeviceStatus Value (NOTE 1)	Type (NOTE 2)
0x5102 to 0x510F	Reserved		
0x5110	Primary supply voltage overrun – Check tolerance	2	Warning
0x5111	Primary supply voltage underrun – Check tolerance	2	Warning
0x5112	Secondary supply voltage fault (Port Class B) – Check tolerance	2	Warning
0x5113 to 0x5FFF	Reserved		
0x6000	Device software fault – Check firmware revision	4	Error
0x6001 to 0x631F	Reserved		
0x6320	Parameter error – Check data sheet and values	4	Error
0x6321	Parameter missing – Check data sheet	4	Error
0x6322 to 0x634F	Reserved		
0x6350	Reserved		
0x6351 to 0x76FF	Reserved		
0x7700	Wire break of a subordinate device – Check installation	4	Error
0x7701 to 0x770F	Wire break of subordinate device 1 …device 15 – Check installation	4	Error
0x7710	Short circuit – Check installation	4	Error
0x7711	Ground fault – Check installation	4	Error
0x7712 to 0x8BFF	Reserved		
0x8C00	Technology specific application fault – Reset Device	4	Error
0x8C01	Simulation active – Check operational mode	3	Warning
0x8C02 to 0x8C0F	Reserved		
0x8C10	Process variable range overrun – Process Data uncertain	2	Warning
0x8C11 to 0x8C1F	Reserved		
0x8C20	Measurement range exceeded – Check application	4	Error
0x8C21 to 0x8C2F	Reserved		
0x8C30	Process variable range underrun – Process Data uncertain	2	Warning
0x8C31 to 0x8C3F	Reserved		
0x8C40	Maintenance required – Cleaning	1	Warning
0x8C41	Maintenance required – Refill	1	Warning
0x8C42	Maintenance required – Exchange wear and tear parts	1	Warning
0x8C43 to 0x8C9F	Reserved		
0x8CA0 to 0x8DFF	Vendor specific		
0x8E00 to 0xAFFF	Reserved		

EventCode ID	Definition and recommended maintenance action	Preferred DeviceStatus Value (NOTE 1)	Type (NOTE 2)
0xB000 to 0xB0FF	Reserved for Safety extensions	See [10]	See [10]
0xB100 to 0xBFFF	Reserved for profiles		
0xC000 to 0xFF90	Reserved		
0xFF91	Data Storage upload request ("DS_UPLOAD_REQ") – internal, not visible to user	0	Notification
0xFF92 to 0xFFAF	Reserved		
0xFFB0 to 0xFFB7	Reserved for Wireless extensions	See [11]	See [11]
0xFFB8 to 0xFFFF	Reserved		
NOTE 1 See B.2.21 for a description of this parameter NOTE 2 See Table A.19 for a description of Event types			

D.3 EventCodes for Ports

Table D.2 lists the specified EventCode identifiers and their definitions for Ports. The EventCodes are created by the Master (Source = "Master/Port", see Table A.18, and "application" (APP) or "communication system" (SYS) as INSTANCE, see Table Table A.17). EventCode identifiers 0xFF21 to 0xFFFF are internal system information and shall not be visible to users.

The following rules apply:

– Port Events referring to SDCI communication are mandatory (exceptions 0xFF26/0xFF27) and are specified in detail (Event INSTANCE = SYS). The other Port Events (Event INSTANCE = APP) are optional.

– Each appearing Port Event of Type "Error" requires a disappearing Port Event whenever the cause of the Error has been fixed.

– Occurring PortStatusInfo "PORT_DIAG" leads to an appearing EventCode 0x180x or 0x600x depending on "SYS" Error (see Table 126).

– Leaving PortStatusInfo "PORT_DIAG" to others leads to disappearing EventCodes for each pending Error (0x180x).

– Every appearing/disappearing Event leads to an update of the DiagEntry section in the PortStatusList (see Table E.4).

Table D.2 – EventCodes for Ports

EventCode ID	Definition and recommended maintenance action	Event INSTANCE	Type
0x0000 to 0x17FF	Reserved		
0x1800	No Device (communication) - Occurring PortStatusInfo "NO_Device" leads to an appearing EventCode 0x1800 - Appearing EventCode 0x1800 causes disappearing of all pending EventCodes of INSTANCE "SYS". - Leaving PortStatusInfo "NO_DEVICE" to others leads to a disappearing EventCode 0x1800	SYS	Error

EventCode ID	Definition and recommended maintenance action	Event INSTANCE	Type
0x1801	Startup parametrization error – check parameter	APP	Error
0x1802	Incorrect VendorID – Inspection Level mismatch Trigger: SMI_PortEvent(0x1802) by SM_PortMode (COMP_FAULT)	SYS	Error
0x1803	Incorrect DeviceID – Inspection Level mismatch Trigger: SMI_PortEvent(0x1803) by SM_PortMode (COMP_FAULT)	SYS	Error
0x1804	Short circuit at C/Q – check wire connection	APP	Error
0x1805	Overtemperature – check Master temperature and load	APP	Error
0x1806	Short circuit at L+ – check wire connection	APP	Error
0x1807	Overcurrent at L+ – check power supply (e.g. L1+)	APP	Error
0x1808	Reserved		
0x1809	Backup inconsistency – memory out of range (2048 octets) Trigger: SMI_PortEvent (0x1809) by DS_Fault (SizeCheck_Fault)	SYS	Error
0x180A	Backup inconsistency – identity fault Trigger: SMI_PortEvent (0x180A) by DS_Fault (Identification_Fault)	SYS	Error
0x180B	Backup inconsistency – Data Storage unspecific error Trigger: SMI_PortEvent (0x180B) by DS_Fault (All other incidents)	SYS	Error
0x180C	Backup inconsistency – upload fault Trigger: SMI_PortEvent (0x180C) by DS_Fault (Upload)	SYS	Error
0x180D	Parameter inconsistency – download fault Trigger: SMI_PortEvent (0x180D) by DS_Fault (Download)	SYS	Error
0x180E	P24 (Class B) missing or undervoltage	APP	Error
0x180F	Short circuit at P24 (Class B) – check wire connection (e.g. L2+)	APP	Error
0x1810	Short circuit at I/Q – check wiring	APP	Error
0x1811	Short circuit at C/Q (if digital output) – check wiring	APP	Error
0x1812	Overcurrent at I/Q – check load	APP	Error
0x1813	Overcurrent at C/Q (if digital output) – check load	APP	Error
0x1814 to 0x1EFF	Reserved		
0x1F00 to 0x1FFF	Vendor specific		
0x2000 to 0x2FFF	Safety extensions		See [10]
0x3000 to 0x3FFF	Wireless extensions		See [11]
0x4000 to 0x5FFF	Reserved		
0x6000	Invalid cycle time Trigger: SM_PortMode (CYCTIME_FAULT)	SYS	Error
0x6001	Revision fault – incompatible protocol version Trigger: SM_PortMode (REVISION_FAULT)	SYS	Error
0x6002	ISDU batch failed – parameter inconsistency?	SYS	Error
0x6003 to 0xFF20	Reserved		
0xFF21 a)	DL: Device plugged in ("NEW_SLAVE") – PD stop Trigger: SM_PortMode (COMREADY); see Figure 71 (T10)		Notification
0xFF22 a)	Device communication lost ("DEV_COM_LOST")		Notification
0xFF23 a)	Data Storage identification mismatch ("DS_IDENT_MISMATCH")		Notification
0xFF24 a)	Data Storage buffer overflow ("DS_BUFFER_OVERFLOW")		Notification
0xFF25 a)	Data Storage parameter access denied ("DS_ACCESS_DENIED")		Notification

EventCode ID	Definition and recommended maintenance action	Event INSTANCE	Type
0xFF26 b)	Port status changed – Use "SMI_PortStatus" service for Port status in detail. Each change of "PortStatusInfo" causes this Event via SMI_PortEvent	SYS	Notification
0xFF27 b)	Data Storage upload completed and new data object available. Each completion of a Data Storage upload causes this Event via SMI_PortEvent	SYS	Notification
0xFF28 to 0xFF30	Reserved		
0xFF31 a)	DL: Incorrect Event signalling ("EVENT") Trigger: none		Notification
0xFF32 to 0xFFFF	Reserved		
	a) No more required due to SMI Event concept. Not recommended for implementations. b) These Events are optional.		

Annex E
(normative)

Coding of ArgBlocks

E.1 General

The purpose of ArgBlocks is explained in 11.2.2. Each ArgBlock is uniquely defined by its ArgBlock identifier (ArgBlockID) and its ArgBlock length (ArgBlockLength). Extension of ArgBlocks by just using a larger ArgBlock length is not permitted. Manufacturer specific ArgBlocks are possible by using the service groups B to E (see Figure E.1).

Transmission of ArgBlocks is following the convention in Figure E.1 as octet stream beginning with octet offset 0.

The four-nibble structure of the ArgBlockID is shown in Figure E.1

The ArgBlockID "0x0000" is reserved. The fourth nibble (N4) is assigned to SMI service groups. The third nibble (N3) is assigned to domains and to SMI management. Nibble 1 (N1) and nibble 2 (N2) define ArgBlocks within the particular domain.

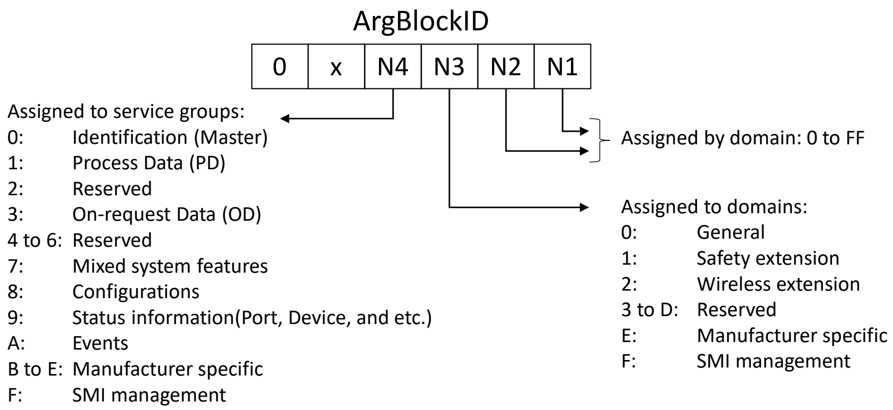


Figure E.1 – Assignment of ArgBlock identifiers

Table E.1 shows all defined ArgBlock types and their IDs including those for system extensions in order to avoid ambiguities. ArgBlockIDs are assigned by the IO-Link Community.

Table E.1 – ArgBlock types and their ArgBlockIDs

ArgBlock type	ArgBlockID	Definition	Used by SMI_xxx services
MasterIdent	0x0001	Annex E.2	SMI_MasterIdentification (see 11.2.4)
FSMasterAccess	0x0100	[10]	–
WMasterConfig	0x0200	[11]	–
PDIn	0x1001	Annex E.10	SMI_PDIn (see 11.2.17)
PDOut	0x1002	Annex E.11	SMI_PDOut (see 11.2.18)
PDInOut	0x1003	Annex E.12	SMI_PDInOut (see 11.2.19)
SPDUIn	0x1101	[10]	–
SPDUOut	0x1102	[10]	–
PDInIQ	0x1FFE	Annex E.13	SMI_PDInIQ (see 11.2.20)
PDOutIQ	0x1FFF	Annex E.14	SMI_PDOutIQ (see 11.2.21) SMI_PDReadbackOutIQ (see 11.2.22)
On-requestData	0x3000	Annex E.5	SMI_DeviceWrite (see 11.2.10)
	0x3001		SMI_DeviceRead (see 11.2.11)

ArgBlock type	ArgBlockID	Definition	Used by SMI_xxx services
DS_Data	0x7000	Annex E.6	SMI_DSToParServ (see 11.2.8) SMI_ParServToDS (see 11.2.9)
DeviceParBatch	0x7001	Annex E.7	SMI_ParamWriteBatch (see 11.2.12) SMI_ParamReadBatch (see 11.2.13)
IndexList	0x7002	Annex E.8	SMI_ParamReadBatch (see 11.2.13)
PortPowerOffOn	0x7003	Annex E.9	SMI_PortPowerOffOn (see 11.2.14)
PortConfigList	0x8000	Annex E.3	SMI_PortConfiguration (see 11.2.5) SMI_ReadBackPortConfiguration (see 11.2.6)
FSPortConfigList	0x8100	[10]	–
WTrackConfigList	0x8200	[11]	–
PortStatusList	0x9000	Annex E.4	SMI_PortStatus (see 11.2.7)
FSPortStatusList	0x9100	[10]	–
WTrackStatusList	0x9200	[11]	–
WTrackScanResult	0x9201	[11]	–
DeviceEvent	0xA000	Annex E.15	SMI_DeviceEvent (see 11.2.15)
PortEvent	0xA001	Annex E.16	SMI_PortEvent (11.2.16)
VoidBlock	0xFFF0	Annex E.17	SMI service management
JobError	0xFFFF	Annex E.18	SMI service management

E.2 MasterIdent

This ArgBlock is used by the service SMI_MasterIdentification (see 11.2.4). Table E.2 shows coding of the MasterIdent ArgBlock.

Table E.2 – MasterIdent

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x0001
2	VendorID	Unique VendorID of the Master (see B.1.8)	Unsigned16	1 to 0xFFFF
4	MasterID	4 octets long vendor specific unique identification of the Master	Unsigned32	1 to 0xFFFFFFFF
8	MasterType	0: Unspecific (manufacturer specific) 1: Reserved 2: Master acc. to this specification or later 3: FS_Master; see [10] 4: W_Master; see [11] 5 to 255: Reserved	Unsigned8	0 to 0xFF
9	Features_1	7 6 5 4 3 2 1 0 Bit 0: DeviceParBatch (SMI_ParamWriteBatch) 0 = not supported 1 = supported Bit 1: DeviceParBatch (SMI_ParamReadBatch) 0 = not supported 1 = supported Bit 2: PortPowerOffOn (SMI_PortPowerOffOn) 0 = not supported 1 = supported Bit 3 to 7: Reserved (= 0)	Unsigned8	0 to 0xFF
10	Features_2	7 6 5 4 3 2 1 0 Reserved for future use (= 0)	Unsigned8	0 to 0xFF
11	MaxNumberOfPorts	Maximum number (n) of ports of this Master	Unsigned8	1 to 0xFF

Octet Offset	Element name	Definition	Data type	Values
12	PortTypes	Array indicating for all *n* ports the type of port 0: Class A 1: Class A with PortPowerOffOn 2: Class B; see 5.4.2 3: FS_Port_A without OSSDe; see [10] 4: FS_Port_A with OSSDe; see [10] 5: FS_Port_B; see [10] 6: W_Port; see [11] 7 to 127: Reserved 128 to 255: Manufacturer specific	Array [1 to *n*] of Unsigned8	1 to 6

E.3 PortConfigList

This ArgBlock is used by the services SMI_PortConfiguration (see 11.2.5) and SMI_Readback-PortConfiguration (see 11.2.6). Table E.3 shows the coding of the PortConfigList ArgBlock.

Table E.3 – PortConfigList

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x8000
2	PortMode [c]	This element contains the port mode expected by the SMI client, e.g. gateway application. All modes are mandatory. They shall be mapped to the Target Modes of "SM_SetPortConfig" (see 9.2.2.2). 0: DEACTIVATED (SM: INACTIVE → Port is deactivated; input and output Process Data are "0"; Master shall not perform activities at this port) 1: IOL_MANUAL (SM: CFGCOM → Target Mode based on user defined configuration including validation of RID, VID, DID) 2: IOL_AUTOSTART [a] (SM: AUTOCOM →Target Mode w/o configuration and w/o validation of VID/DID; RID gets highest revision the Master is supporting; Validation: NO_CHECK) 3: DI_C/Q (Pin 4 at M12) [b] (SM: DI → Port in input mode SIO) 4: DO_C/Q (Pin 4 at M12) [b] (SM: DO → Port in output mode SIO) 5 to 48: Reserved for future versions 49 to 96: Reserved for extensions (see [10], [11]) 97 to 255: Manufacturer specific	Unsigned8	0 to 0xFF
3	Validation&Backup	This element contains the InspectionLevel to be performed by the Device and the Backup/Restore behavior. 0: No Device check 1: Type compatible Device V1.0 2: Type compatible Device V1.1 3: Type compatible Device V1.1, Backup + Restore 4: Type compatible Device V1.1, Restore 5 to 255: Reserved	Unsigned8	0 to 0xFF

Octet Offset	Element name	Definition	Data type	Values
4	I/Q behavior (manufacturer or profile specific, see [10], [11])	This element defines the behavior of the I/Q signal (Pin 2 at M12 connector) 0: Not supported 1: Digital Input 2: Digital Output 3: Reserved 4: Reserved 5: Power 2 (Port class B) 6 to 255: Reserved	Unsigned8	0 to 0xFF
5	PortCycleTime	This element contains the port cycle time expected by the SMI client. AFAP is default. They shall be mapped to the ConfiguredCycleTime of "SM_SetPortConfig" (see 9.2.2.2) 0: AFAP (As fast as possible – SM: FreeRunning → Port cycle timing is not restricted. Default value in port mode IOL_MANUAL) 1 to 255: TIME (SM: For coding see Table B.3. Device shall achieve the indicated port cycle time. An error shall be created if this value is below MinCycleTime of the Device or in case of other misfits)	Unsigned8	0 to 0xFF
6	VendorID	This element contains the 2 octets long VendorID expected by the SMI client (see B.1.8)	Unsigned16	1 to 0xFFFF
8	DeviceID	This element contains the 3 octets long DeviceID expected by the SMI client (see B.1.9)	Unsigned32	1 to 0xFFFFFF
a In PortMode "IOL_Autostart" parameters VendorID, DeviceID, and Validation&Backup are treated don't care. b In PortModes "DI_C/Q" and "DO_C/Q" parameters Validation&Backup, VendorID, DeviceID, and PortCycleTime are treated don't care. c It is recommended to state the default setting of the PortMode in the Master manual or integration specification				

E.4 PortStatusList

This ArgBlock is used by the service SMI_PortStatus (see 11.2.7). Table E.4 shows the coding of the ArgBlock "PortStatusList". It refers to the state machine of the Configuration Manager in Figure 101 and shows its current states.

Content of "PortStatusInfo" is derived from "PortMode" in 9.2.2.4. Values not available shall be set to "0".

Table E.4 – PortStatusList

Octet	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x9000

Octet	Element name	Definition	Data type	Values
2	PortStatusInfo	This element contains status information of the Port. 0: NO_DEVICE No communication (COMLOST). However, Port configuration IOL_MANUAL or IOL_AUTOSTART was set (see Table E.3). 1: DEACTIVATED Port configuration DEACTIVATED was set (see Table E.3). 2: PORT_DIAG This value to be set if the Port encounters a failure during startup, validation, or Data Storage (group error). Device is in PREOPERATE and DiagEntry contains the diagnosis cause. 3: Reserved 4: OPERATE This value to be set if the Device is in OPERATE, even in case of Device error. 5: DI_C/Q Port configuration "DI" was set (see Table E.3). 6: DO_C/Q Port configuration "DO" was set (see Table E.3). 7 to 8: Reserved for IO-Link Safety [10] 9 to 199: Reserved 200 to 249: Manufacturer specific 250 to 253: Reserved 254: PORT_POWER_OFF Shutdown of Port is active caused by SMI_PortPowerOffOn 255: NOT_AVAILABLE PortStatusInfo currently not available	Unsigned8 (enum)	0 to 0xFF
3	PortQualityInfo a)	This element contains status information on Process Data (see 8.2.2.12). Bit0: 0 = VALID 1 = INVALID Bit1: 0 = PDOUTVALID 1 = PDOUTINVALID Bit2 to Bit7: Reserved	Unsigned8	–
4	RevisionID	This element contains information of the SDCI protocol revision of the Device (see B.1.5) 0: NOT_DETECTED (No communication at that port) <>0: Copied from Direct parameter page, address 4	Unsigned8	0 to 0xFF
5	TransmissionRate	This element contains information on the effective port transmission rate. 0: NOT_DETECTED (No communication at that port) 1: COM1 (transmission rate 4,8 kbit/s) 2: COM2 (transmission rate 38,4 kbit/s) 3: COM3 (transmission rate 230,4 kbit/s) 4 to 255: Reserved for future use	Unsigned8	0 to 0xFF

Octet	Element name	Definition	Data type	Values
6	MasterCycleTime	This element contains information on the Master cycle time. For coding see B.1.3.	Unsigned8	–
7	InputDataLength	This element contains the input data length as number of octets of the Device provided by the PDIn service (see Annex E.10)	Unsigned8	0 to 0x20
8	OutputDataLength	This element contains the output data length as number of octets for the Device accepted by the PDOut service (see Annex E.11	Unsigned8	0 to 0x20
9	VendorID	This element contains the 2 octets long VendorID connected to the SMI client	Unsigned16	0 to 0xFFFF
11	DeviceID	This element contains the 3 octets long DeviceID connected to the SMI client	Unsigned32	0 to 0xFFFFFF
15	NumberOfDiags	This element contains the provided number x of pending Events via DiagEntries	Unsigned8	0 to 0xFF
16 + 3*(n-1)	DiagEntry0 … DiagEntry(x-1)	These elements contain the "EventQualifier" and "EventCode" of pending Events. See B.2.22 for coding and how to deal with "Event appears / disappears".	Struct Unsigned8/16	–
Key	n: 1 .. x a) the PortQualityInfo shall be ignored in case of DI, DO, or not OPERATE			

E.5 On-request_Data

This ArgBlock with ArgBlockID 0x3000 is used by the service SMI_DeviceWrite (see 11.2.10) and with ArgBlockID 0x3001 (Index only) by the service SMI_DeviceRead (see 11.2.11). Table E.5 shows the coding of the ArgBlockType "On-request_Data".

Table E.5 – On-request_Data

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x3000 (Write) 0x3001 (Read)
2	Index	This element contains the Index to be used for the AL_Write or AL_Read service	Unsigned16	0 to 0xFFFF
4	Subindex	This element contains the Subindex to be used for the AL_Write or AL_Read service	Unsigned8	0 to 0xFF
5 to n	On-request Data	This element contains the On-request Data for ArgBlock 0x3000 if available.	Octet string	–

E.6 DS_Data

This ArgBlock is used by the services SMI_DSToParServ (see 11.2.8) and SMI_ParServToDS (see 11.2.9). Table E.6 shows the coding of the ArgBlockType "DS_Data".

Table E.6 – DS_Data

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x7000
2 to n	DataStorageObject	This element contains the Device parameter set coded according to 11.4.2 (Table G.2 followed by Table G.1)	Record (octet string)	0 to 2×2^{10} +12

E.7 DeviceParBatch

This ArgBlock provides means to transfer a large number of Device parameters via a number of ISDU write or read requests to the Device. It is used by the services SMI_ParamWriteBatch (see 11.2.12) or SMI_ParamReadBatch (see 11.2.13). Table E.7 shows the coding of the ArgBlockType "DeviceParBatch".

Table E.7 – DeviceParBatch

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x7001
2	Object1_Index	Index of 1st parameter	Unsigned16	0 to 0xFFFF
4	Object1_Subindex	Subindex of 1st parameter	Unsigned8	0 to 0xFF
5	Object1_Length	Length of parameter record or	Unsigned8	0 to 0xE8
		ISDU error (implicitly 2 octets)	Unsigned8	0xFF (error)
6	Object1_Data	Parameter record or	Record	0 to r
		ISDU ErrorType (return value)	Unsigned16	ErrorType
6+r	Object2_Index	Index of 2nd parameter	Unsigned16	0 to 0xFFFF
6+r+2	Object2_Subindex	Subindex of 2nd parameter	Unsigned8	0 to 0xFF
6+r+3	Object2_Length	Length of parameter record or	Unsigned8	0 to 0xE8
		ISDU error (implicitly 2 octets)	Unsigned8	0xFF (error)
6+r+4	Object2_Data	Parameter record or	Record	0 to s
		ISDU ErrorType (return value)	Unsigned16	ErrorType
…				
…	Object$x$_Index	Index of x^{th} parameter	Unsigned16	0 to 0xFFFF
…	Object$x$_Subindex	Subindex of x^{th} parameter	Unsigned8	0 to 0xFF
…	Object$x$_Length	Length of parameter record or	Unsigned8	0 to 0xE8
		ISDU error (implicitly 2 octets)	Unsigned8	0xFF (error)
…	Object$x$_Data	Parameter record or	Record	0 to t
		ISDU ErrorType (return value)	Unsigned16	ErrorType
In case of SMI_ParamWriteBatch, this ArgBlock will return ErrorType "0x0000" for each successfully written object				

E.8 IndexList

This ArgBlock provides a list of the Indices of several requested Device parameters to be retrieved from a Device via the service SMI_ParamReadBatch (see 11.2.13). Table E.8 shows the coding of the ArgBlockType "IndexList".

Table E.8 – IndexList

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x7002
2	Object1_Index	Index of 1st object	Unsigned16	0 to 0xFFFF
4	Object1_Subindex	Subindex of 1st object	Unsigned8	0 to 0xFF
5	Object2_Index	Index of 2nd object	Unsigned16	0 to 0xFFFF
7	Object2_Subindex	Subindex of 2nd object	Unsigned8	0 to 0xFF
8	Object3_Index	Index of 3rd object	Unsigned16	0 to 0xFFFF

Octet Offset	Element name	Definition	Data type	Values
10	Object3_Subindex	Subindex of 3rd object	Unsigned8	0 to 0xFF
…				

E.9 PortPowerOffOn

Table E.9 shows the ArgBlockType "PortPowerOffOn". The service "SMI_PortPowerOffOn" (see 11.2.14) together with this ArgBlock can be used for energy saving purposes during production stops or alike, the dynamic behaviour is defined in 11.8. Minimum PowerOffTime shall be 500 ms.

Table E.9 – PortPowerOffOn

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x7003
2	PortPowerMode	0: One time switch off (PowerOffTime) 1: Switch PortPowerOff (permanent) 2: Switch PortPowerOn (permanent)	Unsigned8	–
3	PowerOffTime	Duration of Master port power off (ms). See also [10].	Unsigned16	0x01F4 to 0xFFFF

E.10 PDIn

This ArgBlock provides means to retrieve input Process Data from the InBuffer within the Master. It is used by the service SMI_PDIn (see 11.2.17). Table E.10 shows the coding of the "PDIn" ArgBlockType.

Mapping principles of input Process Data (PD) are specified in 11.7.2. The following rules apply for the ArgBlock PDIn:

- The first 2 octets are occupied by the ArgBlockID (0x1001);
- The third octet (offset = 2) carries the Port Qualifier Information (PQI);
- The fourth octet specifies the length of input Process Data (cyclic values or the DI bit on the C/Q line);
- Subsequent octets are occupied by the input Process Data of the Device.

Table E.10 – PDIn

Octet offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x1001
2	PQI	Port Qualifier Information a)	Unsigned8	–
3	InputDataLength	This element contains the length of the Device's input Process Data contained in the following elements.	Unsigned8	0 to 0x20
4	PDI0	Input Process Data (octet 0)	Unsigned8	0 to 0xFF
5	PDI1	Input Process Data (octet 1)	Unsigned8	0 to 0xFF
…				
InputDataLength + 4	PDI*n*	Input Process Data (octet *n*)	Unsigned8	0 to 0xFF
Key: a) the PQI shall be ignored in case of DI, DO, or not OPERATE, see 11.7.2 Bit 7				

E.11 PDOut

This ArgBlock provides means to transfer output Process Data to the OutBuffer within the Master. It is used by the service SMI_PDOut (see 11.2.18). Table E.11 shows coding of the "PDOut" ArgBlockType.

Mapping principles of output Process Data (PD) are specified in 11.7.3. The following rules apply for the ArgBlock PDOut:

- The first 2 octets are occupied by the ArgBlockID (0x1002);
- The third octet (offset = 2) carries the port qualifier (OE);
- The fourth octet specifies the length of output Process Data (cyclic values or the DO bit on the C/Q line);
- Subsequent octets are occupied by the output Process Data, which are propagated to the Device.

Table E.11 – PDOut

Octet offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x1002
2	OE	Output Enable	Unsigned8	0x00 to 0x01
3	OutputDataLength	This element contains the length of the output Process Data for the Device contained in the following elements.	Unsigned8	0 to 0x20
4	PDO0	Output Process Data (octet 0)	Unsigned8	0 to 0xFF
5	PDO1	Output Process Data (octet 1)	Unsigned8	0 to 0xFF
…				
OutputDataLength + 4	PDOm	Output Process Data (octet m)	Unsigned8	0 to 0xFF

E.12 PDInOut

This ArgBlock provides means to retrieve input Process Data from the InBuffer and output Process Data from the OutBuffer within the Master. It is used by the service SMI_PDInOut (see 11.2.19). Table E.12 shows the coding of the "PDInOut" ArgBlockType using mapping principles of Annex E.10 and Annex E.11.

Table E.12 – PDInOut

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x1003
2	PQI	Port Qualifier Information a)	Unsigned8	–
3	OE	Output Enable b)	Unsigned8	–
4	InputDataLength	This element contains the length of the Device's input Process Data contained in the following elements.	Unsigned8	0 to 0x20
5	PDI0 *	Input Process Data (octet 0)	Unsigned8	0 to 0xFF
6	PDI1 *	Input Process Data (octet 1)	Unsigned8	0 to 0xFF
…				
InputDataLength +4	PDIn *	Input Process Data (octet n)	Unsigned8	0 to 0xFF

Octet Offset	Element name	Definition	Data type	Values
InputDataLength +5	OutputDataLength	This element contains the length of the output Process Data for the Device contained in the following elements.	Unsigned8	0 to 0x20
InputDataLength + 6	PDO0 **	Output Process Data (octet 0)	Unsigned8	0 to 0xFF
InputDataLength + 7	PDO1 **	Output Process Data (octet 1)	Unsigned8	0 to 0xFF
…				
InputDataLength + OutputDataLength +5	PDO*m* **	Output Process Data (octet *m*)	Unsigned8	0 to 0xFF
Key: a) the PQI shall be ignored in case of DI, DO, or not OPERATE, see 11.7.2 Bit 7 b) The OutputEnable shall mirror the OutputEnable set by the PDOut ArgBlock				

E.13 PDInIQ

This ArgBlock provides means to retrieve input Process Data (I/Q signal) from the InBuffer within the Master. It is used by the service SMI_PDInIQ (see 11.2.20). Table E.13 shows the coding of the "PDInIQ" ArgBlockType.

Mapping principles of input Process Data (PD) are specified in 11.7.2. The following rules apply for the ArgBlock PDInIQ:

- The first 2 octets are occupied by the ArgBlockID (0x1FFE);
- Subsequent octet is occupied by the input Process Data of the signal line;
- Padding (unused) bits shall be filled with "0".

Table E.13 – PDInIQ

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x1FFE
2	PDI0	Input Process Data I/Q signal (octet 0)	Unsigned8	0 to 0x01

E.14 PDOutIQ

This ArgBlock provides means to transfer output Process Data (I/Q signal) to the OutBuffer within the Master. It is used by the services SMI_PDOutIQ (see 11.2.21) and SMI_PDReadbackOutIQ (see 11.2.22). Table E.14 shows the coding of the "PDOutIQ" ArgBlockType.

Mapping principles of output Process Data (PD) are specified in 11.7.3. The following rules apply for the ArgBlock PDOutIQ:

- The first 2 octets are occupied by the ArgBlockID (0x1FFF)
- Subsequent octet is occupied by the output Process Data that is propagated to the signal line.
- Padding (unused) bits shall be filled with "0"

Table E.14 – PDOutIQ

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0x1FFF
2	PDO0	Output Process Data I/Q signal (octet 0)	Unsigned8	0 to 0x01

E.15 DeviceEvent

This ArgBlock is used by the services SMI_DeviceEvent (see 11.2.15). Table E.15 shows the coding of the ArgBlockType "DeviceEvent".

Table E.15 – DeviceEvent

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0xA000
2	EventQualifier	EventQualifier according Annex A.6.4.	Unsigned8	0 to 0xFF
3,4	EventCode	EventCode according to Table D.1	Unsigned16	0 to 0xFFFF

E.16 PortEvent

This ArgBlock is used by the services SMI_PortEvent (see 11.2.16). Table E.16 shows the coding of the ArgBlockType "PortEvent".

Table E.16 – PortEvent

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0xA001
2	EventQualifier	EventQualifier according Annex A.6.4.	Unsigned8	0 to 0xFF
3,4	EventCode	EventCode according to Table D.2	Unsigned16	0 to 0xFFFF

E.17 VoidBlock

This ArgBlock is used in SMI services to indicate read requests within the argument. Table E.17 shows the coding of the ArgBlockType "VoidBlock".

Table E.17 – VoidBlock

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0xFFF0

E.18 JobError

This ArgBlock is used in SMI services to indicate negative acknowledgments "Result (-)" together with an ErrorType according to Table C.3. Table E.18 shows the coding of the ArgBlockType "JobError".

Table E.18 – JobError

Octet Offset	Element name	Definition	Data type	Values
0	ArgBlockID	Unique ID	Unsigned16	0xFFFF
2	ExpArgBlockID	Expected ArgBlockID of the service result	Unsigned16	0x0001 to 0xFFFF
4	ErrorCode	SMI service related ErrorType or propagated Device/Master error (upper value)	Unsigned8	Table C.3

Octet Offset	Element name	Definition	Data type	Values
5	AdditionalCode	SMI service related ErrorType or propagated Device/Master error (lower value)	Unsigned8	

Annex F
(normative)

Data types

F.1 General

This annex specifies basic and composite data types. Examples demonstrate the structures and the transmission aspects of data types for singular use or in a packed manner.

NOTE More examples are available in [6].

F.2 Basic data types

F.2.1 General

The coding of basic data types is shown only for singular use, which is characterized by

- Process Data consisting of one basic data type
- Parameter consisting of one basic data type
- Subindex (>0) access on individual data items of parameters of composite data types (arrays, records)

F.2.2 BooleanT

A BooleanT is representing a data type that can have only two different values i.e. TRUE and FALSE. The data type is specified in Table F.1. For singular use the coding is shown in Table F.2. A sender shall always use 0xFF for 'TRUE' or 0x00 for 'FALSE'. Since some upperlevel software tools are not used to this restricted use of Booleans, a receiver can interpret the range from 0x01 through 0xFE for 'TRUE' or reject with an error message. The packed form is demonstrated in Table F.22 and Figure F.9.

Table F.1 – BooleanT

Data type name	Value range	Resolution	Length
BooleanT	TRUE / FALSE	-	1 bit or 1 octet

Table F.2 – BooleanT coding

Bit	7	6	5	4	3	2	1	0	Values
TRUE	1	1	1	1	1	1	1	1	0xFF
FALSE	0	0	0	0	0	0	0	0	0x00

F.2.3 UIntegerT

A UIntegerT is representing an unsigned number depicted by 2 up to 64 bits ("enumerated"). The number is accommodated and right-aligned within the following permitted octet containers: 1, 2, 4, or 8. High order padding bits are filled with "0". Coding examples are shown in Figure F.1 and Figure F.2.

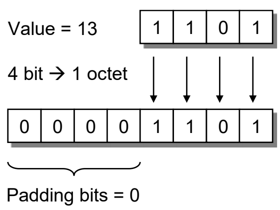


Figure F.1 – Coding example of small UIntegerT

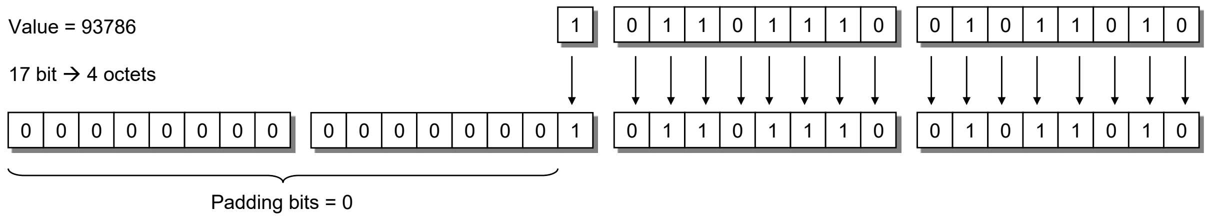


Figure F.2 – Coding example of large UIntegerT

The data type UIntegerT is specified in Table F.3 for singular use.

Table F.3 – UIntegerT

Data type name	Value range	Resolution	Length
UIntegerT	0 … $2^{\text{bitlength}}$ - 1	1	1 octet, or 2 octets, or 4 octets, or 8 octets
NOTE 1 High order padding bits are filled with "0". NOTE 2 Most significant octet (MSO) sent first.			

F.2.4 IntegerT

An IntegerT is representing a signed number depicted by 2 up to 64 bits. The number is accommodated within the following permitted octet containers: 1, 2, 4, or 8 and right-aligned and extended correctly signed to the chosen number of bits. The data type is specified in Table F.4 for singular use. SN represents the sign with "0" for all positive numbers and zero, and "1" for all negative numbers. Padding bits are filled with the content of the sign bit (SN).

Table F.4 – IntegerT

Data type name	Value range	Resolution	Length
IntegerT	$-2^{\text{bitlength -1}}$ … $2^{\text{bitlength -1}}$ - 1	1	1 octet, or 2 octets, or 4 octets, or 8 octets
NOTE 1 High order padding bits are filled with the value of the sign bit (SN). NOTE 2 Most significant octet (MSO) sent first (lowest respective octet number in Table F.5).			

The 4 coding possibilities in containers are listed in Table F.5 through Table F.8.

Table F.5 – IntegerT coding (8 octets)

Bit	7	6	5	4	3	2	1	0	Container
Octet 1	SN	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}	8 octets
Octet 2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}	
Octet 3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}	
Octet 4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}	
Octet 5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	
Octet 6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

Table F.6 – IntegerT coding (4 octets)

Bit	7	6	5	4	3	2	1	0	Container
Octet 1	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	4 octets
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

Table F.7 – IntegerT coding (2 octets)

Bit	7	6	5	4	3	2	1	0	Container
Octet 1	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2 octets
Octet 2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

Table F.8 – IntegerT coding (1 octet)

Bit	7	6	5	4	3	2	1	0	Container
Octet 1	SN	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	1 octet

Coding examples within containers are shown in Figure F.3

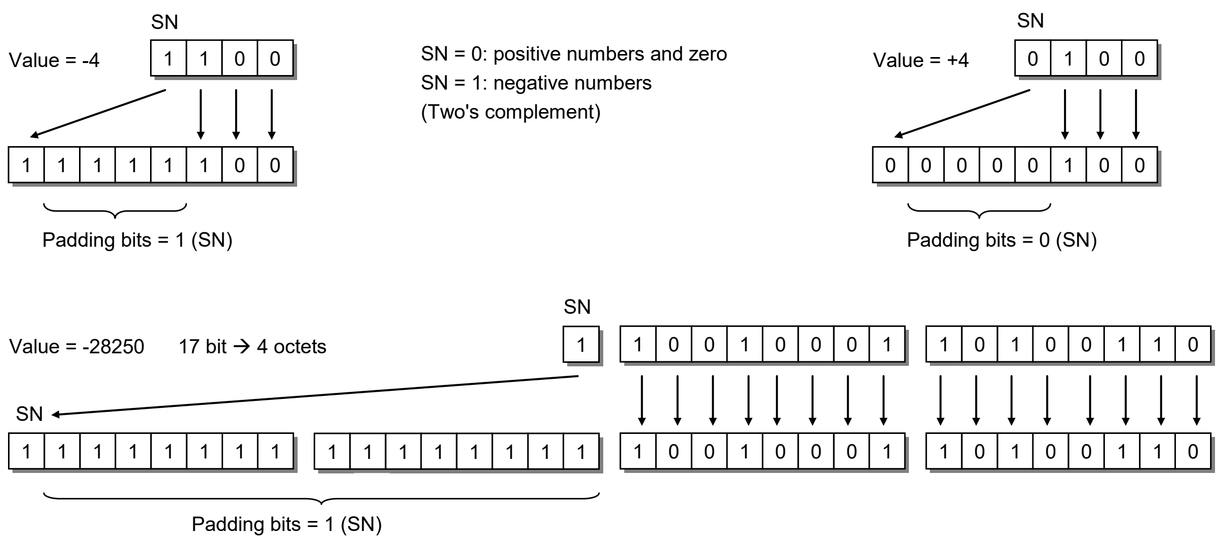


Figure F.3 – Coding examples of IntegerT

F.2.5 Float32T

A Float32T is representing a number specified by IEEE Std 754-1985 as single precision (32 bit). Table F.9 gives the definition and Table F.10 the coding. SN represents the sign with "0" for all positive numbers and zero, and "1" for all negative numbers.

Table F.9 – Float32T

Data type name	Value range	Resolution	Length
Float32T	See IEEE Std 754-1985	See IEEE Std 754-1985	4 octets

Table F.10 – Coding of Float32T

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

In order to realize negative exponent values a special exponent encoding mechanism is set in place as follows:

The Float32T exponent (E) is encoded using an offset binary representation, with the zero offset being 127; also known as exponent bias in IEEE Std 754-1985.

E_{min} = 0x01 - 0x7F = -126

E_{max} = 0xFE - 0x7F = 127

Exponent bias = 0x7F = 127

Thus, as defined by the offset binary representation, in order to get the true exponent the offset of 127 shall be subtracted from the stored exponent.

F.2.6 StringT

A StringT is representing an ordered sequence of symbols (characters) with a variable or fixed length of octets (maximum of 232 octets) coded in US-ASCII (7 bit) or UTF-8. UTF-8 uses one octet for all ASCII characters and up to 4 octets for other characters. 0x00 is not permitted as a character. Table F.11 gives the definition.

Table F.11 – StringT

Data type name	Encoding	Standards	Length [a]
StringT	US-ASCII	see ISO/IEC 646	Any length of character string with a maximum of 232 octets
	UTF-8 [b]	see ISO/IEC 10646	
NOTE a Length can be obtained from a Device's IODD via the attribute 'fixedLength'.			
NOTE b In order to ensure proper handling of client applications it is recommended not to use US-ASCII or UTF-8 codes from 0x00 to 0x1F and 0xFF.			

An instance of StringT can be shorter than defined by the IODD attribute 'fixedLength'. 0x00 shall be used for the padding of unused octets.

A condensed form can be used for optimization, where the character string is transmitted in its actual length and the padding octets are omitted. The receiver can deduce the original length from the length of the ISDU or by searching the first NULL (0x00) character (see A.5.2 and A.5.3). This condensed form can be used in case of singular access (see Figure F.4).

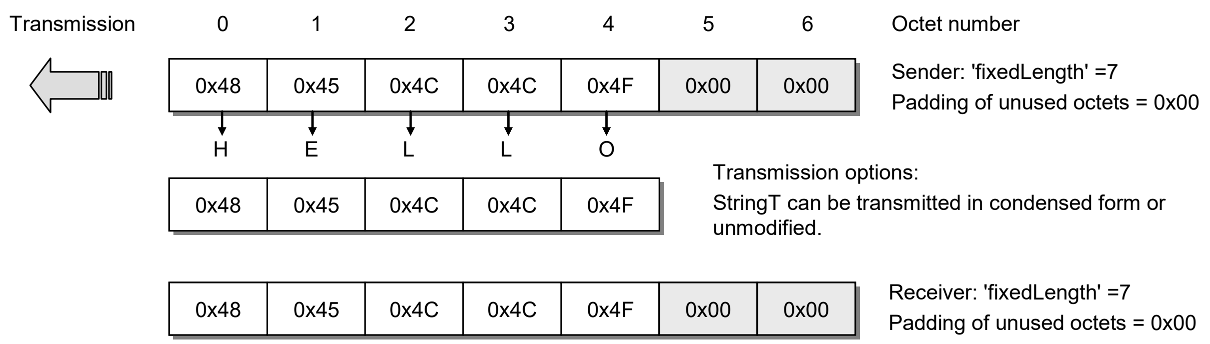


Figure F.4 – Singular access of StringT

F.2.7 OctetStringT

An OctetStringT is representing an ordered sequence of octets with a fixed length (maximum of 232 octets). Table F.12 gives the definition and Figure F.5 a coding example for a fixed length of 7.

Table F.12 – OctetStringT

Data type name	Value range	Standards	Length
OctetStringT	0x00 … 0xFF per octet	-	Fixed length with a maximum of 232 octets
NOTE The length may be obtained from a Device's IODD via the attribute 'fixedLength'.			

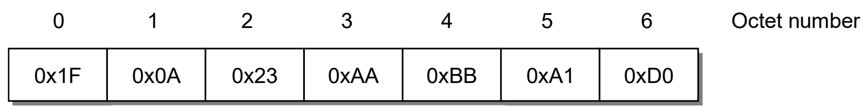


Figure F.5 – Coding example of OctetStringT

F.2.8 TimeT

A TimeT is based on the RFC 1305 standard and composed of two unsigned values that express the network time related to a particular date. Its semantic has changed from RFC 1305 according to Figure F.6. Table F.13 gives the definition and Table F.14 the coding of TimeT.

The first element is a 32-bit unsigned integer data type that provides the network time in seconds since 1900-01-01 0.00,00(UTC) or since 2036-02-07 6.28,16(UTC) for time values less than 0x9DFF4400, which represents the 1984-01-01 0:00,00(UTC). The second element is a 32-bit unsigned integer data type that provides the fractional portion of seconds in $1/2^{32}$ s. Rollovers after 136 years are not automatically detectable and shall be maintained by the application.

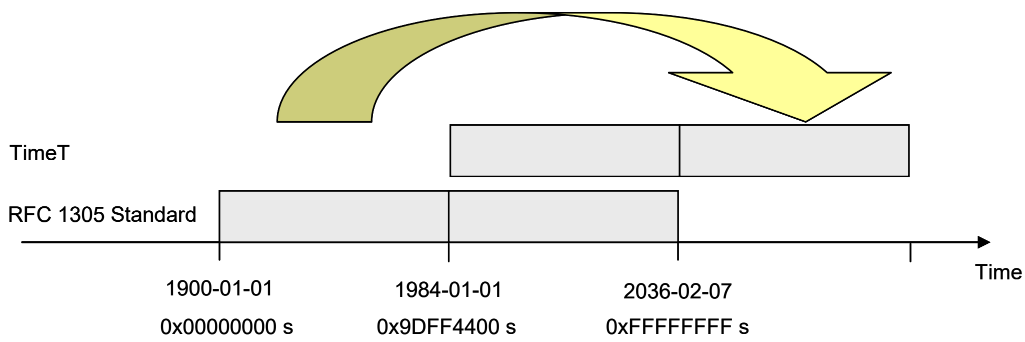


Figure F.6 – Definition of TimeT

Table F.13 – TimeT

Data type name	Value range	Resolution	Length
TimeT	Octet 1 to 4 (see Table F.14): $0 \leq i \leq (2^{32}-1)$	s (Seconds)	8 Octets (32-bit unsigned integer + 32 bit unsigned integer)
	Octet 5 to 8 (see Table F.14): $0 \leq i \leq (2^{32}-1)$	$(1/2^{32})$ s	
NOTE 32-bit unsigned integer are normal computer science data types			

Table F.14 – Coding of TimeT

Bit	7	6	5	4	3	2	1	0	Definitions
Octet 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Seconds since 1900-01-01 0.00,00 or since 2036-02-07 6.28,16 when time value less than 0x9DFF4400.00000000
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octet 5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Fractional part of seconds. One unit is $1/(2^{32})$ s
Octet 6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
	MSB							LSB	MSB = Most significant bit LSB = Least significant bit

F.2.9 TimeSpanT

A TimeSpanT is a 64-bit integer value i.e. a two's complement binary number with a length of eight octets, providing the network time difference in fractional portion of seconds in $1/2^{32}$ seconds.

Table F.15 gives the definition and Table F.16 the coding of TimeSpanT.

Table F.15 – TimeSpanT

Data type name	Value range	Resolution	Length
TimeSpanT	Octet 1 to 4 (see Table F.16): $-2^{31} \leq i \leq (2^{31}-1)$	s (Seconds)	8 octets (32-bit integer + 32 bit unsigned integer)
	Octet 5 to 8 (see Table F.16): $0 \leq i \leq (2^{32}-1)$	$(1/2^{32})$ s	
NOTE 32-bit integer and unsigned integer are normal computer science data type			

Table F.16 – Coding of TimeSpanT

Bit	7	6	5	4	3	2	1	0	Definitions
Octet 1	Sign	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Seconds as 32-bit integer.
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octet 5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Fractional part of seconds. One unit is $1/(2^{32})$ s.
Octet 6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
	MSB							LSB	MSB = Most significant bit LSB = Least significant bit

F.3 Composite data types

F.3.1 General

Composite data types are combinations of basic data types only. A composite data type consists of several basic data types packed within a sequence of octets. Unused bit space shall be padded with "0".

F.3.2 ArrayT

An ArrayT addressed by an Index is a data structure with data items of the same data type. The individual data items are addressable by the Subindex. Subindex 0 addresses the whole array within the Index space. The structuring rules for arrays are given in Table F.17.

Table F.17 – Structuring rules for ArrayT

Rule number	Rule specification
1	The Subindex data items are packed in a row without gaps describing an octet sequence
2	The highest Subindex data item n starts right aligned within the octet sequence
3	UIntegerT and IntegerT with a length of ≥ 58 bit and < 64 bit are not permitted

Table F.18 and Figure F.7 give an example for the access of an array. Its content is a set of parameters of the same basic data type.

Table F.18 – Example for the access of an ArrayT

Index	Subindex	Offset	Data items	Data Type
66	1	12	0x2	IntegerT, 'bitLength' = 3
	2	9	0x6	
	3	6	0x4	
	4	3	0x7	
	5	0	0x5	

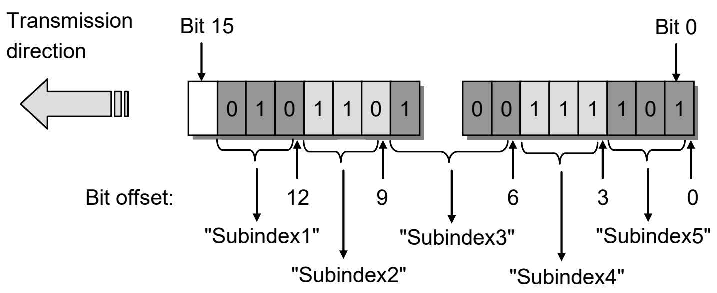


Figure F.7 – Example of an ArrayT data structure

F.3.3 RecordT

A record addressed by an Index is a data structure with data items of different data types. The Subindex allows addressing individual data items within the record on certain bit positions.

NOTE Bit positions within a RecordT may be obtained from the IODD of the particular Device.

The structuring rules for records are given in Table F.19.

Table F.19 – Structuring rules for RecordT

Rule number	Rule specification
1	The Subindices within the IODD shall be listed in ascending order from 1 to *n* describing an octet sequence. Gaps within the list of Subindices are allowed
2	Bit offsets shall always be indicated within this octet sequence (may show no strict order in the IODD)
3	The bit offset starts with the last octet within the sequence; this octet starts with offset 0 for the least significant bit and offset 7 for the most significant bit
4	The following data types shall always be aligned on octet boundaries: Float32T, StringT, OctetStringT, TimeT, and TimeSpanT
5	UIntegerT and IntegerT with a length of ≥ 58 bit shall always be aligned on one side of an octet boundary
6	It is highly recommended for UIntegerT and IntegerT with a length of ≥ 8 bit to align always on one side of an octet boundary
7	It is highly recommended for UIntegerT and IntegerT with a length of < 8 bit not to cross octet boundaries
8	A bit position shall not be used by more than one record item

Table F.20 gives an example 1 for the access of a RecordT. It consists of varied parameters named "Status", "Text", and "Value".

Table F.20 – Example 1 for the access of a RecordT

Index	Subindex	Offset	Data items							Data Type	Name
47	1	88	0x23	0x45						UIntegerT, 'bitLength' = 16	Status
	2	32	H	E	L	L	O	0x00	0x00	StringT, 'fixedLength' = 7	Text
	3	0	0x56	0x12	0x22	0x34				UIntegerT, 'bitLength' = 32	Value
NOTE 'bitLength' and 'fixedLength' are defined in the IODD of the particular Device.											

Table F.21 gives an example 2 for the access of a RecordT. It consists of varied parameters named "Level", "Min", and "Max". Figure F.8 shows the corresponding data structure.

Table F.21 – Example 2 for the access of a RecordT

Index	Subindex	Offset	Data items		Data Type	Name
46	1	2	0x32	0xF1	UIntegerT, 'bitLength' = 14	Level
	2	1	FALSE		BooleanT	Min
	3	0	TRUE		BooleanT	Max
NOTE 'bitLength' is defined in the IODD of the particular Device.						

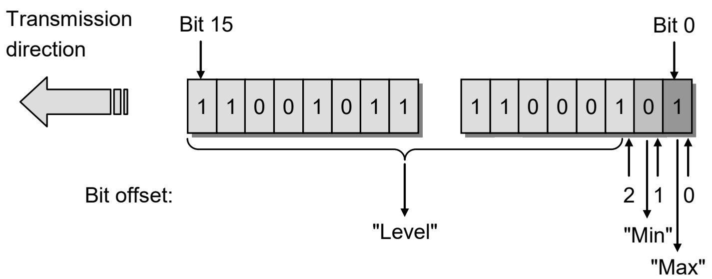


Figure F.8 – Example 2 of a RecordT structure

Table F.22 gives an example 3 for the access of a RecordT. It consists of varied parameters named "Control" through "Enable". Figure F.9 demonstrates the corresponding RecordT structure of example 3 with the bit offsets.

Table F.22 – Example 3 for the access of a RecordT

Index	Subindex	Offset	Data items		Data Type	Name
45	1	32	TRUE		BooleanT	NewBit
	2	33	FALSE		BooleanT	DR4
	3	34	FALSE		BooleanT	CR3
	4	35	TRUE		BooleanT	CR2
	5	38	TRUE		BooleanT	Control
	6	16	0xF8	0x23	OctetStringT, 'fixedLength' = 2	Setpoint
	7	8	0x41		StringT, 'fixedLength' = 1	Unit
	8	0	0xC3		OctetStringT, 'fixedLength' = 1	Enable
NOTE 'fixedLength' is defined in the IODD of the particular Device						

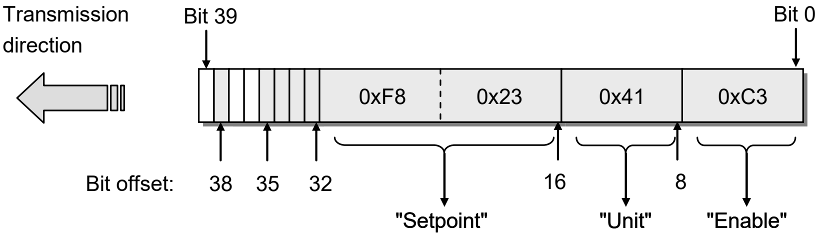


Figure F.9 – Example 3 of a RecordT structure

Figure F.10 shows a selective write request of a variable within the RecordT of example 3 and a write request of the complete RecordT (see A.5.7).

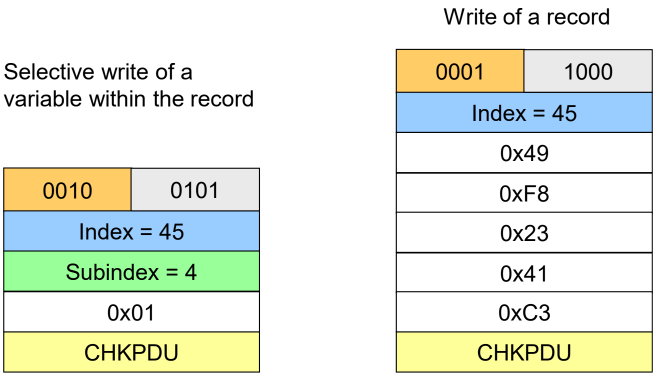


Figure F.10 – Write requests for example 3

Annex G
(normative)

Structure of the Data Storage data object

Table G.1 gives the structure of a Data Storage (DS) data object within the Master (see 11.4.2).

Table G.1 – Structure of the stored DS data object

Part	Parameter name	Definition	Data type
Object 1	ISDU_Index	ISDU Index (0 to 0xFFFF)	Unsigned16
	ISDU_Subindex	ISDU Index (0 to 0xFF)	Unsigned8
	ISDU_Length	Length of the subsequent record	Unsigned8
	ISDU_Data	Record of length ISDU_Length	Record
Object 2	ISDU_Index	ISDU Index (0 to 0xFFFF)	Unsigned16
	ISDU_Subindex	ISDU Index (0 to 0xFF)	Unsigned8
	ISDU_Length	Length of the subsequent record	Unsigned8
	ISDU_Data	Record of length ISDU_Length	Record

Object *n*	ISDU_Index	ISDU Index (0 to 0xFFFF)	Unsigned16
	ISDU_Subindex	ISDU Index (0 to 0xFF)	Unsigned8
	ISDU_Length	Length of the subsequent record	Unsigned8
	ISDU _Data	Record of length ISDU_Length	Record

The Device shall calculate the required memory size by summarizing the objects 1 to *n* (see Table B.10, Subindex 3).

The Master shall store locally in non-volatile memory the header information specified in Table G.2. See Table B.10.

Table G.2 – Associated header information for stored DS data objects

Part	Parameter name	Definition	Data type
Header	Parameter Checksum	32-bit CRC signature or revision counter (see 10.4.8)	Unsigned32
	VendorID	See B.1.8	Unsigned16
	DeviceID	See B.1.9	Unsigned32
	FunctionID	See B.1.10	Unsigned16

In case of empty Data Storage data object, the header shall be set to "0" and the ArgBlockLength shall be set to 12.

Annex H
(normative)

Master and Device conformity

H.1 Electromagnetic compatibility requirements (EMC)

H.1.1 General

The EMC requirements of this specification are only relevant for the SDCI interface part of a particular Master or Device. The technology functions of a Device and its relevant EMC requirements are not in the scope of this specification. For this purpose, the Device specific product standards shall apply. For Master usually the EMC requirements for peripherals are specified in IEC 61131-2 or IEC 61000-6-2.

To ensure proper operating conditions of the SDCI interface, the test configurations specified in section H.1.6 (Master) or H.1.7 (Device) shall be maintained during all the EMC tests. The tests required in the product standard of equipment under test (EUT) can alternatively be performed in SIO mode.

H.1.2 Operating conditions

It is highly recommended to evaluate the SDCI during the startup phase with the cycle times given in Table H.1. In most cases, this leads to the minimal time requirements for the performance of these tests. Alternatively, the SDCI may be evaluated during normal operation of the Device, provided that the required number of M-sequences specified in Table H.1 took place during each test.

In case a test requires longer M-sequences than an M-sequence group specified in Table H.1, the error criteria shall be applied to every M-sequence group.

In case of Class B devices it is recommended to perform the EMC test under maximum ripple and load switching on Power 2.

H.1.3 Performance criteria

a) Performance criterion A

The SDCI operating at an average cycle time as specified in Table H.1 shall not show more than six detected M-sequence errors within the number of M-sequences given in Table H.1. Multiple kinds of errors within one M-sequence shall be counted as one error. No interruption of communication is permitted.

Table H.1 – EMC test conditions for SDCI

Transmission rate	Master		Device		Maximum of M-sequence errors
	t_{CYC}	Number of M-sequences of TYPE_2_5 (read) (6 octets)	t_{CYC}	Number of M-sequences of TYPE_0 (read) (4 octets)	
4,8 kbit/s	18,0 ms	300 (6 000)	100 T_{BIT} (20,84 ms)	350 (7 000)	6
38,4 kbit/s	2,3 ms	450 (9 000)	100 T_{BIT} (2,61 ms)	500 (10 000)	6
230,4 kbit/s	0,4 ms	700 (14 000)	100 T_{BIT} (0,44 ms)	800 (16 000)	6

NOTE1 The numbers of M-sequences are calculated according to the algorithm in I.2 and rounded up. The larger number of M-sequences (in brackets) are required if a certain test (for example fast transients/burst) applies interferences only with a burst/cycle ratio (see Table H.2)

NOTE2 "Number of M-sequences" is defined as a group for the performance criteria for which the maximum number of detected errors is valid.

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.

H.1.4 Required immunity tests

Table H.2 specifies the EMC tests to be performed.

Table H.2 – EMC test levels

Phenomena	Test Level	Performance Criterion	Constraints
Electrostatic discharges (ESD) IEC 61000-4-2	Air discharge: ± 8 kV Contact discharge: ± 4 kV	B	See H.1.4, a)
Radiofrequency electromagnetic field. Amplitude modulated IEC 61000-4-3	80 MHz – 1 000 MHz 10 V/m 1 400 MHz – 2 000 MHz 3 V/m 2 000 MHz – 2 700 MHz 3 V/m	A	See H.1.4, a), H.1.4, b), H.1.4, e).
Fast transients (Burst) IEC 61000-4-4	± 1 kV	A	5 kHz or 100 kHz.The number of M-sequences in Table H.1 shall be increased by a factor of 20 due to the burst/cycle ratio 15 ms/300 ms. See H.1.4, c)
	± 2 kV	B	
Surge IEC 61000-4-5	Not required for an SDCI link (SDCI link is limited to 20 m)		-
Radio-frequency common mode IEC 61000-4-6	0,15 MHz – 80 MHz 10 VEMF	A	See H.1.4, b) and H.1.4, d)
Voltage dips and interruptions IEC 61000-4-11	Not required for an SDCI link		

The following requirements also apply as specified in Table H.2.

a) As this phenomenon influences the entire device under test, an existing device specific product standard shall take precedence over the test levels specified here.

b) The test shall be performed with a step size of 1 % and a dwell of 1 s. If a single M-sequence error occurs at a certain frequency, that frequency shall be tested until the number of M-sequences according to Table H.1 has been transmitted or until 6 M-sequence errors occurred.

c) Depending on the transmission rate the test time varies. The test time shall be at least one minute (with the transmitted M-sequences and the permitted errors increased accordingly).

d) This phenomenon is expected to influence most probably the EUTs internal analog signal processing and only with a very small probability the functionality of the SDCI communication. Therefore, an existing device specific product standard shall take precedence over the test levels specified here.

e) Measurement shall be performed at least for three orthogonal orientations of the Device with respect to the direction of the electromagnetic wave propagation.

H.1.5 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.

H.1.6 Test configurations for Master

H.1.6.1 General rules

The following rules apply for the test of Masters:

- In the following test setup diagrams only the SDCI and the power supply cables are shown. All other cables shall be treated as required by the relevant product standard.
- Grounding of power supply, Master, and Devices shall be according to the relevant product standard or manual.
- Where not otherwise stated, the SDCI 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 reference ground.
- Where applicable, the auxiliary Devices shall be placed 10 cm above RefGND.
- A typical test configuration consists of the Master and two Devices, except for the RF common mode test, where only one Device shall be used.
- Each port shall fulfill the EMC requirements.

H.1.6.2 Electrostatic discharges

Figure H.1 shows the test setup for electrostatic discharge according to IEC 61000-4-2.

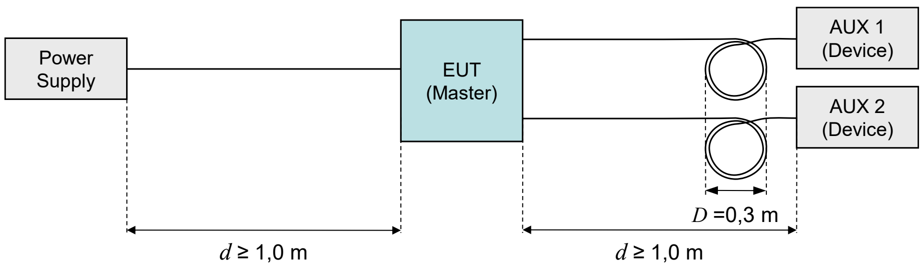


Figure H.1 – Test setup for electrostatic discharge (Master)

H.1.6.3 Radio-frequency electromagnetic field

Figure H.2 shows the test setup for radio-frequency electromagnetic field according to IEC 61000-4-3.

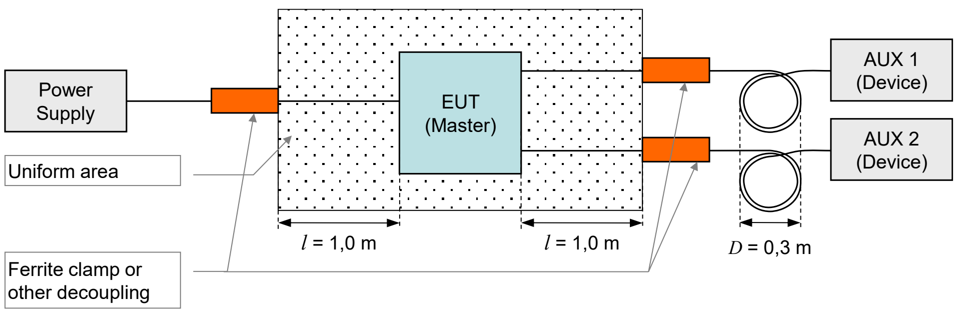


Figure H.2 – Test setup for RF electromagnetic field (Master)

H.1.6.4 Fast transients (burst)

Figure H.3 shows the test setup for fast transients according to IEC 61000-4-4. No coupling into SDCI line to AUX 2 is required.

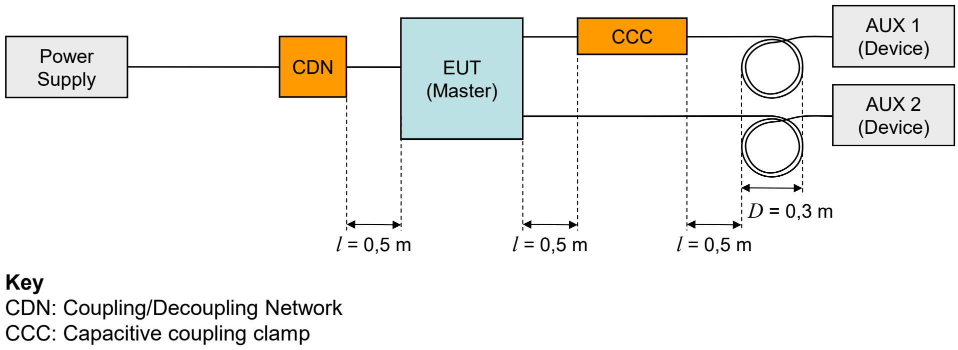


Figure H.3 – Test setup for fast transients (Master)

H.1.6.5 Radio-frequency common mode

Figure H.4 shows the test setup for radio-frequency common mode according to IEC 61000-4-6.

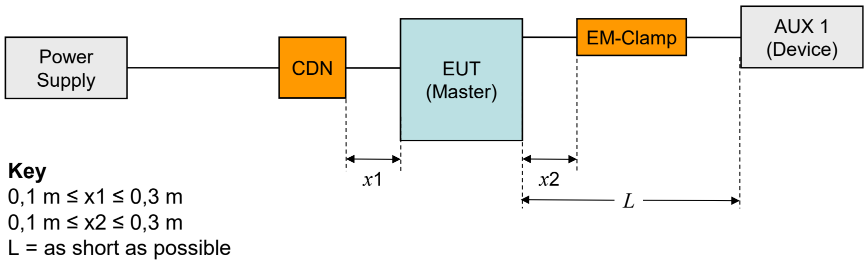


Figure H.4 – Test setup for RF common mode (Master)

H.1.7 Test configurations for Devices

H.1.7.1 General rules

For the test of Devices, the following rules apply:

- In the following test setup diagrams only the SDCI 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 user manual.
- Where not otherwise stated, the SDCI 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.
- Test with Device AUX 2 is optional

H.1.7.2 Electrostatic discharges

Figure H.5 shows the test setup for electrostatic discharge according to IEC 61000-4-2.

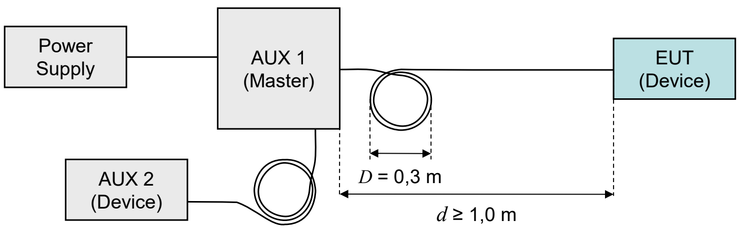


Figure H.5 – Test setup for electrostatic discharges (Device)

H.1.7.3 Radio-frequency electromagnetic field

Figure H.6 shows the test setup for radio-frequency electromagnetic field according to IEC 61000-4-3.

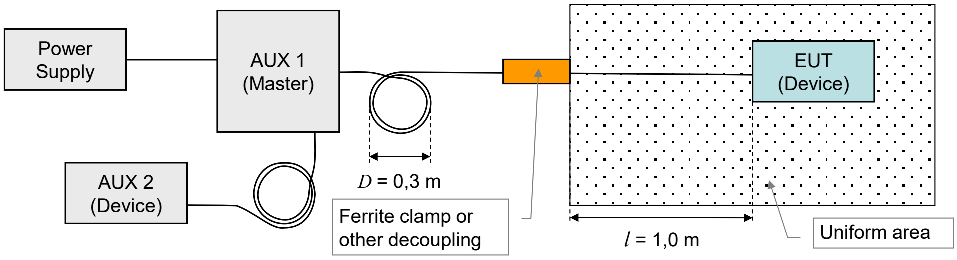
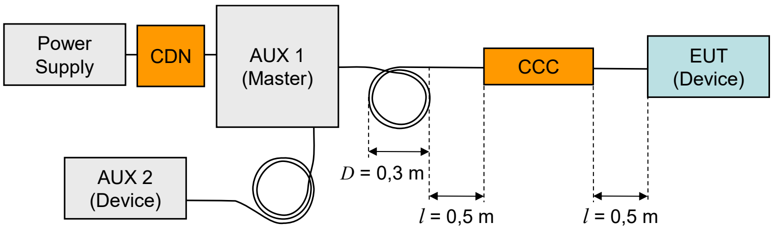


Figure H.6 – Test setup for RF electromagnetic field (Device)

H.1.7.4 Fast transients (burst)

Figure H.7 shows the test setup for fast transients according to IEC 61000-4-4.

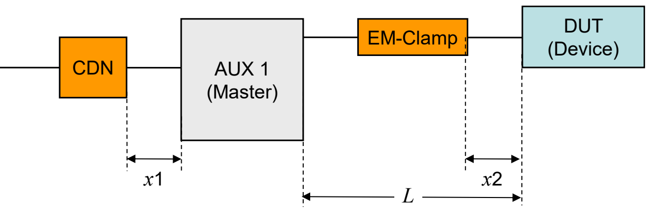


Key
CDN: Coupling/Decoupling Network, here only used for decoupling
CCC: Capacitive coupling clamp

Figure H.7 – Test setup for fast transients (Device)

H.1.7.5 Radio-frequency common mode

Figure H.8 shows the test setup for radio-frequency common mode according to IEC 61000-4-6.



Key
0,1 m ≤ x1 ≤ 0,3 m
0,1 m ≤ x2 ≤ 0,3 m
L = as short as possible

Figure H.8 – Test setup for RF common mode (Device)

H.2 Test strategies for conformity

H.2.1 Test of a Device

The Master AUX 1 (see Figure H.5 to Figure H.8) shall continuously send an M-sequence TYPE_0 (read Direct Parameter page 2) message at the cycle time specified in Table H.1 and count the missing and the erroneous Device responses. Both numbers shall be added and indicated.

NOTE Detailed instructions for the Device tests are specified in [9].

H.2.2 Test of a Master

The Device AUX 1 (see Figure H.1 to Figure H.4) shall use M-sequence TYPE_2_5. Its input Process Data shall be generated by an 8 bit random or pseudo random generator. The Master shall copy the input Process Data of any received Device message to the output Process Data of the next Master message to be sent. The cycle time should be according to Table H.1. If not possible, the number of M-sequences for the test shall be calculated according to the algorithm in I.2 and rounded up. Used cycle time and number of M-sequences shall be documented in test records. The Device AUX 1 shall compare the output Process Data with the previously sent input Process Data and count the number of deviations. The Device shall also count the number of missing (not received within the expected cycle time) or received perturbed Master messages. All numbers shall be added and indicated.

NOTE 1 A deviation of sent and received Process Data indicates to the AUX1 that the EUT (Master) did not receive the Device message.

NOTE 2 Detailed instructions for the Master tests are specified in [9].

Annex I
(informative)

Residual error probabilities

I.1 Residual error probability of the SDCI data integrity mechanism

Figure I.1 shows the residual error probability (REP) of the SDCI data integrity mechanism consisting of the checksum data integrity procedure ("XOR6") as specified in A.1.6 and the UART parity. The diagram refers to IEC 60870-5-1 with its data integrity class I2 for a minimum Hamming distance of 4 (red dotted line).

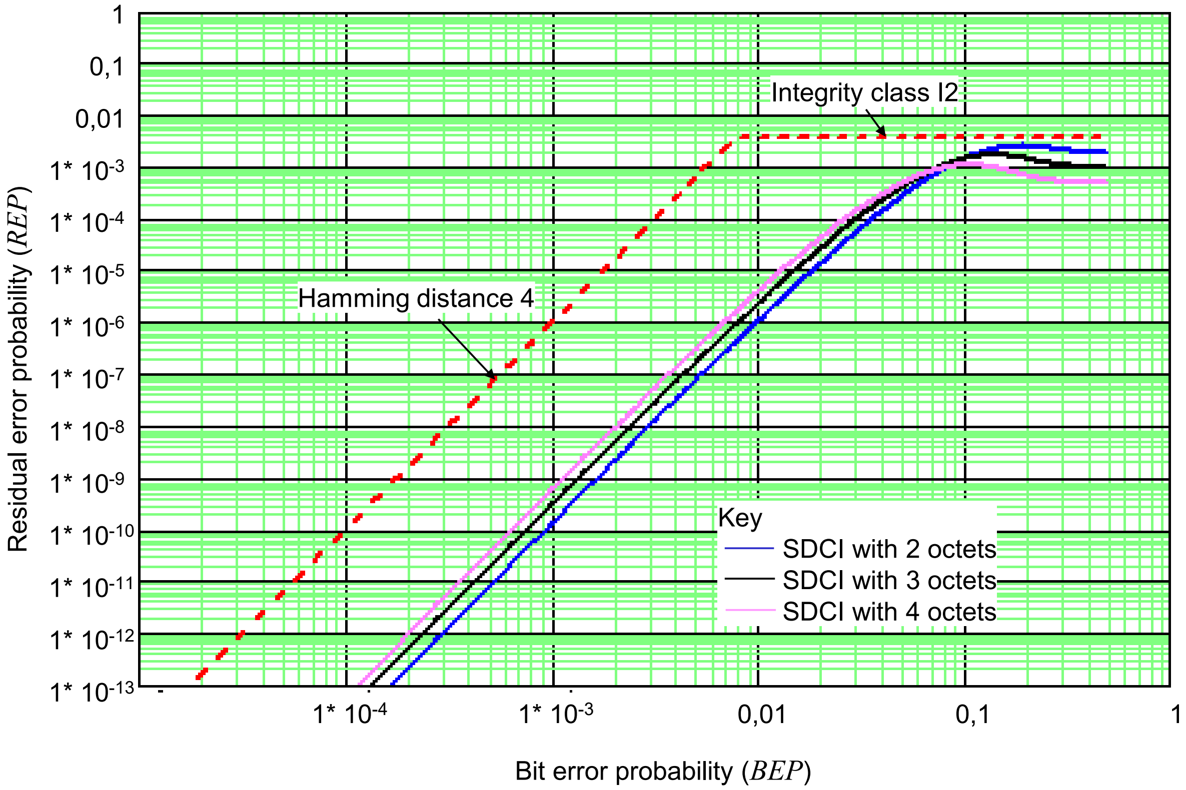


Figure I.1 – Residual error probability for the SDCI data integrity mechanism

The blue line shows the residual error curve for a data length of 2 octets. The black curve shows the residual error curve for a data length of 3 octets. The purple curve shows the residual error curve for a data length of 4 octets.

I.2 Derivation of EMC test conditions

The performance criterion A in H.1.3 is derived from requirements specified in IEC 61158-2 in respect to interference susceptibility and error rates (citation; "*frames*" translates into "messages" within this standard):

- *Only 1 undetected erroneous frame in 20 years at 1 600 frames/s*
- *The ratio of undetected to detected frames shall not exceed* 10^{-6}
- *EMC tests shall not show more than 6 erroneous frames within 100 000 frames*

With SDCI, the first requirement transforms into the Equation (I.1). This equation allows determining a value of BEP. The equation can be resolved in a numerical way.

$$F20 \times R(BEP) \leq 1 \qquad \text{(I.1)}$$

The Terms in equation (I.1) are:

$F20$ = Number of messages in 20 years

$R(BEP)$ = Residual error probability of the checksum and parity mechanism (Figure I.1)

BEP = Bit error probability from Figure I.1

The objective of the EMC test is to prove that the BEP of the SDCI communication meets the value determined in the first step. The maximum number of detected perturbed messages is chosen to be 6 here for practical reasons. The number of required SDCI test messages can be determined with the help of equation (I.2) and the value of BEP determined in the first step.

$$NoTF \geq \frac{1}{BEP} \times \frac{1}{BitPerF} \times NopErr \qquad \text{(I.2)}$$

The Terms in equation (I.2) are:

$NoTF$ = Number of test messages

$BitPerF$ = Number of bits per message

$NopErr$ = Maximum number of detected perturbed messages = 6

Equation (I.2) is only valid under the assumption that messages with 1 bit error are more frequent than messages with more bit errors. An M-sequence consists of two messages. Therefore, the calculated number of test messages has to be divided by 2 to provide the numbers of M-sequences for Table H.1.

Annex J
(informative)

Example sequence of an ISDU transmission

Figure J.1 demonstrates an example for the transmission of ISDUs using an AL_Read service with a 16-bit Index and Subindex for 19 octets of user data with mapping to an M-sequence TYPE_2_5 for sensors and with interruption in case of an Event transmission.

Master						Device			
		FC	CKT	PD	OD	OD	PD	CKS	
comment (state, action) (see in Table 46)	cycle nr	R W 1bit, Com Chan. 2bit, Flow CTRL 5bit	Frame Typ 2bit, CHK 6bit	Process Data 8bit	OnReq Data Master 8bit	OnReq Data Device 8bit	Process Data	E, PD, CHK	comment (state, action)
Idle_1	0	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	OnReq idle
ISDURequest_2, transmission,	1	0111 0000	10 xxxxxx	xxxxxxxx	1011 0101		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	2	0110 0001	10 xxxxxx	xxxxxxxx	Index(hi)		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	3	0110 0010	10 xxxxxx	xxxxxxxx	Index(lo)		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	4	0110 0011	10 xxxxxx	xxxxxxxx	Subindex		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	5	0110 0100	10 xxxxxx	xxxxxxxx	CHKPDU		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDUWait_3, start ISDU Timer	6	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUWait_3, inc. ISDU timer	7	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUWait_3, inc. ISDU timer	8	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUWait_3, inc. ISDU timer	9	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUWait_3, inc. ISDU timer	10	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUResponse_4, reception Stop ISDU Timer	11	1111 0000	10 xxxxxx	xxxxxxxx		1101 0001	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	12	1110 0001	10 xxxxxx	xxxxxxxx		0001 0011	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	13	1110 0010	10 xxxxxx	xxxxxxxx		Data 1	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	14	1110 0011	10 xxxxxx	xxxxxxxx		Data 2	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	15	1110 0100	10 xxxxxx	xxxxxxxx		Data 3	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	16	1110 0101	10 xxxxxx	xxxxxxxx		Data 4	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	17	1110 0110	10 xxxxxx	xxxxxxxx		Data 5	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	18	1110 0111	10 xxxxxx	xxxxxxxx		Data 6	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	19	1110 1000	10 xxxxxx	xxxxxxxx		Data 7	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, no response, retry in next cycle	20	1110 1001	10 Err	xxxxxxxx				xxxxxx	ISDUResponse_4, korrupted CHK, don' t send resp.
ISDUResponse_4, no response, retry in next cycle	21	1110 1001	10 Err	xxxxxxxx				xxxxxx	ISDUResponse_4, corrupted CHK, don' t send resp.
ISDUResponse_4, reception	22	1110 1001	10 xxxxxx	xxxxxxxx		Data 8	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	34	1110 1010	10 xxxxxx	xxxxxxxx		Data 9	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception, start eventhandler	35	1110 1011	10 xxxxxx	xxxxxxxx		Data 10	xxxxxxxx	1 0 xxxxxx	ISDUResponse_4, transmission, freeze event
Read_Event_2, reception	36	1100 0000	10 xxxxxx	xxxxxxxx		Diag State with detail	xxxxxxxx	1 0 xxxxxx	Read_Event_2, transmission
Read_Event_2, reception	37	110x xxxx	10 xxxxxx	xxxxxxxx		Event qualifier	xxxxxxxx	1 0 xxxxxx	Read_Event_2, transmission
Command handler_2, transmission set PDOutdata state to invalid	38	0010 0000	10 xxxxxx	xxxxxxxx	1001 1001		xxxxxxxx	1 0 xxxxxx	ComandHandler_2, reception, set PDOutdata state to invalid
Read_Event_2, reception	39	110x xxxx	10 xxxxxx	xxxxxxxx		ErrorCode msb	xxxxxxxx	1 0 xxxxxx	Read_Event_2, transmission
Read_Event_2, reception	40	110x xxxx	10 xxxxxx	xxxxxxxx		ErrorCode lsb	xxxxxxxx	1 0 xxxxxx	Read_Event_2, transmission
EventConfirmation_4, transmission, event handler idle	41	0100 0000	10 xxxxxx	xxxxxxxx	xxxxxxxx		xxxxxxxx	0 0 xxxxxx	EventConfirmation, reception
ISDUResponse_4, reception	42	1110 1100	10 xxxxxx	xxxxxxxx		Data 11	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	43	1110 1101	10 xxxxxx	xxxxxxxx		Data 12	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	44	1110 1110	10 xxxxxx	xxxxxxxx		Data 13	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	45	1110 1111	10 xxxxxx	xxxxxxxx		Data 14	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	46	1110 0000	10 xxxxxx	xxxxxxxx		Data 15	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	47	1110 0001	10 xxxxxx	xxxxxxxx		Data 16	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	48	1110 0010	10 xxxxxx	xxxxxxxx		CHKPDU	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
Idle_1	49	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1
Idle_1	50	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1
Idle_1	51	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1
Idle_1	52	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1
Write Parameter, transmission	53	0011 0000	10 xxxxxx	xxxxxxxx	xxxxxxxx		xxxxxxxx	0 0 xxxxxx	Write Parameter, reception
Read Parameter, reception	54	1011 0000	10 xxxxxx	xxxxxxxx		xxxxxxxx	xxxxxxxx	0 0 xxxxxx	Read Parameter, transmission
Idle_1	55	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1
Idle_1	56	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1
Idle_1	57	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1

Figure J.1 – Example for ISDU transmissions (1 of 2)

ISDURequest_2, transmission	58	0111 0000	10 xxxxxx	xxxxxxxx	0001 1011		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	59	0110 0001	10 xxxxxx	xxxxxxxx	Index		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	60	0110 0010	10 xxxxxx	xxxxxxxx	Data 1		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	61	0110 0011	10 xxxxxx	xxxxxxxx	Data 2		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	62	0110 0100	10 xxxxxx	xxxxxxxx	Data 3		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	63	0110 0101	10 xxxxxx	xxxxxxxx	Data 4		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	64	0110 0110	10 xxxxxx	xxxxxxxx	Data 5		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	65	0110 0111	10 xxxxxx	xxxxxxxx	Data 6		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	66	0110 1000	10 xxxxxx	xxxxxxxx	Data 7		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	67	0110 1001	10 xxxxxx	xxxxxxxx	Data 8		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	68	0110 1010	10 xxxxxx	xxxxxxxx	CHKPDU		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDUWait_3, start ISDU Timer	69	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUResponse_4, reception Stop ISDU Timer	70	1111 0000	10 xxxxxx	xxxxxxxx		0101 0010	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	71	1110 0001	10 xxxxxx	xxxxxxxx		CHKPDU	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
Idle_1	72	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1
Idle_1	73	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1
ISDURequest_2, transmission,	74	0111 0000	10 xxxxxx	xxxxxxxx	1011 0101		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	75	0110 0001	10 xxxxxx	xxxxxxxx	Index(hi)		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	76	0110 0010	10 xxxxxx	xxxxxxxx	Index(lo)		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	77	0110 0011	10 xxxxxx	xxxxxxxx	Subindex		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDURequest_2, transmission	78	0110 0100	10 xxxxxx	xxxxxxxx	CHKPDU		xxxxxxxx	0 0 xxxxxx	ISDURequest_2, reception
ISDUWait_3, start ISDU Timer	79	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUWait_3, inc. ISDU timer	80	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUWait_3, inc. ISDU timer	81	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUWait_3, inc. ISDU timer	82	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUWait_3, inc. ISDU timer	83	1111 0000	10 xxxxxx	xxxxxxxx		0000 0001	xxxxxxxx	0 0 xxxxxx	ISDUWait_3, application busy
ISDUResponse_4, reception Stop ISDU Timer	84	1111 0000	10 xxxxxx	xxxxxxxx		1101 0001	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	85	1110 0001	10 xxxxxx	xxxxxxxx		0001 1110	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, reception	86	1110 0010	10 xxxxxx	xxxxxxxx		Data 1	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, transmission
ISDUResponse_4, ABORT	87	1111 1111	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	ISDUResponse_4, ABORT
Idle_1	88	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1
Idle_1	89	1111 0001	10 xxxxxx	xxxxxxxx		0000 0000	xxxxxxxx	0 0 xxxxxx	Idle_1

Figure J.1 (2 of 2)

Annex K
(informative)

Recommended methods for detecting parameter changes

K.1 CRC signature

Cyclic Redundancy Checking belongs to the HASH function family. A CRC signature across all changeable parameters can be calculated by the Device with the help of a so-called proper generator polynomial. The calculation results in a different signature whenever the parameter set has been changed. It should be noted that the signature secures also the octet order within the parameter set. Any change in the order when calculating the signature will lead to a different value. The quality of securing (undetected changes) depends heavily on both the CRC generator polynomial and the length (number of octets) of the parameter set. The seed value should be > 0. One calculation method uses directly the formula, another one uses octet shifting and lookup tables. The first one requests less program memory and is a bit slower, the other one requires memory for a lookup table (1 x 2^{10} octets for a 32-bit signature) and is fast. The parameter data set comparison is performed in state "Checksum_9" of the Data Storage (DS) state machine in Figure 104. Table K.1 lists several possible generator polynomials and their detection level.

Table K.1 – Proper CRC generator polynomials

Generator polynomial	Signature	Data length	Undetected changes
0x9B	8 bits	1 octet	$< 2^{-8}$ (not recommended)
0x4EAB	16 bits	1 < octets < 3	$< 2^{-16}$ (not recommended)
0x5D6DCB	24 bits	1 < octets < 4	$< 2^{-24}$ (not recommended)
0xF4ACFB13	32 bits	1 < octets < 2^{32}	$< 2^{-32}$ (recommended)

K.2 Revision counter

A 32-bit revision counter can be implemented, counting any change of the parameter set. The Device shall use a random initial value for the Revision Counter. The counter itself shall not be stored via Index List of the Device. After the download the actual counter value is read back from the Device to avoid multiple writing initiated by the download sequence. The parameter data set comparison is performed in state "Checksum_9" of the Data Storage (DS) state machine in Figure 104.

Bibliography

[1] IEC 60050 (all parts), *International Electrotechnical Vocabulary,* (available at <http://www.electropedia.org>)

[2] IEC 60870-5-1:1990, *Telecontrol equipment and systems – Part 5: Transmission protocols – Section One: Transmission frame formats*

[3] IEC 61158-2, *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

[4] IEC/TR 62453-61, *Field device tool interface specification – Part 61: Device Type Manager (DTM) Styleguide for common object model*

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

[6] IO-Link Community, *IO Device Description (IODD), Order No. 10.012* (available at <http://www.io-link.com>)

[7] IO-Link Community, *IO-Link Common Profile, Order No. 10.072* (available at <http://www.io-link.com>)

[8] IO-Link Community, *IO-Link Communication, V1.0, January 2009, Order No. 10.002* (available at <http://www.io-link.com>)

[9] IO-Link Community, *IO-Link Test Specification, Order No. 10.032* (available at <http://www.io-link.com>)

[10] IO-Link Community, *IO-Link Safety System Extensions, Order No. 10.092* (available at <http://www.io-link.com>)

[11] IO-Link Community, *IO-Link Wireless System Extensions, Order No. 10.112* (available at <http://www.io-link.com>)

[12] IO-Link Community, *IO-Link Common Gateway Profile*, work in progress

____

IO-Link Community
Ohiostrasse 8
76149 Karlsruhe
Germany
Phone: +49 (0) 721 / 98 61 97 0
Fax: +49 (0) 721 / 98 61 97 11
e-mail: info@io-link.com
http://www.io-link.com/